USER MANUAL

Power PMAC IDE

Power PMAC

Integrated Development Environment

MN-000291

August 12, 2022

Single Source Machine Control *Power // Flexibility // Ease of Use*
9200 Oakdale Ave. Suite 900 Chatsworth, CA 91311
https://automation.omron.com/en/us/

Copyright Information

To report errors or inconsistencies, email:

Delta Tau Data Systems, Inc. Technical Support
Email : odt-support@omron.com
Web site : https://automation.omron.com/en/us/

Operating Conditions

All Delta Tau Data Systems, Inc. motion controller products, accessories, and amplifiers contain static sensitive components that can be damaged by incorrect handling. When installing or handling Delta Tau Data Systems, Inc. products, avoid contact with highly insulated materials. Only qualified personnel should be allowed to handle this equipment.

In the case of industrial applications, we expect our products to be protected from hazardous or conductive materials and/or environments that could cause harm to the controller by damaging components or causing electrical shorts. When our products are used in an industrial environment, install them into an industrial electrical cabinet or industrial PC to protect them from excessive or corrosive moisture, abnormal ambient temperatures, and conductive materials. If Delta Tau Data Systems, Inc. products are directly exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation.

Security Measures

To prevent computer viruses, install antivirus software on a computer where you use this software.

Make sure to keep the antivirus software updated.

Keep your computer's OS updated to avoid security risks caused by a vulnerability in the OS.

Always use the highest version of this software to add new features, increase operability, and enhance security.

Manage usernames and passwords for this software carefully to protect them from unauthorized uses.

Set up a firewall (E.g., disabling unused communication ports, limiting communication hosts, etc.) on a network for a control system and devices to separate them from other IT networks.

Make sure to connect to the control system inside the firewall.

Use a virtual private network (VPN) for remote access to a control system and devices from this software.

WARNING

A Warning identifies hazards that could result in personal injury or death. It precedes the discussion of interest.

Caution

A Caution identifies hazards that could result in equipment damage. It precedes the discussion of interest.

Note

A Note identifies information critical to the user's understanding or use of the equipment. It follows the discussion of interest.

REVISION HISTORY				
REV.	**DESCRIPTION**	**DATE**	**CHG**	**APPVD**
0	MANUAL CREATION	3/20/18	AS/DG	AG
1	FIRST ISSUE	4/4/18	TT	AG
2	CHANGES for IDE v4.1	07/03/18	TT	AG
3	UPDATED APPENDIX FOR UPGRADES	30/07/18	TT	AG
4	CHANGES for IDE V4.2	11/19/18	TT	AG
5	CHANGES for IDE V4.3	06/20/19	TT	AG
6	CHANGES for IDE V4.3.2.x			
7	CHANGES for IDE V4.4.0.x	04/02/2020	SB	AG
8	CHANGES for IDE V4.4.1.x	09/01/2020	BJ	AG
9	CHANGES for IDE V4.4.2.x	10/06/2020	DG	AG
10	CHANGES for IDE V4.5.0.x	03/31/2021	DG	AG
11	CHANGES for IDE V4.5.1.x	06/21/2021	DG	AG
12	CHANGES for IDE V4.5.2.x	10/08/2021	DG	AG
13	CHANGES for IDE V4.6.0.x	05/05/2022	DG	AH
C	CHANGES for IDE V4.6.1.x ADDED SECURITY MEASURES	08/12/2022	DG	AH

Table of Contents

INTRODUCTION

The Power PMAC Integrated Development Environment (IDE) software is based on the Visual Studio 2015 programming environment. It is used to develop, debug, and test Power PMAC programs developed in Delta Tau's proprietary Power PMAC Script language or in the C programming language. The programs are organized as a project that includes folders such as the Power PMAC Script Language, C Language, etc. The programming environment supports program debugging capabilities and allows the user to insert breakpoints and step through the program. It supports setup tools to detect, configure, and diagnose Power PMAC hardware through its System Setup utility. The Power PMAC IDE also supports setup of EtherCAT and MACRO devices.

This manual attempts' to thoroughly explain how to use the IDE and how to set up the system using the System Setup software. If, however any support is required please call Technical Support at 1(800) 556 6766 (Select 1 and then 6) or email: ODT-Support@Omron.com

SYSTEM REQUIREMENTS

Operating system
The Power PMAC IDE is an application that runs on Microsoft Windows ™.
It will run on the following versions of Microsoft Windows.

- Windows 10

The Power PMAC IDE requires .NET Framework 4.6 and above. The installation will identify the missing framework and installs it automatically.

Hardware

- 1.6 GHz or faster processor
- 4 GB of RAM (2 GB if running on a virtual machine)
- 20 GB of available hard disk space
- 5400 RPM hard disk drive or faster
- DirectX 9-capable video card (1024 x 768 or higher resolution)

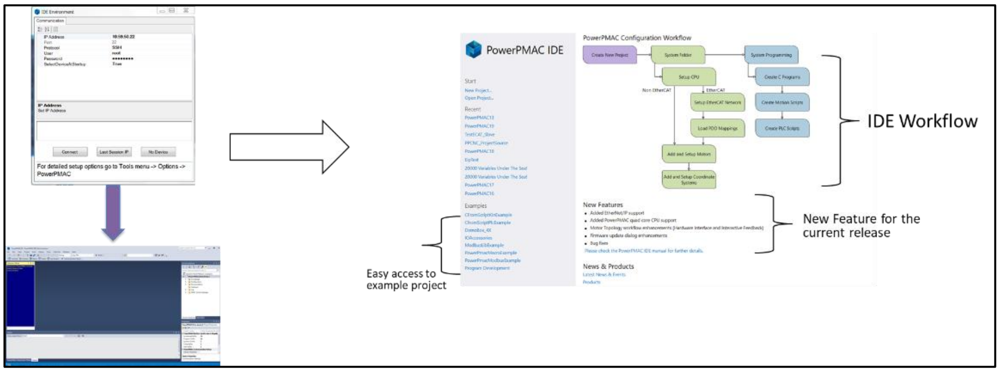
Note

The performance is directly dependent on the processor speed and RAM. Better the processor speed and bigger the RAM better performance.

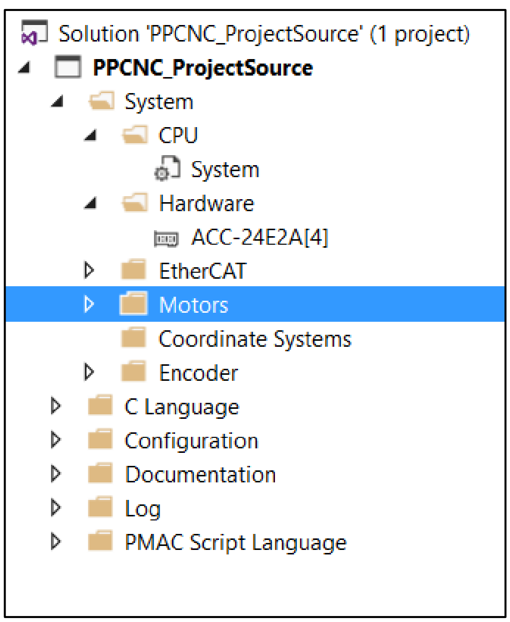
DIFFERENCES BETWEEN V3.X AND V4.X

Overview: Changes from V3.x

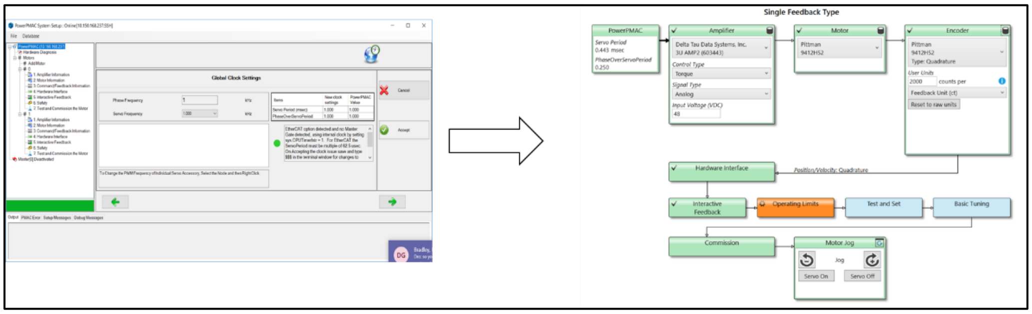
1. Intuitive Start Page saves User time and enhances configuration. Future extensible by connecting to the Delta Tau website for live updates and news.



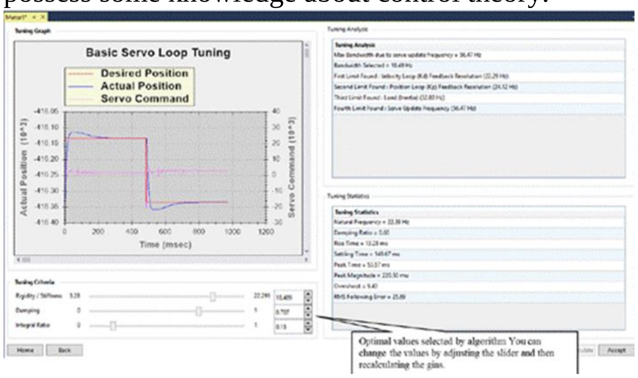
2. Open system configuration to project based configuration. In IDE 4.x all the Power PMAC configuration is in one place, Project, unlike in IDE3.x system setup, where EtherCAT setup is a separate application.



3. IDE 4.x automatically manages changes to Motor and Coordinate parameters through the user interface (not Terminal window) and creates the systemsetup.cfg file during build and download. In IDE3.x user has to maintain the configuration file manually.
4. Graphical/Intuitive motor Setup based on Topology (graphical view) and integrated with the project system, whereas in IDE 3.x it's a separate non-graphical application.

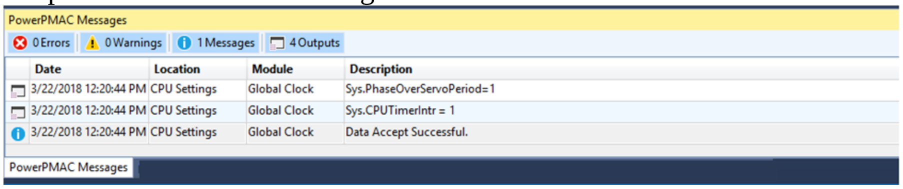


5. Coordinate system element setup integrated with Project system for usability.
6. Enhanced Basic Tuning: A major difference between V3.x and V4.x is the Servo loop tuning previously accessed from Test and Set. We now have a Basic Tuning block. The concept of the basic tuning is for new and basic Users. The tuning algorithm will achieve the performance, so they do not need to use advanced tuning. Advanced tuning is still available for expert users who possess some knowledge about control theory.



7. Intellectual property encryption support: IP (Intellectual property) protection allows OEM builders, independent integrators and users to protect their intellectual property by encrypting script programs. The encryption is password protected.
   The current implementation of IP protection is three level.
   a. Customer-A can encrypt the script programs and pass the project on to Customer-B. This is level one.
   b. Customer-B can take the project from Customer-A and add their own logic and protect it by encrypting to give it to Customer-C. This is level two. Customer-B cannot list or view Customer-A's code.
   c. Customer-C can take the Project from Customer-B and add their logic and protect their part by encrypting it to give to Customer-D. This is level three. Customer-C cannot list or view Customer-A's or Customer-B code.
   d. Customer-D cannot list or view Customer-A,B or C’s code.
8. Power PMAC messages window displays errors, warnings and messages. Parameter settings, motor setup, coordinate setup and EtherCAT setup write to this window.
   Error tab shows error.
   Warning tabs shows warning.
   Messages tab shows the messages.

Output tab shows all the settings that are written to Power PMAC.



9. Integrated EtherCAT setup into the project system for easy maintainance. EC-Engineer is integrated to project system unlike in IDE3.x, where it was a seperate application and required manual work to add EtherCAT configuration(ENI) to the project.

10. Bug fixes

Release notes V4.2

Reason: Bug fixes reported in Bugzilla, new feature addition and enhancement.
List of new feature and enhancement:

1. Support for exporting, Importing and deleting of custom project templates.
2. Support for Exporting, Importing and Deleting of Custom Item Templates. Example: Motor settings, CS settings etc.
3. Following folder nodes in the solution explorer will not automatically sort the file that are added or scan but will display as they are added or detected maintaining the sequence.
    a. Hardware card under Hardware node
    b. Motor under Motor node
    c. Coordinate systems under coordinate system node
    d. Script language files
    e. ECAT devices under EtherCAT node
4. PLC and Motion programs files can be move up or down
5. Improve Motor setup topology navigation adding support for Next and Prev state.
6. Clearly visible Alarm Indicator. Alarm provides symptom and possible remedy information.
7. New wizard style Image backup and restore.
8. New single Position window displaying Position, Velocity and Following Error.
9. New designed watch window.
10. New standard Toolbar with commonly used command
11. Improved EtherCAT error reporting while scanning and activating network.
12. Common connection title bar clearly indicating Power PMAC connection status
13. Advance tuning settings are exported to motor in the project.

Release notes V4.3

Reason: Bug fixes reported in Bugzilla, new feature addition and enhancement.
List of new feature and enhancement:

1. Update Amplifier, Motor and Encoder view compare to V4.2
2. Support Part Manager as menu so user can enter custom Amplifier, Motor or Encoder without using system setup.
3. Improve System –CPU block and categorizing it for usability
4. Update Global Clock page compare to V4.2.
5. Support Import and export of Encoder like Motor and Amplifier.
6. Topology view improvement to show clear flow.
7. Every Topology is appended with Jog Ribbon block for user to test the motor
8. Support New CK3M hardware: Digital IO (MDxx) and Analog IO (ADxx)
9. EtherCAT setup Enhancement
    a. Improve EtherCAT Motor topology
    b. Improve header file organization(.pmh and .h)

   c. Support naming a slave from EtherCAT network setup
   d. Improve error handling when scanning and enabling the EtherCAT network
   e. Support Slave template configuration for easy setup
   f. Support EtherCAT slave template import/export
   g. Support Slave disable
   h. EtherCAT variable viewer to support easy commissioning
   i. Visual indicator showing EtherCAT active status
10. Project compare and diff the files from project menu.
11. Support font size to Position and watch window.
12. Visual indicator for Build and Download completion
13. Improve Motor compare view showing Power PMAC defaults column.
14. Support commonly used Power PMAC commands on the toolbar.
15. New Project template for EtherCAT projects.

Release Notes V4.3.2.x

Reason: Bug fixes reported in Bugzilla, new feature addition and feature enhancement.
List of new feature and enhancement:

1. Support Drag and Drop EtherCAT Slave (1S and G5 only) to motor and setup EtherCAT motor.
2. Support Hot connect group for EtherCAT slave.
3. Enhance Project Compare functionality
   a. Expanding eni file to compare slaves.
4. Add support for Project Sync (copy from Power PMAC to PC)
5. Add Template Manager for Project and Item Templates
6. Enhance Motor/Co-ordinate System Compare view
7. Enhance the Topology view by adding a Safety Block
8. Add the ability to ‘refresh’ the Hardware Node

Release Notes V4.4.0.x

Reason: Bug fixes reported in Bugzilla, new feature addition and feature enhancement.
List of new features and enhancements:

1. Support for Ethernet IP (EIP) setup.(Available after July 2020)
2. QUAD core support
   a. Compile
   b. Compare System, Project settings for core task allocation and buffer settings.
   c. Core management
   d. Image restore
3. Simplified and unified communication setup dialog
4. Revamped Firmware update dialog and Package install dialog
5. Revamped Hardware interface and Interactive dialog from Motor topology view.
6. F1 help support extended to commissioning dialogs.
7. Revamped graph integrated to basic tuning and interactive feedback.

Release Notes V4.4.1.x

Reason: Bug fixes reported in Bugzilla, new feature addition and feature enhancement.
List of new features and enhancements:

1. Supporting Ethernet/IP (EIP) setup for CK3E and CK3M and UMAC CPU (Requires FW V2.6.x.x)
2. Support core management configuration for CK3M
3. Fix: The Data size is 0 in ethernetip.xml if the connection setting is disabled from EIP setup

4. Fix: EIP Watch variable window cannot be opened if multiple connections are configured
5. Fix: Plot control crashes
6. Fix: EtherNet/IP connection variable are not unique when copy paste connection command is used.
7. Fix: Power PMAC message window should not get focus while Build and Download is in progress.
8. Fix: Sometime PLC or Motion program does not show Motor or Coord or EIP structure in the project editor intellisense list
9. Fix: Delete EIP connection takes very long time.
10. Fix: PMAC IDE hangs when checking EtherNet/IP Watch window.(a large number of variables)
11. Fix: Remove the Dark theme option from Tool -Option-general
12. Fix: EIP Error message saying firmware 2.5.4 instead of 2.6
13. Fix: Block the build and download for EtherNet/IP project if the FW is V2.5.4.0
14. Fix: Task Manager Display goes wrong after automatically re-connection after disconnect.

Release Notes V4.4.2.x

1. QUAD core CPU support
2. EIP support for QUAD UMAC

Release Notes V4.5.0.x

Reason: New feature addition and feature enhancement.
Bug fixes

List of new features and enhancements:

1. Ck3WGCxxxx hardware support and configuration page
2. CK3WGCxxxx TCR application configuration page
3. Motor topology supports adding Virtual and galvo motor configuration
4. Improved Tuning user interface integrating new chart
5. EtherCAT setup improvement
    a. Easy OMRON Safety controller integration.
    b. Drag and drop Multiple Omron Slave drive (1S and G5) to Motor Node and automatically setup EtherCAT Motor
    c. Disable slave
    d. Support for (Hot Connect) Groups
6. Project wizard for generating project framework
7. Homing application configuration page
8. Gantry application configuration page
9. Compensation table integration in the project system
10. Compare Gate X saved structure element

Release Notes V4.5.1.x

Reason: Bug fixes

1. Alarm Pop-up continuously stealing the focus and making it unusable.
2. Alarm pop-up incorrectly showing the status. Alarms are for error only. For example Plc[1].Ldata.Status = Stopped on Quit or CoordExecStatus[1] = Stopped on Quit, is not an Alarm but status.

Release Notes V4.5.2.x

Reason: New feature addition and feature enhancement.
Bug fixes

List of new features and enhancements:

1. Update MATLAB connectivity support to MATLAB 2020b version
2. Support for setting 16 KHz servo frequency for EtherCAT drives that supports 16 KHz

3. EtherCAT Analyzer
    a. Bus Mismatch
    b. Line Cross
4. Support Power Brick-stepper motor w/and w/o encoder from Motor Topology.
5. Enhancing Complete project upload from Power PMAC
6. Supporting expression evaluator from Encoder topology block for entering user units.
7. Supports storing the Tuning filter values to Power PMAC IDE project.

Release Notes 4.6.0.x

Reason: New feature addition and feature enhancement.
Bug fixes

List of new features and enhancements:

1. EtherCAT setup improvements:
    a. EtherCAT stack upgrade to 3.0.13.0
    b. EtherCAT slave enable/disable
    c. EtherCAT cable redundancy
    d. Simplified import of Sysmac Studio safety mapping files
2. CK3C hardware support and configuration
3. New compensation table setup page
4. Communication improvements
5. Updated project workflow on startup page
6. New CK3W-AX1515/ACC-24E3 hardware diagnostic page
7. Interactive feedback verification section
8. Consolidated motor commissioning page
9. Bug fixes

Release Notes 4.6.1.x

Reason: New feature addition and feature enhancement.
Bug fixes

List of new features and enhancements:

1. CK5M hardware support and configuration.
2. User interface and ease-of-use improvements.
3. Enhanced stability and improved safeguards.

Installation compatibility chart

Case1: User has PowerPMAC IDE V2.x on the machine.
Upgrade to Power PMACIDE V3.x: Requires complete uninstallation of Power PMAC IDE V2.x
Upgrade to Power PMACIDE V4.x: Requires complete uninstallation of Power PMAC IDE V2.x

Case2: User has PowerPMAC IDE V3.x on the machine
Upgrade to PowerPMAC IDE V4.x: Install the V4.x. There is NO NEED to uninstall the V3.x PowerPMAC IDE. PowerPMAC IDE V3.x and PowerPMAC IDE V4.x can run side-by-side.

IDE and Firmware Selection chart

Power PMAC 460/465

IDE \ FW	FW 2.3.2.5 Kernal mode (All CPU)			FW 2.4.1.2 (All CPU)			FW 2.5.0.3 (All CPU)			FW 2.5.2.0 (All CPU)		
	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program
PowerPMAC IDE 3.1.4.0	OK	OK	OK	OK	OK	OK	Not Supported	OK	OK	Not Supported	OK	OK
PowerPMAC IDE 4.1.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.2.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.4.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.1.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

CK3E

IDE \ FW	FW 2.3.2.5 User mode			FW 2.4.1.2			FW 2.5.0.3			FW 2.5.1.7			FW 2.5.4.0		
	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program
PowerPMAC IDE 3.1.4.0	OK	OK	OK	Not Supported	OK	OK	Not Supported	OK	OK	Not Supported	OK	OK	Not Supported	OK	OK
PowerPMAC IDE 4.1.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.2.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.4.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.1.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

IDE \ FW	FW 2.5.6.0			FW 2.6.0.0			FW 2.6.1.0			FW 2.7.0.0		
	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program
PowerPMAC IDE 3.1.4.0	Not Supported	OK	OK	Not Supported	OK	OK	Not Supported	OK	OK	Not Supported	OK	OK
PowerPMAC IDE 4.1.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.2.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.4.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.0.x												
PowerPMAC IDE 4.6.1.x												

CK3M

IDE \ FW	FW 2.4.1.2			FW 2.5.0.3			FW 2.5.2.0			FW 2.5.4.0		
	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program
PowerPMAC IDE 4.1.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.2.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.4.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.1.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

IDE \ FW	FW 2.5.6.0			FW 2.6.0.0			FW 2.6.1.0			FW 2.7.0.0		
	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program
PowerPMAC IDE 4.1.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.2.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.4.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.1.x	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

UMAC QUAD-ARM

IDE \ FW	FW 2.6.0.0			FW 2.6.1.0			FW 2.7.0.0		
	User Algo	Background C	Script program	User Algo	Background C	Script program	User Algo	Background C	Script program
PowerPMAC IDE 4.1.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.2.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.4.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.x.x	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.5.2.x	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.0.x	OK	OK	OK	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.6.1.x	OK	OK	OK	OK	OK	OK	OK	OK	OK

IPC LX86

IDE \ FW	FW 2.5.1.5			FW 2.5.2.0		
	User Algo	Background C	Script program	User Algo	Background C	Script program
PowerPMAC IDE 4.1.0.x	NA	NA	NA	NA	NA	NA
PowerPMAC IDE 4.2.0.x	NA	NA	NA	NA	NA	NA
PowerPMAC IDE 4.3.0.x	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.3.2.x	OK	OK	OK	OK	OK	OK
PowerPMAC IDE 4.4.x.x	OK	OK	OK	OK	OK	OK

CPCI

IDE \ FW	FW 2.7.0.0		
	User Algo	Background C	Script program
PowerPMAC IDE 4.6.0.x	OK	OK	OK
PowerPMAC IDE 4.6.1.x	OK	OK	OK

CK3C

IDE \ FW	FW 2.7.0.0		
	User Algo	Background C	Script program
PowerPMAC IDE 4.6.0.x	OK	OK	OK
PowerPMAC IDE 4.6.1.x	OK	OK	OK

EtherCAT stack Support

	EtherLAB stack setup support	Acontis stack setup support
PowerPMAC IDE 3.x	OK	OK (EC-Engineer external)
PowerPMAC IDE 4.x	Not Supported	OK (Integrated EC-Engineer)

Note

Recommended: Use or upgrade IDE4.x with FW version 2.4.x or above

KNOWN INSTALLATION ISSUES CAUSED BY ANTIVIRUS SOFTWARE

Issue: Customers experienced the issue in installing the Power PMAC IDE V2.x, V3.x or V4.x.
Cause: There are two virus scan software packages that, as of today, are known to cause incorrect installation of Power PMAC IDE. These are:

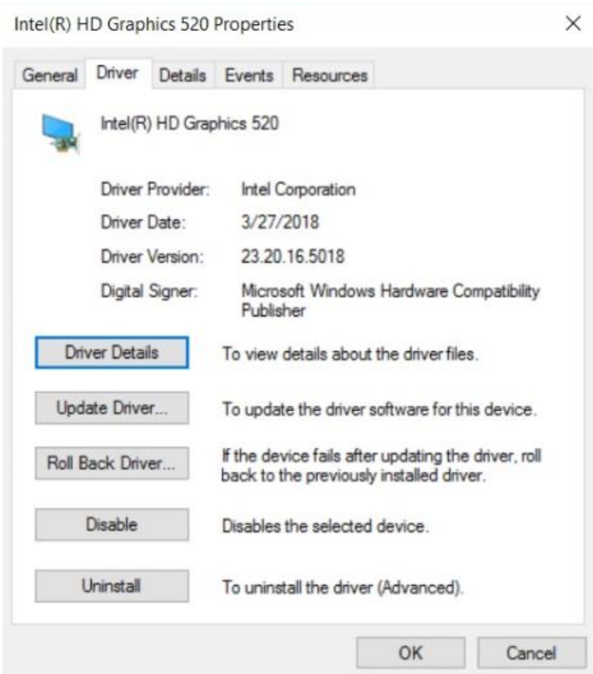
1. Avast Antivirus software
2. Sophos Antivirus software.

DISPLAY ADAPTER COMAPTIBILITY ISSUE

Issue: Customers experience build and download error because of incompatibility with display adapter driver and Cygwin.
Typical error looks like this…

```
Synchronizing Database failed.
Could not map PowerPMAC variables.
C:.....\PowerPMAC2.ppproj(133,5): error : reside in x:\cygwin\bin, where 'x' is the drive on which you have
C:.....\PowerPMAC2.ppproj(133,5): error :      1 [main] make 11116 fork: child -1 - forked process 11936 died unexpectedly, retry 0, exit code 0xC0000142, errno 11
C:.....\PowerPMAC2.ppproj(133,5): error : make: vfork: Resource temporarily unavailable
```

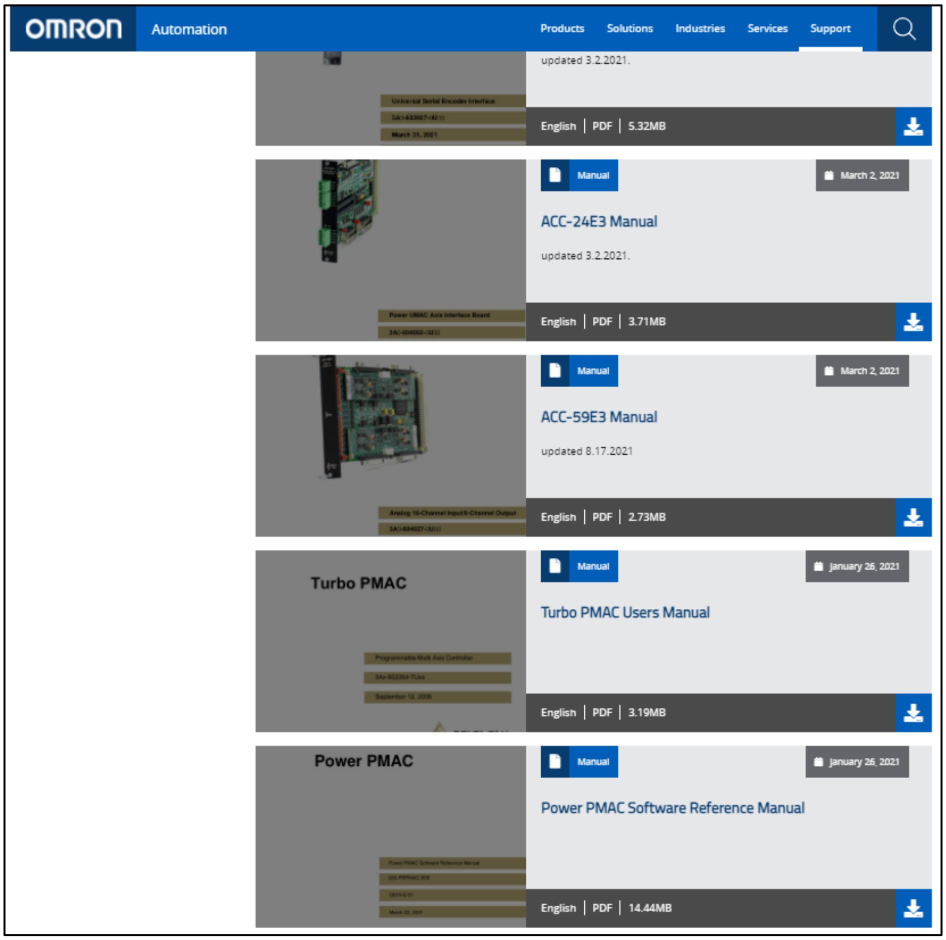
Observed error with Intel ® HD graphics 520.
Solution: Update the device driver. After updating the driver device manager looks like…


Intel(R) HD Graphics 520 Properties
General
Driver
Details
Events
Resources
Intel(R) HD Graphics 520
Driver Provider:
Intel Corporation
Driver Date:
3/27/2018
Driver Version:
23.20.16.5018
Digital Signer:
Microsoft Windows Hardware Compatibility Publisher
Driver Details
To view details about the driver files.
Update Driver...
To update the driver software for this device.
Roll Back Driver...
If the device fails after updating the driver, roll back to the previously installed driver.
Disable
Disables the selected device.
Uninstall
To uninstall the driver (Advanced).
OK
Cancel

OBTAINING THE POWER PMAC MANUALS

The Power PMAC User Manual and the Power PMAC Software Reference Manual on OMRON automation website.
[Industrial Automation | Omron](https://automation.omron.com/en/us/)

https://automation.omron.com/en/us/

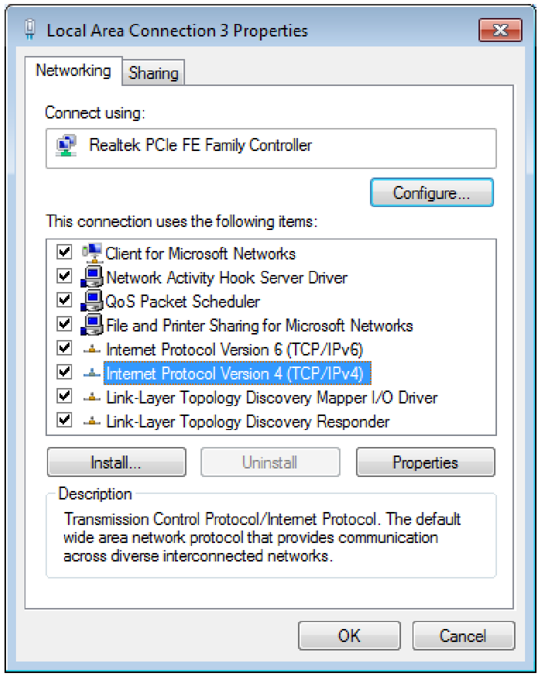


COMMUNICATING WITH POWER PMAC

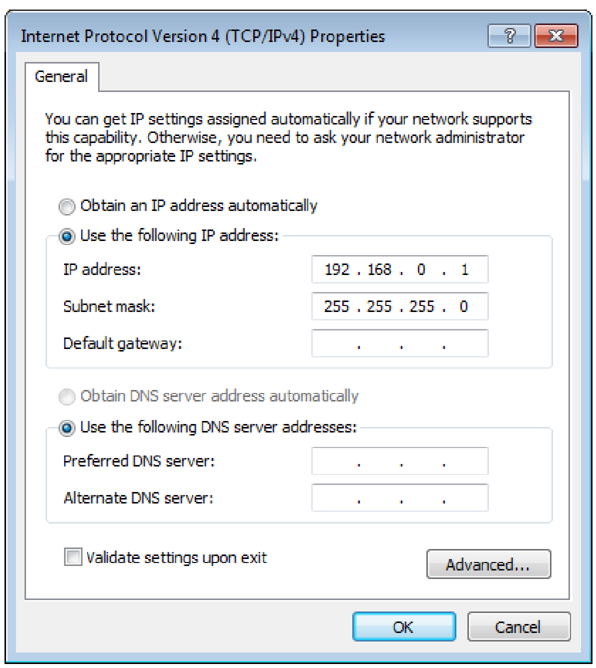
Establishing Communication

Connect the power to the Power PMAC Rack if it is not yet connected. Then connect an Ethernet Cable to the connector on the Power UMAC CPU labeled **ETH 0**, an Ethernet connector on the front of the Power UMAC CPU card, as highlighted by a red circle in the image of an example Power PMAC rack below:

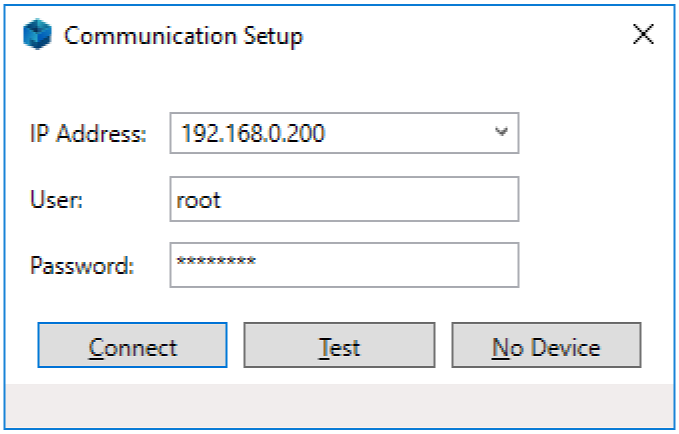
A PC can be connected to Power PMAC directly via a crossover cable, a straight cable or through a network switch. If using a network card dedicated for Power PMAC communication, and thus are connecting directly from a PC's network card to the Power PMAC, then set up a static IP for that network adapter on the same subnet as Power PMAC's IP address. In Windows 7 this can be achieved by clicking Start→Control Panel and then clicking on Network and Sharing Center. Then click on "Change adapter settings" which is usually in the leftmost pane of the window. Right-click the adapter that has been connected to Power PMAC and then click "Properties." Click Internet Protocol Version 4 (TCP/IPv4) and then click Properties:



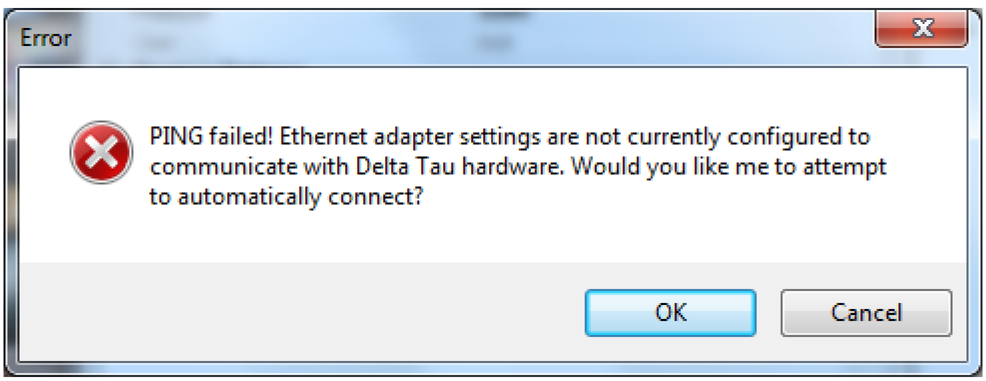
Click "Use the following IP address" and choose an IP address on the same subnet as the Power PMAC. An example is shown in the following screenshot:



Start the IDE by double-clicking the  desktop icon. On startup a valid IP address is required to communicate. The factory default address for Power PMAC is always **192.168.0.200.** Input the default IP address and press Connect.



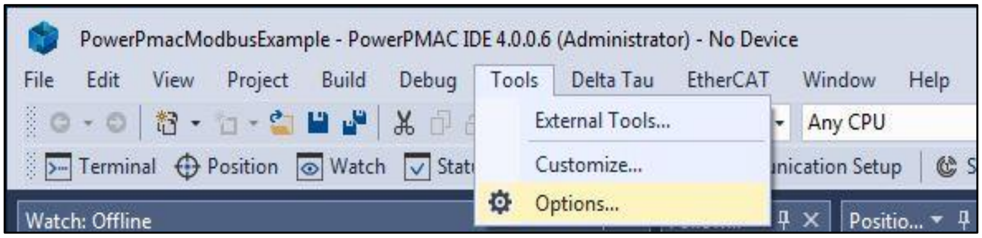
Upon connecting the IDE will try to communicate with Power PMAC. If this is the first time the PC is communicating with Power PMAC, and if using a network switch or hub and the PC is not on the same subnet as Power PMAC, then the routing question dialog box will appear asking for automatic configuration of the PC network settings (see screenshot below). Press OK to continue.



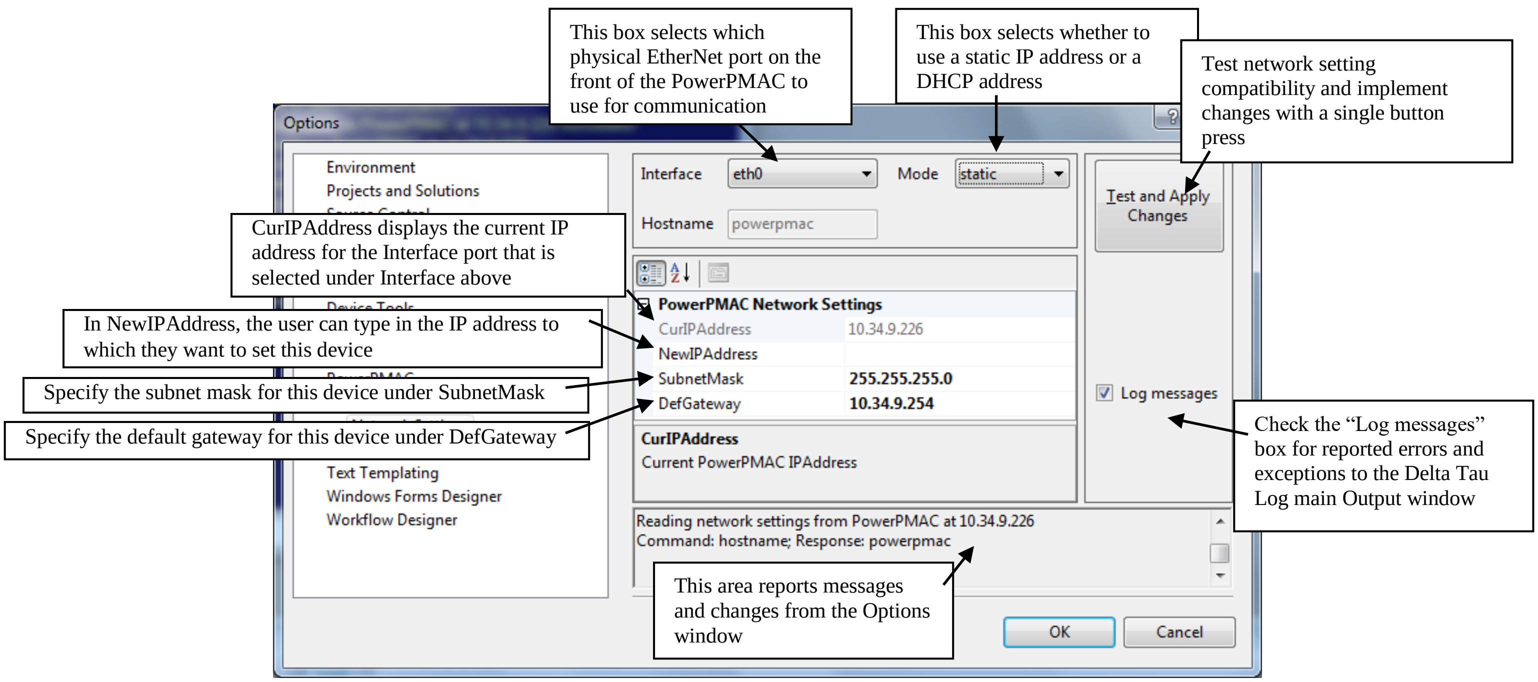
Upon successfully connecting the IDE will open in a default layout displaying the IP address.

Changing Power PMAC's Network Settings

To change Power PMAC's IP Address from within the IDE, click Tools→Options…



Near the bottom of the screen in the left pane, click Power PMAC→Network Settings and then the following window should appear, whose functions are annotated below:



Note

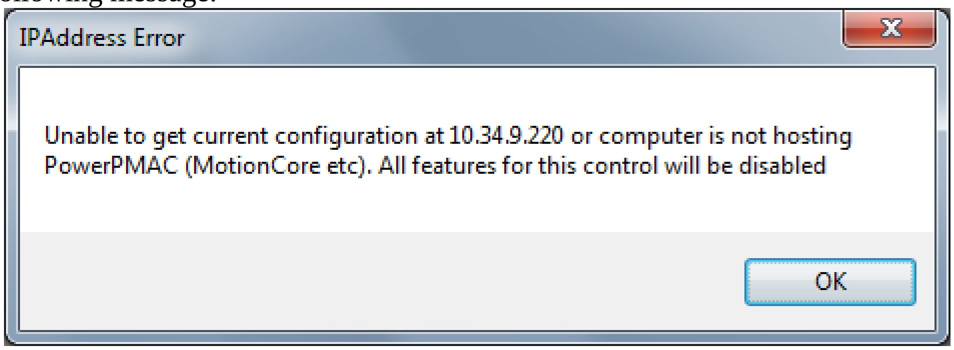
For the CPU types PowerPC, 460EX, if the 2nd interface "eth1" has been preconfigured the above screen can be used to change its settings.

For the CPU type PowerPC, APM86xxx (Dual-Core Power PMAC), the 2nd interface (if present) has been preconfigured as an EtherCAT device and therefore is not available as a 2nd LAN device for communication.

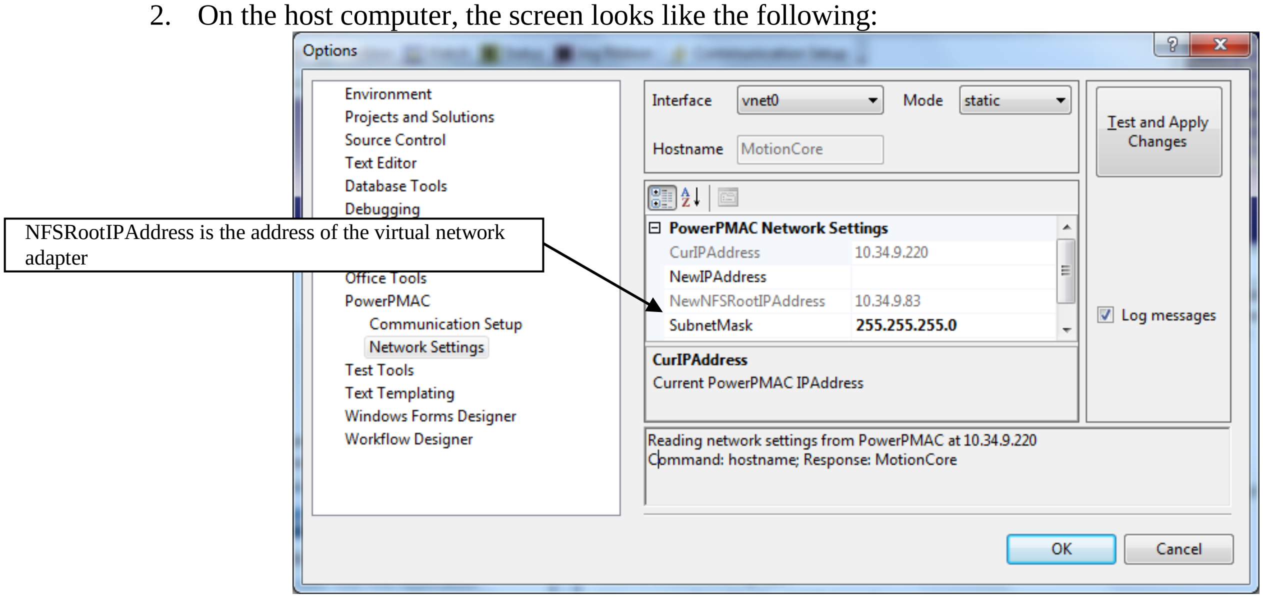
Changing x86, Hypervisor's (MotionCore's) Network Settings

For CPU type "x86, Hypervisor" MotionCore the screen looks slightly different and the procedure is as follows:

1. First, Network settings can only be changed locally on the host computer. Otherwise, the user will get the following message:



2. On the host computer, the screen looks like the following:



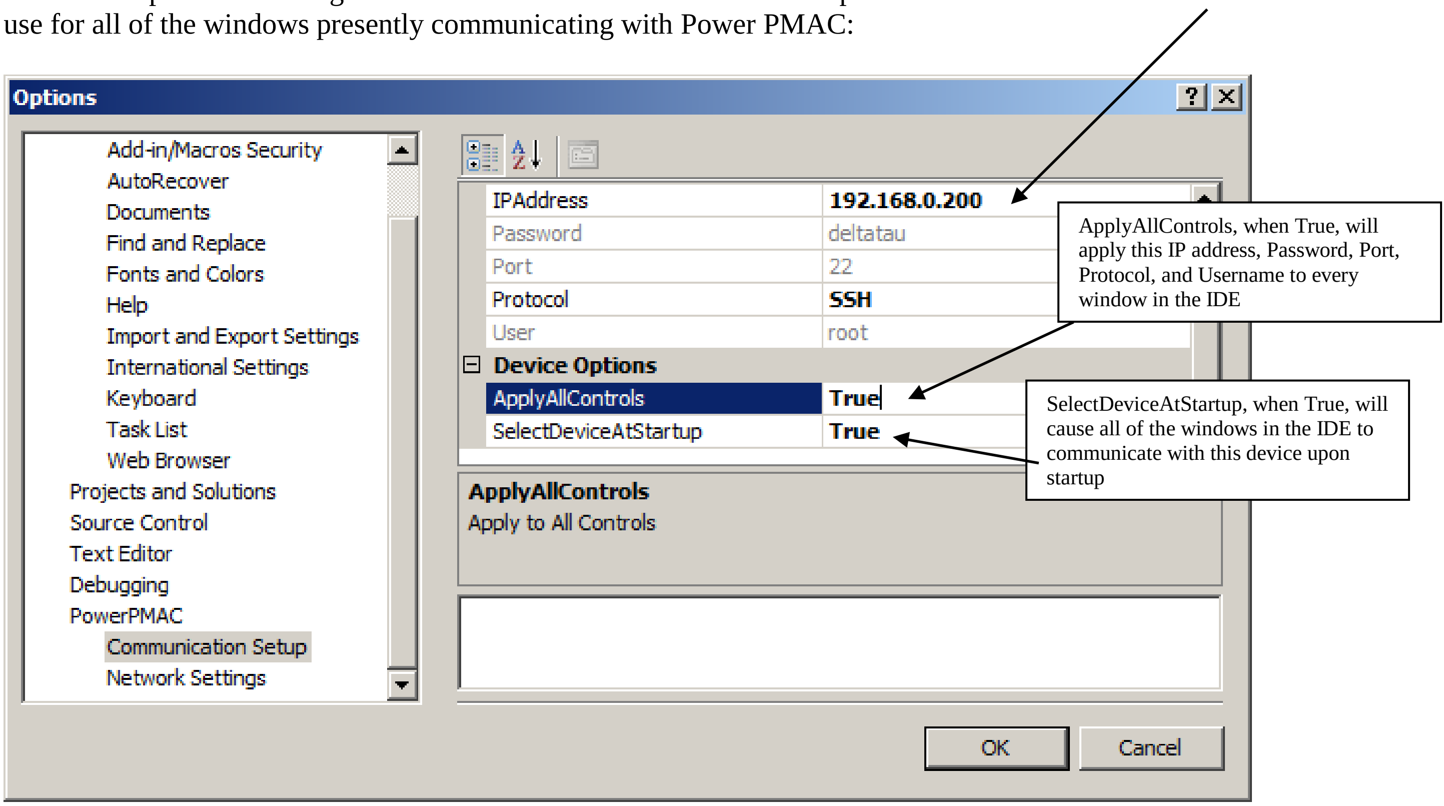
3. In x86,Hypervisor, the additional parameter NFSRootIPAddress is the IPAddress of the virtual network adapter. This address controls the availability of an IP Address subnet. The user cannot change the IP Address to a different subnet than one given by the NFSRoot subnet. The rest of the procedure is same as that of a regular Power PMAC.

4. Delta Tau is in the process of making an additional network interface available so that users will be able to connect to the x86,Hypervisor "MotionCore" externally. We will notify users when that interface is available and will promptly write the procedure as well.

Note

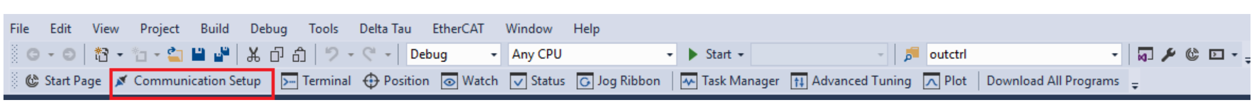
For CPU type "x86" Linux computers hosting the MotionCore the above options are disabled. All Test and Apply buttons are disabled as well. The following message will be displayed in this case: "Network settings change options are not available for CPU type: "x86" at this time."

Under the Options window go to Power PMAC→Communication Setup and select which IP address to use for all of the windows presently communicating with Power PMAC:

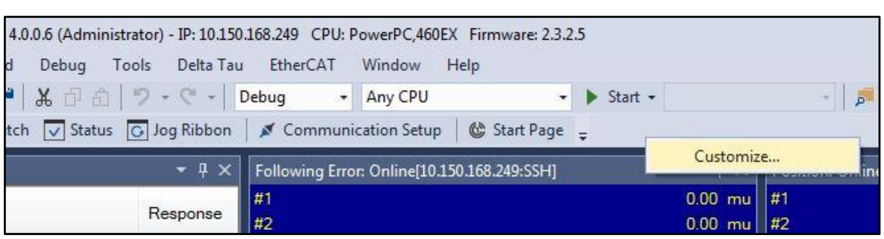


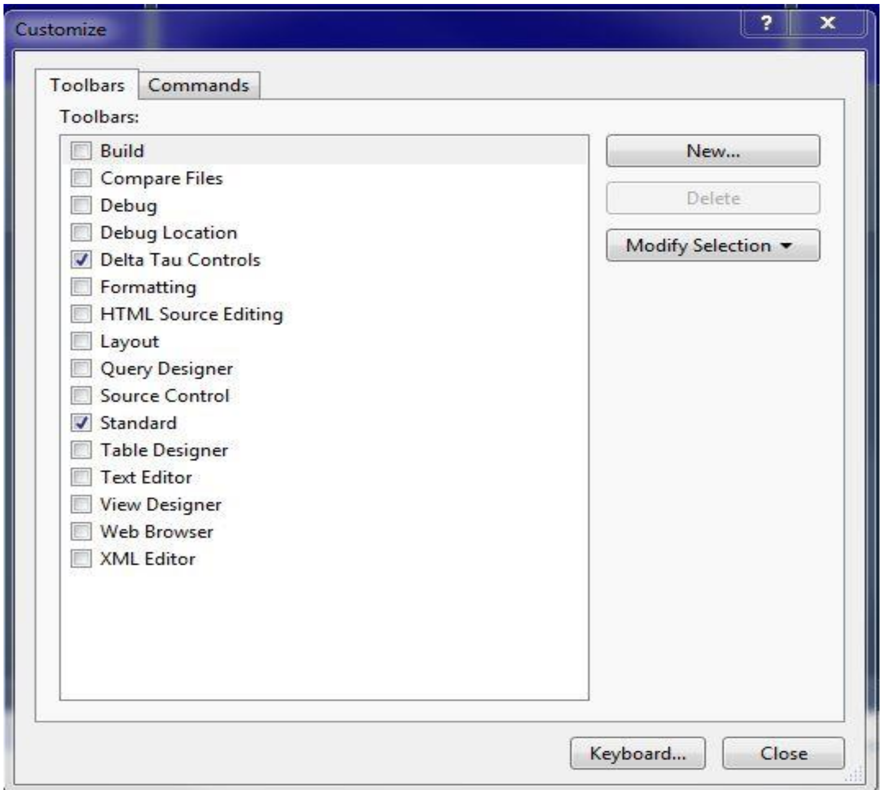
Re-establishing Communication

To re-establish communication click on the Communication Setup button (surrounded by a red box in the image below), which is shown on the Delta Tau Controls Toolbar:



If this button is not showing, right click on a blank, gray space in that toolbar area , go to Customize and make sure “Delta Tau Controls” is checked, as shown below:





Re-establishing communication can also be achiveved through the Communication Setup area of the Options window as described in the section of this manual immediately before this section.

IDE PROJECT EXAMPLES

Several example projects can be found in the Power PMAC IDE's installation folder. By default, its location is as follows:

C:\DeltaTau\PowerPMAC IDE\x\IDE\PowerPMACProjectExamples where x is the main version number i.e. 3, 4 etc.

Currently there are six examples included:

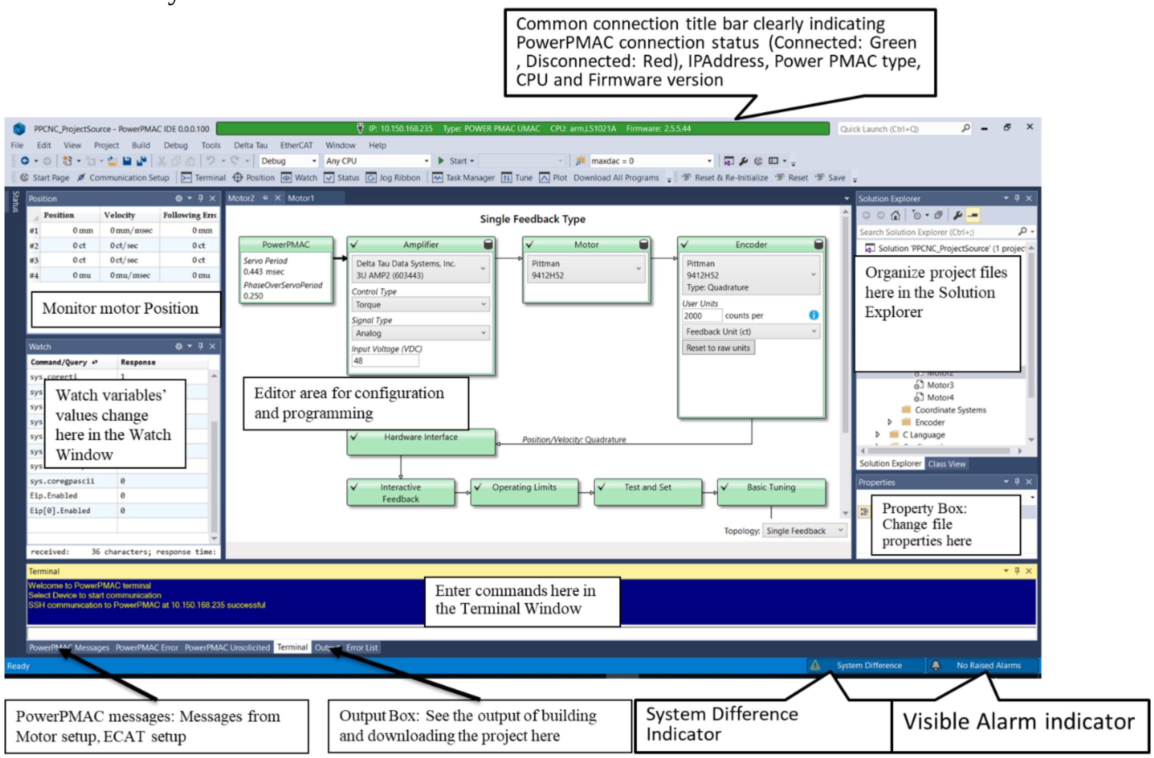
Project Folder Name	Description
DemoBox_4X	Basic motor setup for four Brush DC motors in a single coordinate system, a PLC, and some subprograms.
IOAccessories	Provides header templates for some I/O Accessories and a sample PLC for MACRO. Some are for local and remote (via MACRO 8x & 16x Stations) UMAC cards and some are standalone MACRO Stations.
ModbusLibExample	Sample for making a C library using Modbus as an example.
PowerPmacMacroExample	Example Script and C PLCs for MACRO communication.
PowerPmacModbusExample	PMAC Script and C application for communicating as a Modbus Client to a Modbus Server. Both the Modbus Client and Server are being executed on the PowerPMAC.
Program Development	This sample project and its documentation will explain and give examples of what to put in each folder of the Project Manager. Provides some example programs of different types.
CfromScriptKinExample	Shows how to implement Script Kinematic equations C in usercode.c's CfromScript function.
CfromScriptPlcExample	Shows how to use the CfromScript function in a Real-Time CPLC and in a Background BGPLC. This example also shows how to return data from the CfromScript function.
EipArrayExample	Example project on reading and writing EipArray part of Background Programs. This is C program example transferring Eip data block.

Note that within the Documentation folder in each of these example projects there is a text file explaining the purpose of the project and how to run it.

IDE LAYOUT

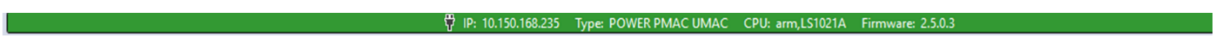
Default Layout

The default layout of the IDE screen is shown below:

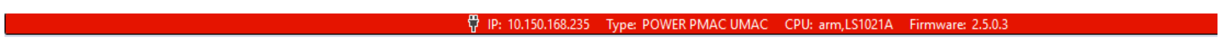


The common connection bar will indicate the connection status of the IDE to the PMAC. Below are the three states for this connection bar:

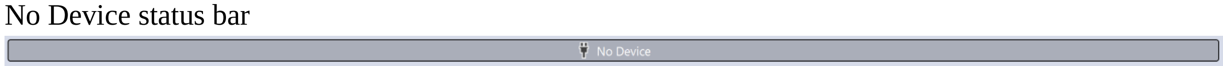
Connected status bar



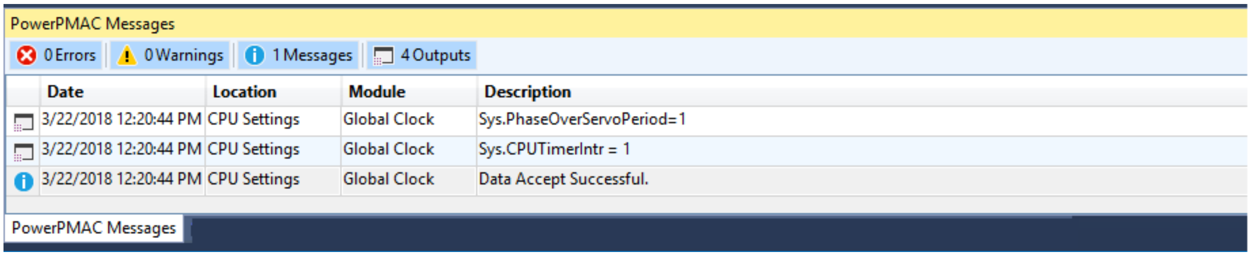
Disconnected status bar



No Device status bar



The Power PMAC messages window displays errors, warnings, messages, parameter settings, motor setup, coordinate setup and ECAT setup writes to this window.
The Error tab shows errors.
The Warning tabs shows warnings.
The Messages tab shows the messages.
The Output tab shows all the settings that are written to Power PMAC.



PowerPMAC Messages

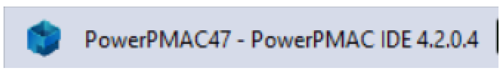
0 Errors | 0 Warnings | 1 Messages | 4 Outputs

Date	Location	Module	Description
3/22/2018 12:20:44 PM	CPU Settings	Global Clock	Sys.PhaseOverServoPeriod=1
3/22/2018 12:20:44 PM	CPU Settings	Global Clock	Sys.CPUTimerIntr = 1
3/22/2018 12:20:44 PM	CPU Settings	Global Clock	Data Accept Successful.

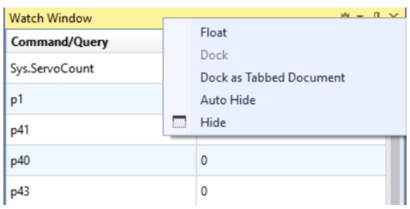
PowerPMAC Messages

The IDE title bar will display the following information:

- IDE version
- Currently open project



Windows can be moved around by clicking and dragging. Right-click the top of a window to choose to float the window, dock it, tab it, hide it automatically or hide it as shown below:



This is common to all windows in the IDE. The Auto Hide function will only appear if this document is tabbed.

Note

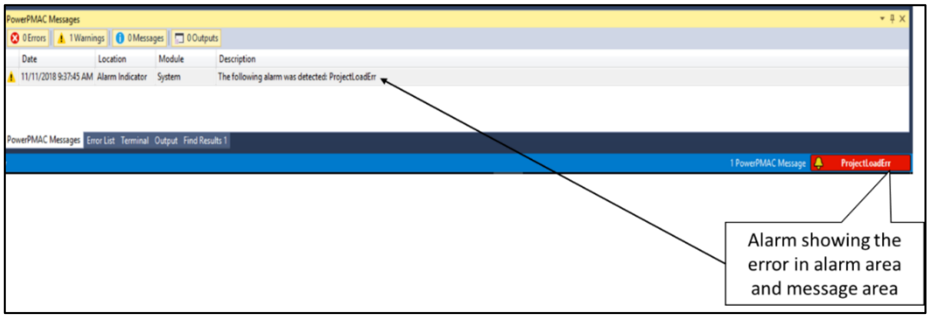
There is now a title bar indicator to display the device connection status. All individual connection information from control is removed

Alarm Indicator

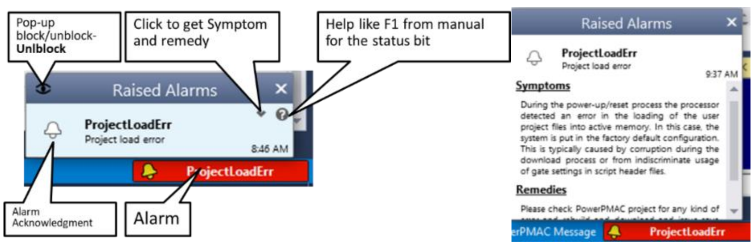
The Alarm indicator is always visible to clearly indicate to the user any Alarm as they are triggered. This view monitors the global status elements (Sys.status). This can be also found in Status window – Global Status Tab.

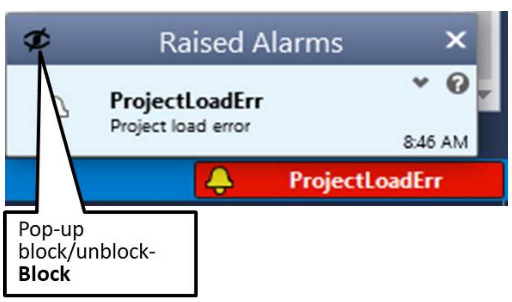
A lost Connection to Power PMAC is also treated as an Alarm and will be indicated, along with RED bar on the top of the IDE, and displayed in the alarm area as shown below:

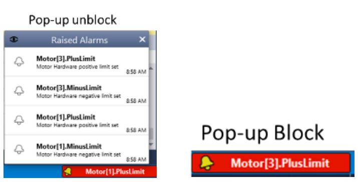
If there is an error other than loss of connection it will be displayed in the alarm area and a message will be displayed in the Power PMAC message area as shown below.



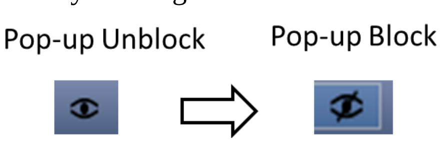
The User can acknowledge theses Alarms, but the Alarms are not removed from the view until they are cleared. The Alarm view shows the symptom of the alarm and possible remedy as shown below:







Default Pop-up blocker is in unblock state so user will see alarms stack-up. To stop this select Pop-up block by clicking EYE icon as shown below



Pop-up block/unblock is per IDE session. User will need to Block pop-up every time IDE is restarted.

System Difference Indicator

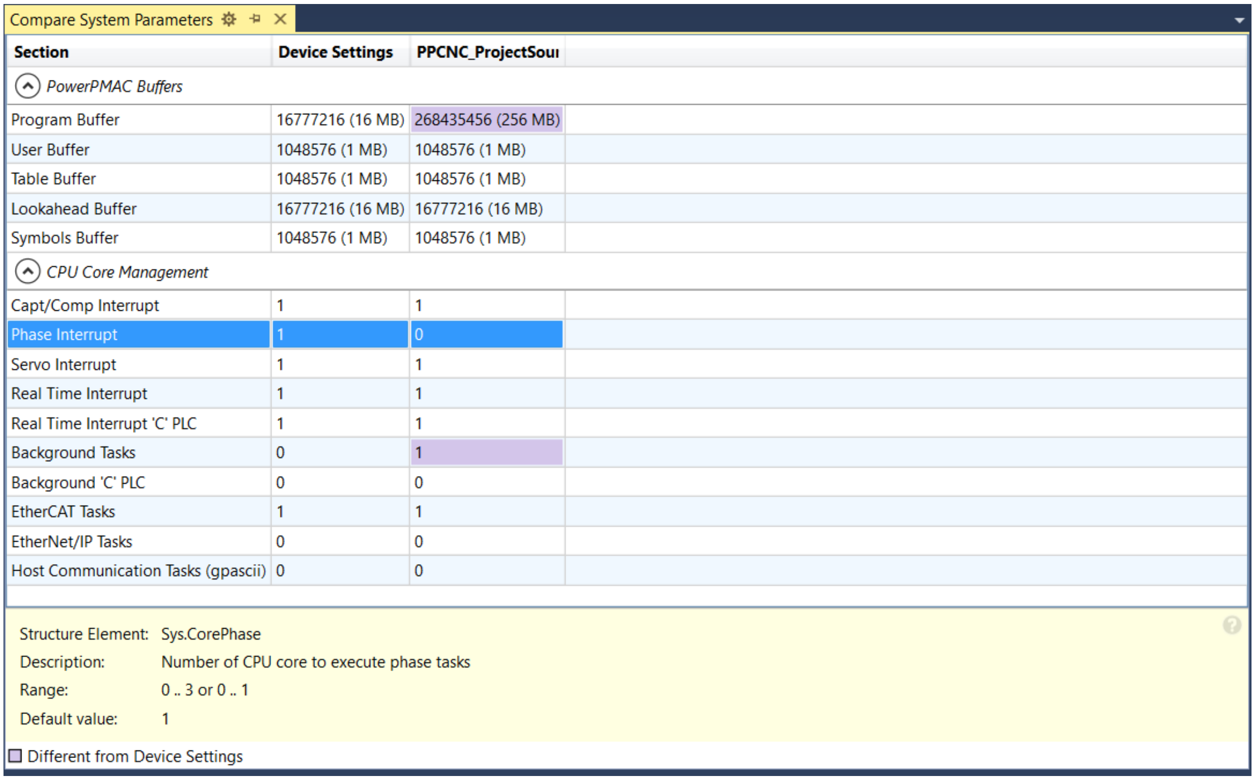
This is new indicator, as shown below, was added in IDE V4.4. It indicates that there is a difference between Power PMAC device settings and currently opened project. If the mouse is hovered over the indicator, it will provide a tooltip. The indicator is automatic and compares the Power PMAC device buffer settings and core assignment settings.

Power PMAC ARM CPU	Compares buffers and core assignment
Other CPU	Buffers only
IPC (Hypervisor)	Not supported

Note

The system difference requires the FW version 2.6.x and above.

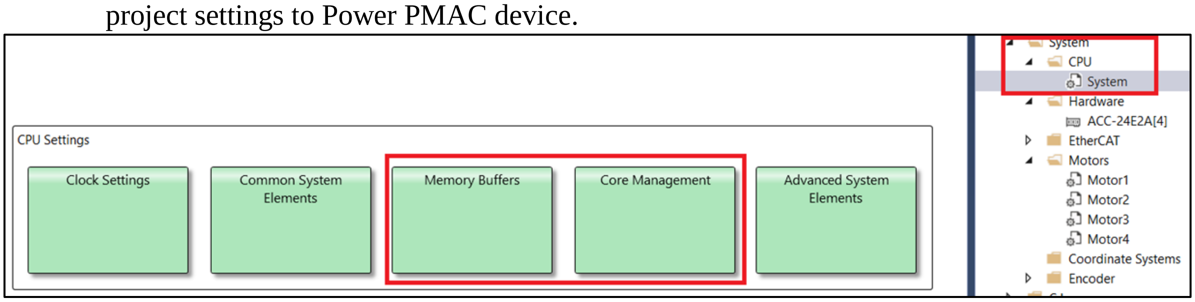
On clicking the System Difference it will show the difference window, as shown below…



Section	Device Settings	PPCNC_ProjectSou
PowerPMAC Buffers		
Program Buffer	16777216 (16 MB)	268435456 (256 MB)
User Buffer	1048576 (1 MB)	1048576 (1 MB)
Table Buffer	1048576 (1 MB)	1048576 (1 MB)
Lookahead Buffer	16777216 (16 MB)	16777216 (16 MB)
Symbols Buffer	1048576 (1 MB)	1048576 (1 MB)
CPU Core Management		
Capt/Comp Interrupt	1	1
Phase Interrupt	1	0
Servo Interrupt	1	1
Real Time Interrupt	1	1
Real Time Interrupt 'C' PLC	1	1
Background Tasks	0	1
Background 'C' PLC	0	0
EtherCAT Tasks	1	1
EtherNet/IP Tasks	0	0
Host Communication Tasks (gpascii)	0	0

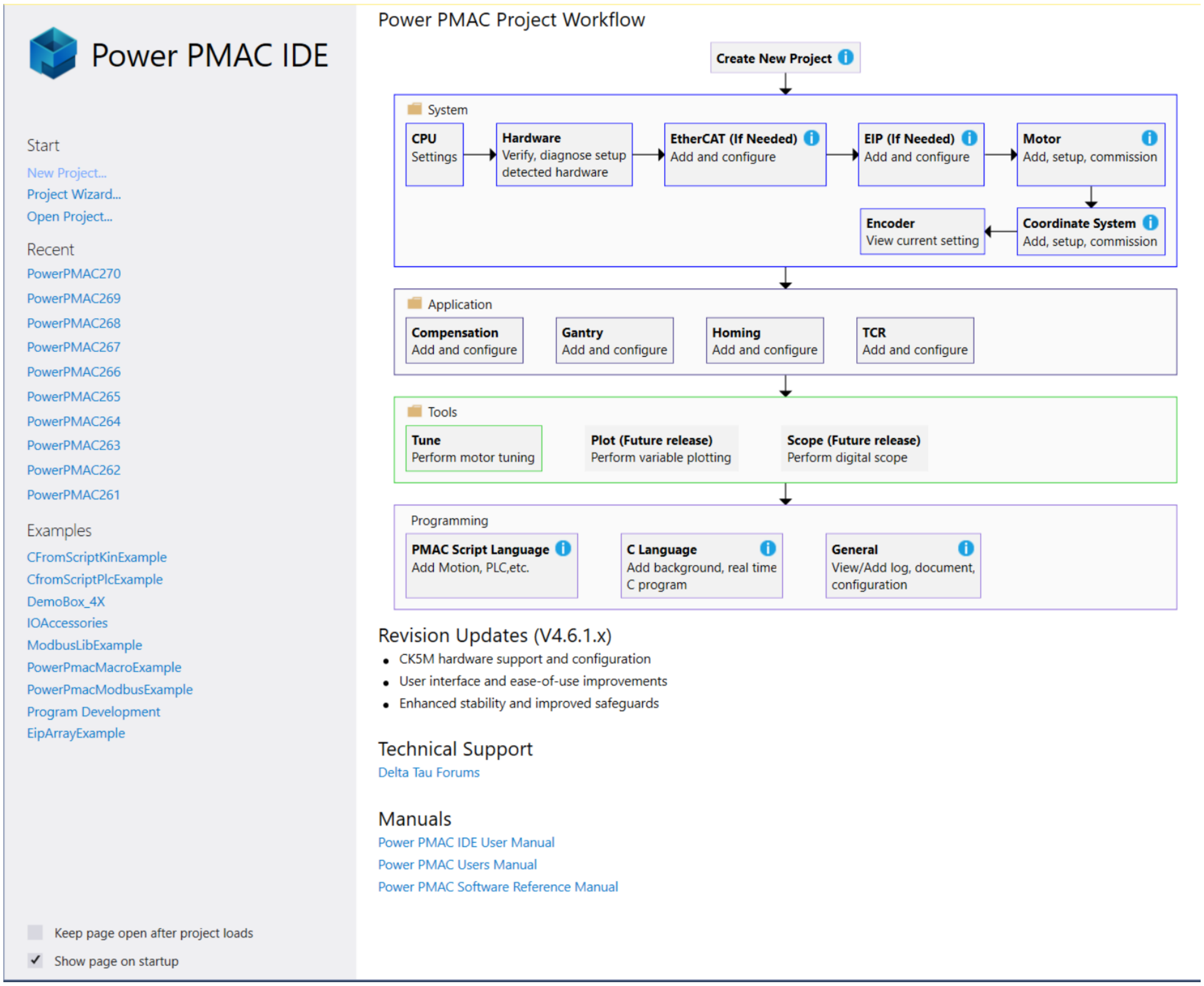
At this point the User has two choices:

1. To match what is on the Power PMAC by going to core management (System-CPU-System-Core management) UI and select the core assignment, and then selecting Memory buffers to match the Power PMAC device settings. (As shown below)
2. Build and download the project, save the project and reboot Power PMAC to apply current project settings to Power PMAC device.



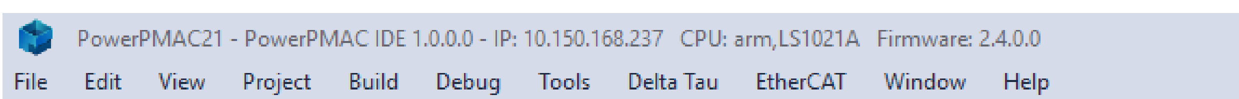
Start Page

The Start page is displayed by default when the IDE is first started after installation. The Power PMAC configuration workflow guides new users on how to use the IDE for configuration and programming. The page displays useful information about Delta Tau products, how to get technical support, etc. Users can disable the start page from being shown when the IDE launches from Tools> Options>Environment>Startup, or by unchecking the checkbox on the lower-left of the page.



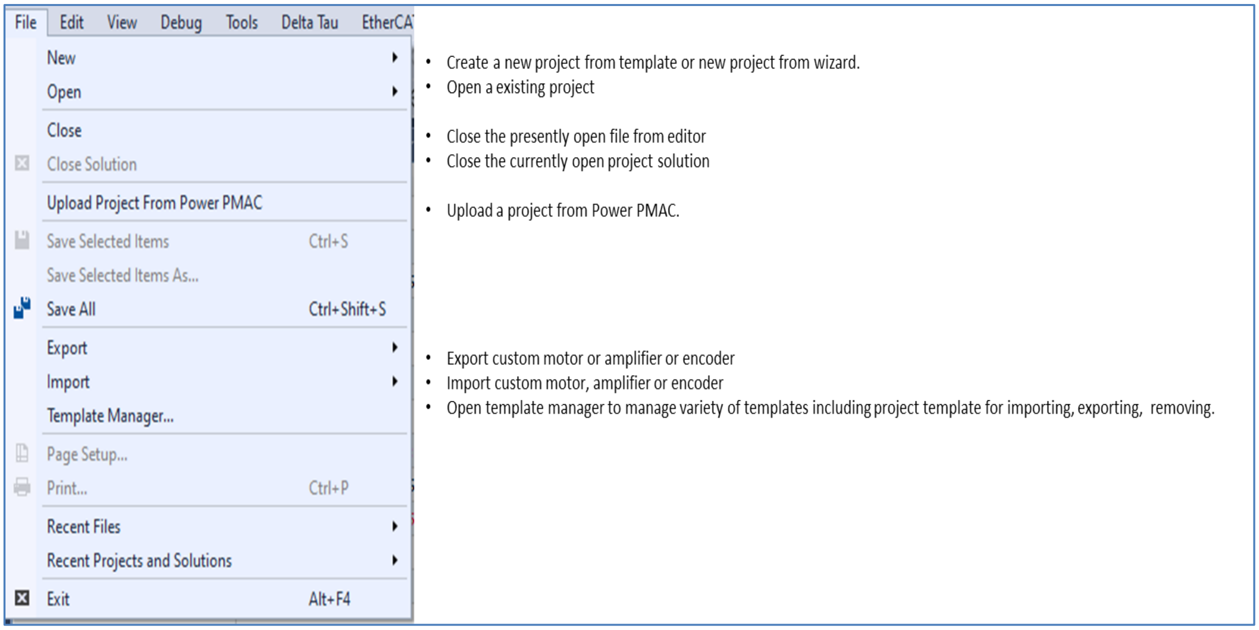
MENUS

The IDE has eleven dropdown menus at the top of its main screen as shown below:



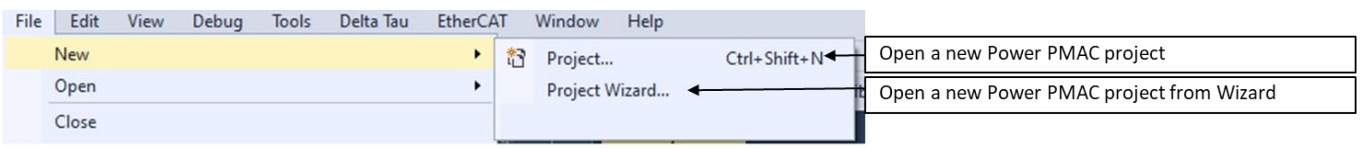
File

This section describes dialog boxes in Visual Studio that pertain to the File menu. The options are described below:



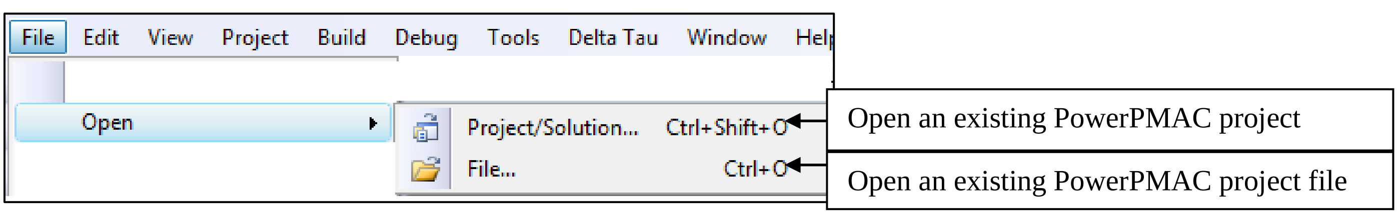
File- New Project/Project wizard

This option allows user to create a new project from template or from project wizard. The option looks like below. It is covered in detail under PROJECT SYSTEM heading.



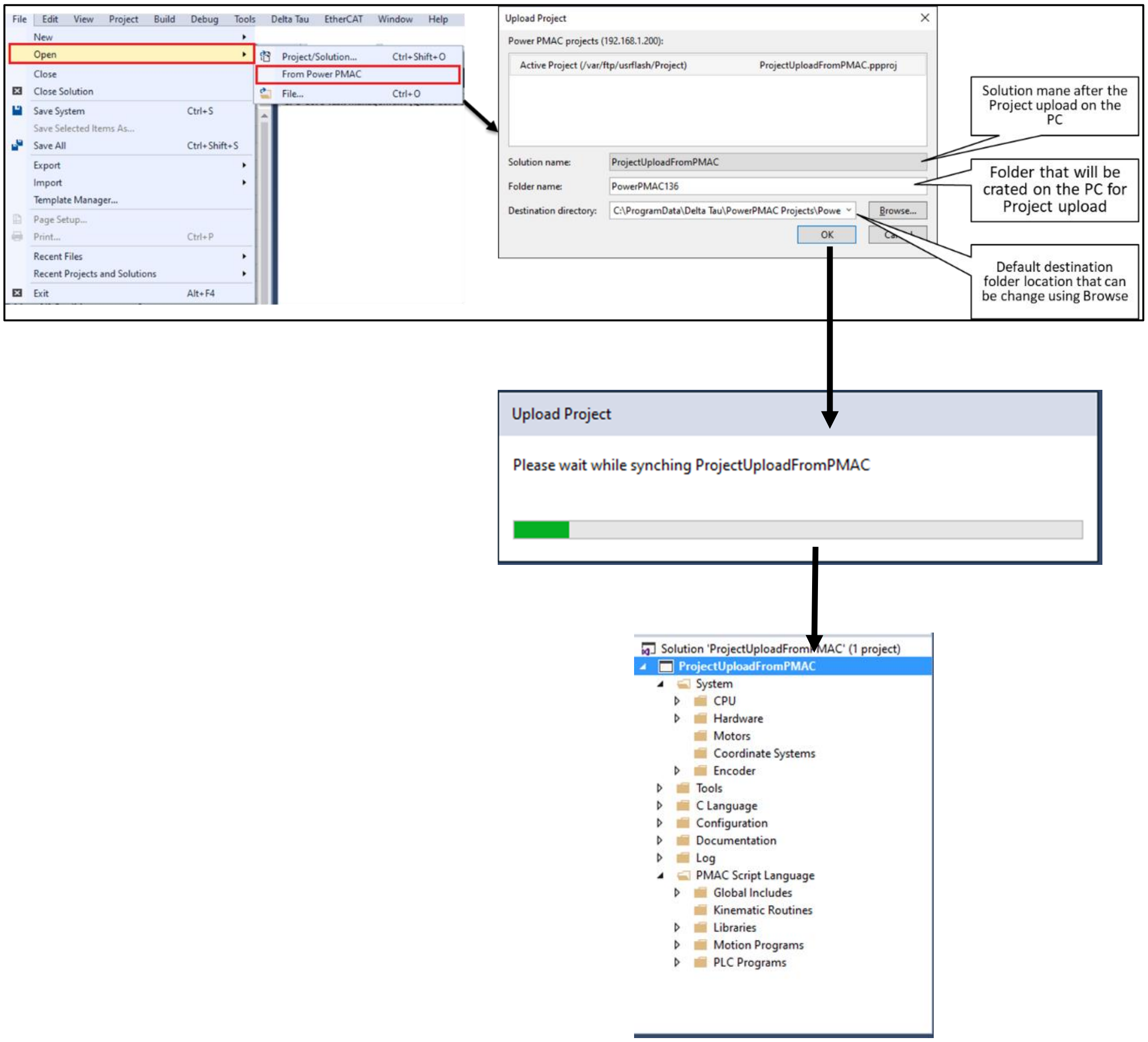
File-Open

This option allows user to open existing project . The option looks like below. It is covered in detail under PROJECT SYSTEM heading.

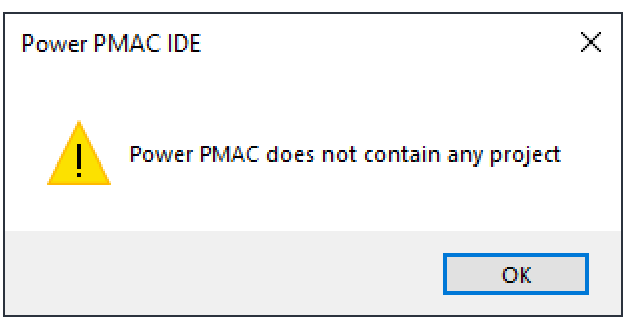


File-Open-From Power PMAC

This is Project upload/Synchronization option.
This option allows user to upload/synchronize the project from Power PMAC to PC. Following workflow shows the upload process ….
As shown the option is available from File-Open-From Power PMAC. Click and it will open the Upload project dialog.
Unlike the previous IDE version (<4.5.2.x) in the current release of the IDE it is not required to have project open to upload the project.
Click OK to upload the project.

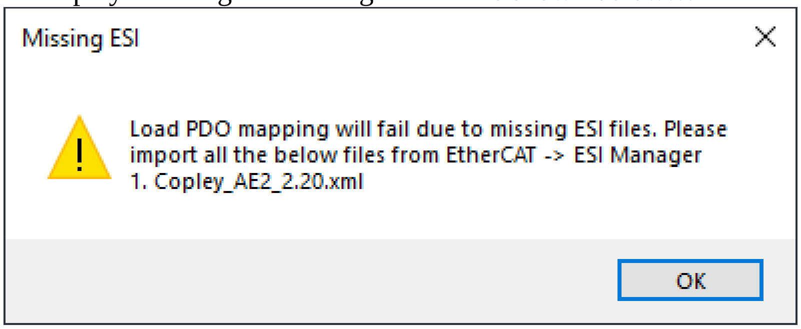


If there is no project on Power PMAC (Typically on $$$*** or brand new Power PMAC board) and user tries to upload the project a clear pop-up message is displayed as shown below…



Project will upload complete project including EtherCAT network setup.
Following are the typcial use cases and how the Project upload handle these cases.
Use case 1: Uploading Power PMAC project with EtherCAT network setup..

On upload if it is determine that requred esi files are not present on the PC a warning pop up message will be displayed listing the missing esi files as shown below…



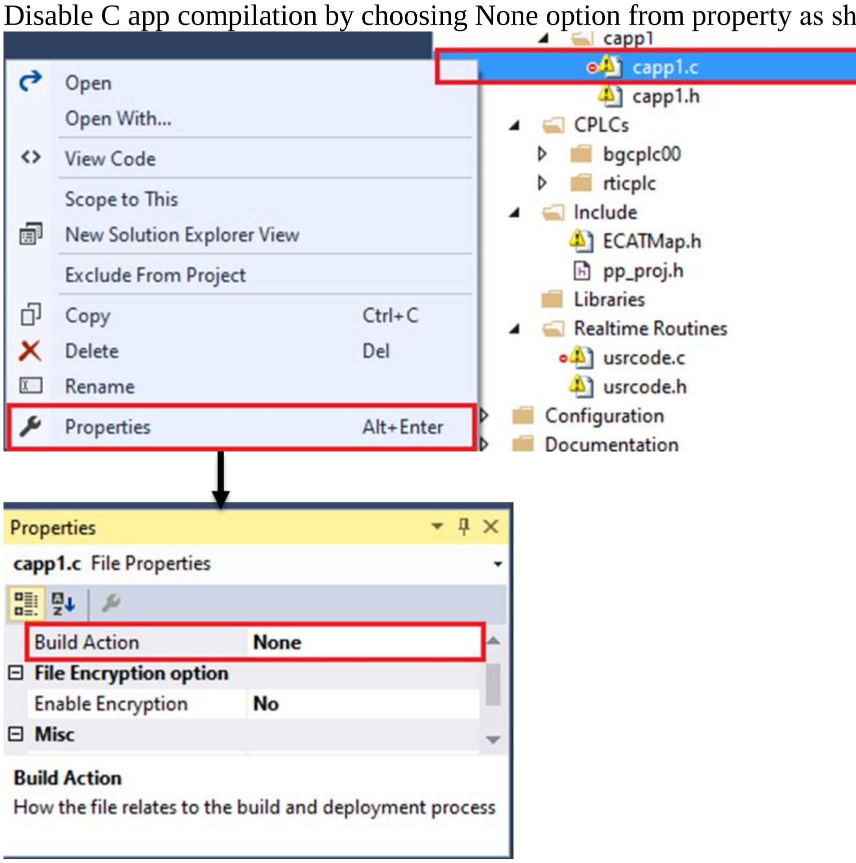
It’s users responsibility to provide the esi files from the EtherCAT device vendor. The esi files are needed for altering the EtherrCAT setup. If it is not required to change the EtherCAT setup then user will be able to download All programs after uploading a sproject from Power PMAC.

Use case 2: If the project is downloaded using the previous Power PMAC IDE (< V 4.5.2.x) then some of the files (mainly EtherCAT and setup files) are not copied to the Power PMAC. This was by design for the previous version of the IDE. In this case a Project Upload will copy all the available files from the Power PMAC and will output the message in the Power PAMC messages window about the files that are not available.

Also by default background c apps source is not part of project download. Thus uploading a project from Power PMAC will not have C source code. Under this circumstances user can only download the project and not build. Build will fail because C source is not available.

Possible choices …

1. Disable C app compilation by choosing None option from property as shown below …



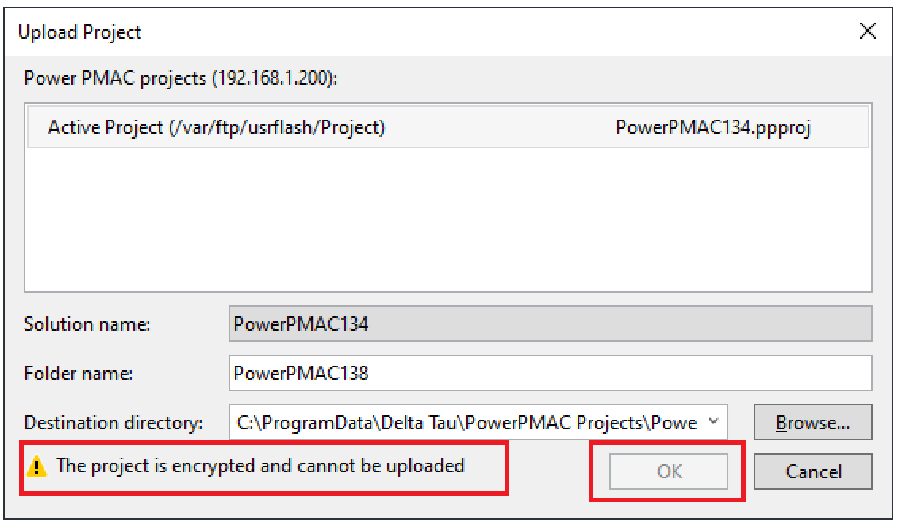
To disable the compilation right click on the file and choose the Properties and then select build action to None. This is shown above workflow.

2. Add the c source file using Add existing file context option from capp1 folder.

Note

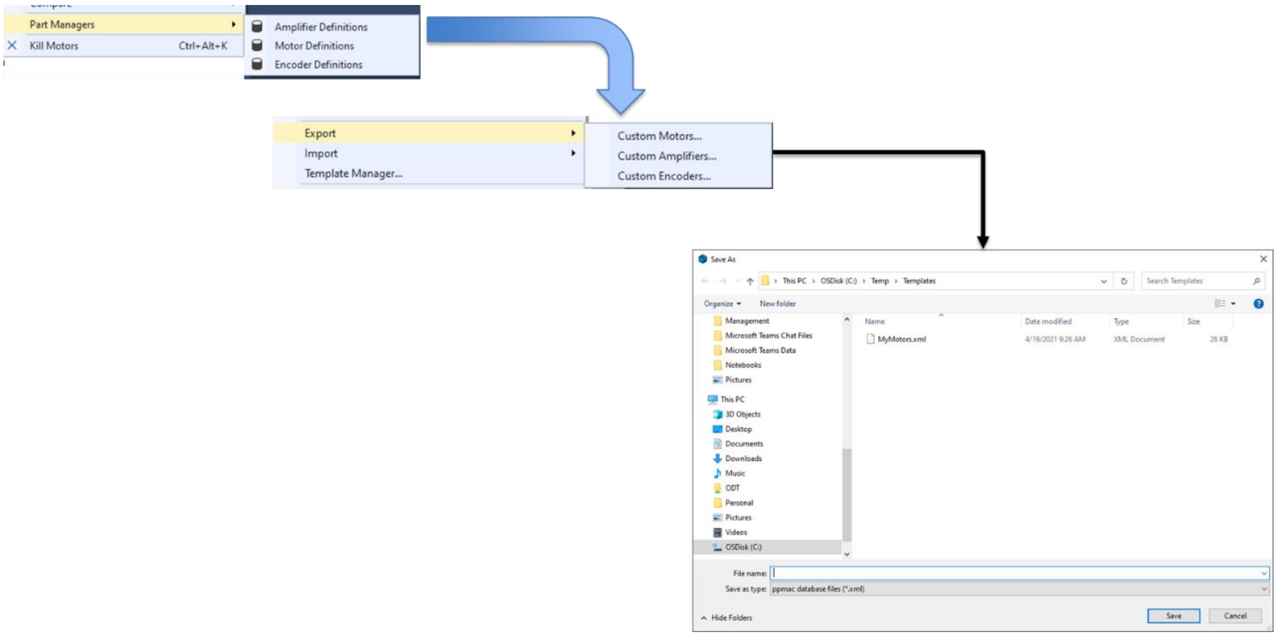
If the uploaded project does not contain the source code for C Libraries, then the uploaded project will show the files in the project tree, but they will not exist project. See the Project Encryption section of this manual for more details.

Use case 3: If the project is encrypted, full encryption or partial encryption and user build and download the encrypted project to Power PMAC then on Project upload is disable with clear indication. This is shown below, OK button is grayed out and warning indicates the reason.



Export

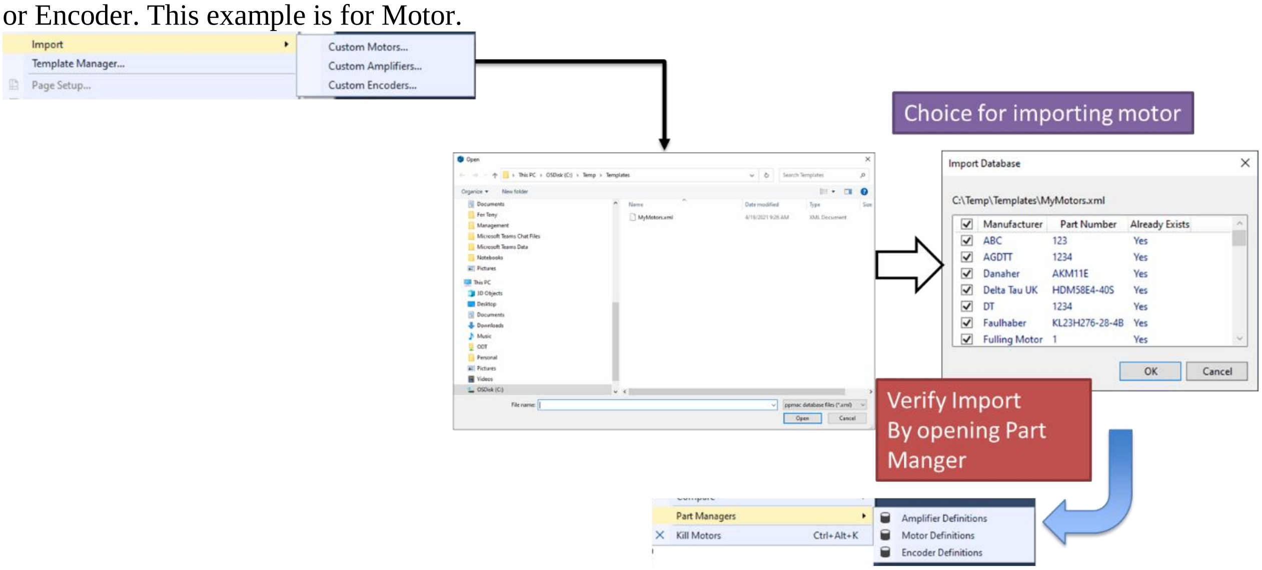
This option is for Exporting Custom Motors, Amplifier or Encoder. User can export any custom data currently present in the Power PMAC IDE system. The purpose of this option is easy share custom Motor, Amplifier or Encoder data with anyone. Typical workflow is below…
On success the xml file will be saved under the selected folder location. The workflow is same for any type of export, Motor, Amplifier or Encoder. This example is for Motor.



Import

This is opposite process of export! This option will import Custom Motor, Amplifier or Encoder in the current Power PMAC IDE system. The purpose is sharing and reusing of databases among or across organization. Typical workflow is below…

On success the xml file will be imported. The workflow is same for any type of export, Motor, Amplifier or Encoder. This example is for Motor.

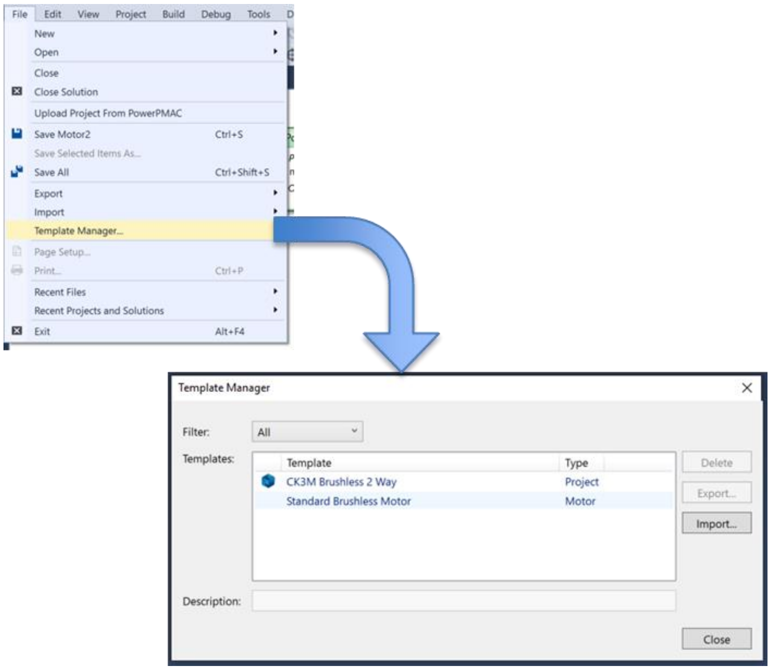


User have a choice for which motor's (Amplifier/Encoder) to be imported. Press OK to import.

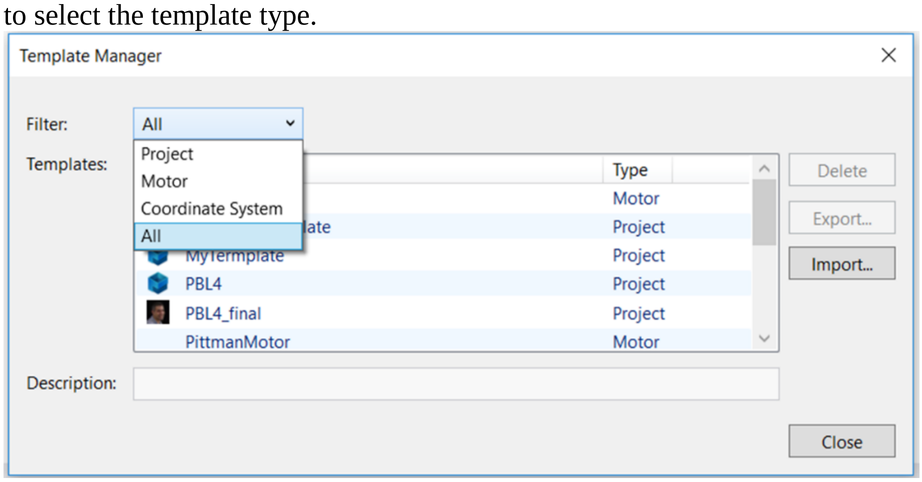
User can verify import by opening Part manage as shown in the picture.

Template Manager

This new dialog is a combined project and item template manager, supporting multiple template types across the IDE, and allowing for future additions and enhancements with all the capabilities of the previous manager. It is available from File menu as shown below:



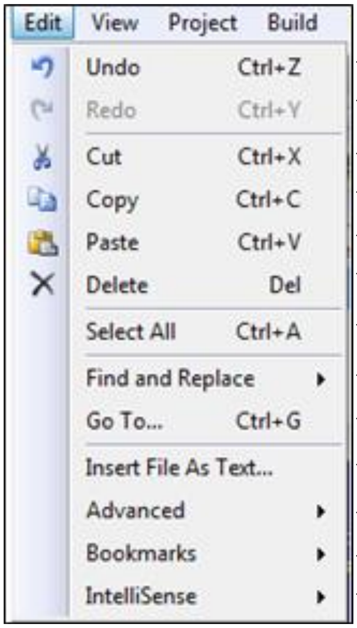
The user can select the template that they need to export or import. The Filter drop-down allows the user to select the template type.



Use Export and Import from the template manager as explained in the earlier section "Export/Import Project and Item template".

Edit

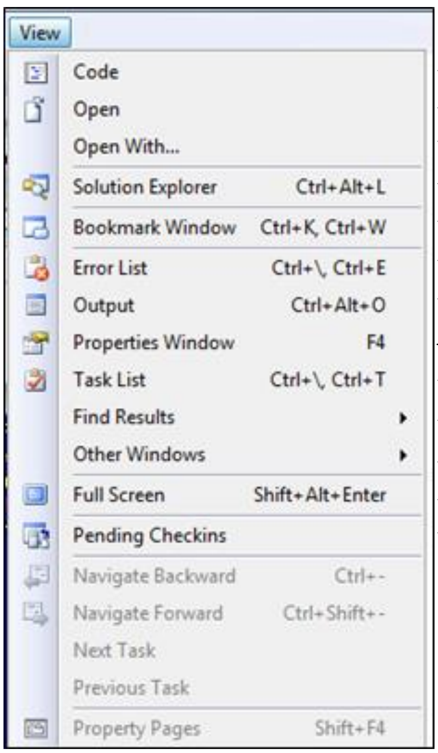
This section describes the functionality of the menu items in the Edit menu. These options are applicable to the file opened in the Editor and the project system. The options are described below:



Undo the last action that was performed in the Editor
Redo the last Undo action in the Editor
Cut the selection in the Editor to the Clipboard
Copy the selection in the Editor to the Clipboard
Paste the contents of the Clipboard to the Editor
Delete the present selection
Select all text in the Editor
Find and/or Replace text in the Editor
Go to a specific line number in the current file
Add a file's contents to the location in the current editing file at the cursor's location
Advanced editing options
Advanced Editor Bookmark options
Advanced IntelliSense options

View

This section describes the functionality of the menu items in the View menu.

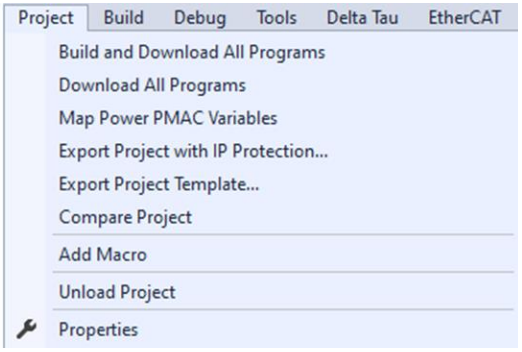


Not implemented
Not implemented
Open the currently opened Editor file in a different editor program
Open the Solution Explorer
Open the Bookmark tab page in the Output Window (cf. Default Layout)
Open the Build Error tab page in the Output Window (cf. Default Layout)
Open the Output Window
Open the Properties Window for the file currently selected in the Solution Explorer
Open the Task List tab page in the Output Window (cf. Default Layout)
Open the Find Result tab page in the Output Window
Open the IDE Command Window (not Terminal Window)
Display the current file from the Editor in Full Screen mode
Not implemented

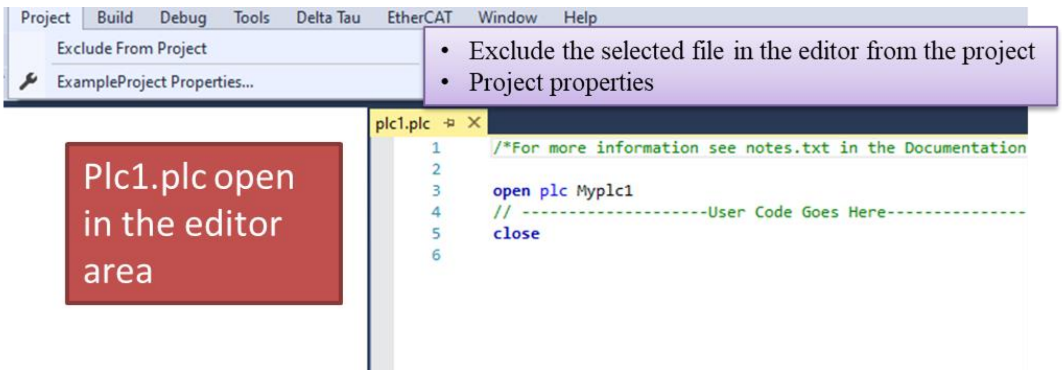
Project

This section describes the functionality of the menu items in the Project menu: This is a dynamic menu and changes with respect to if project is loded or not. If the project is loded and editor area does not have any file or editor area is empty then the Project menu will look like …

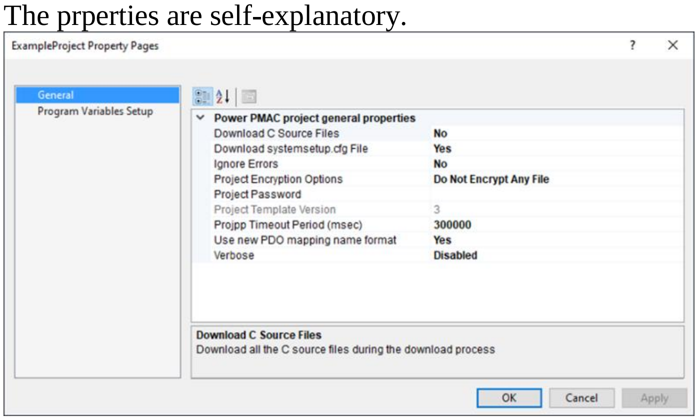
Each item is explained in detail under Project System- Project context menu to avoid duplication.

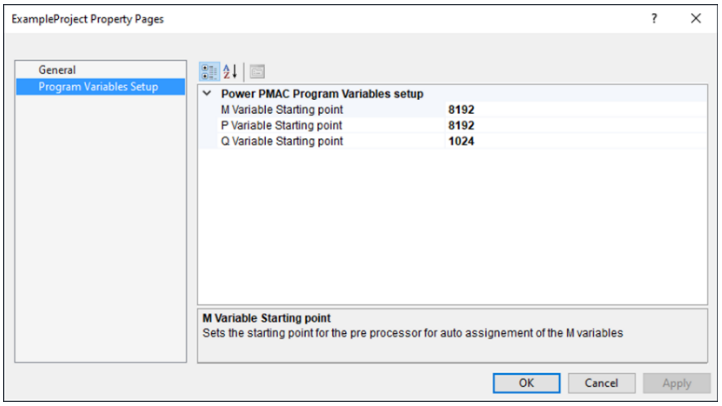


If the project is open and there is any file open the editor area the menu will look line…



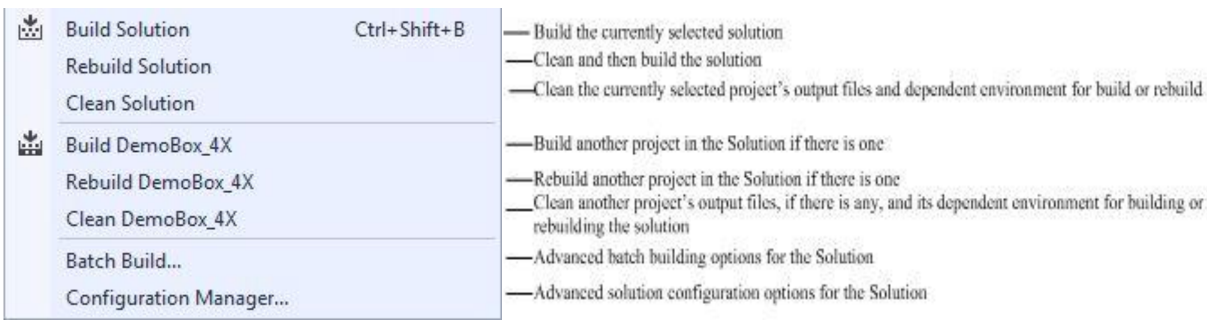
The Project Properties dialog is opened from the Project Properties menu item shown above. Properties are categorised in two parts, General and Program variable setup as shown below. The prperties are self-explanatory.





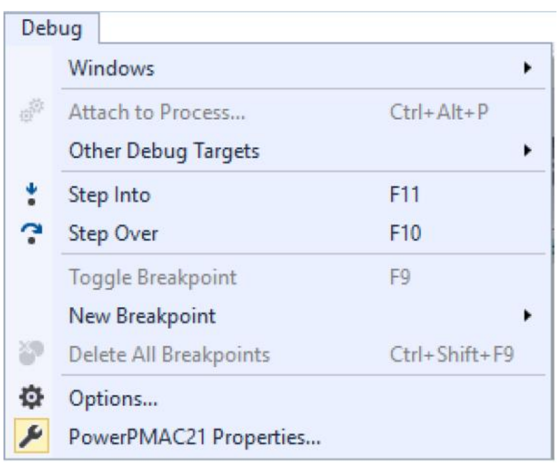
Build

This section describes the functionality of the menu items in build menu:



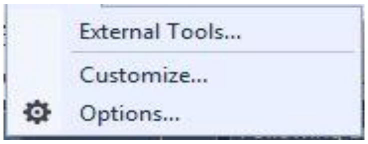
Debug

This section describes the functionality of the menu items in the debug menu.



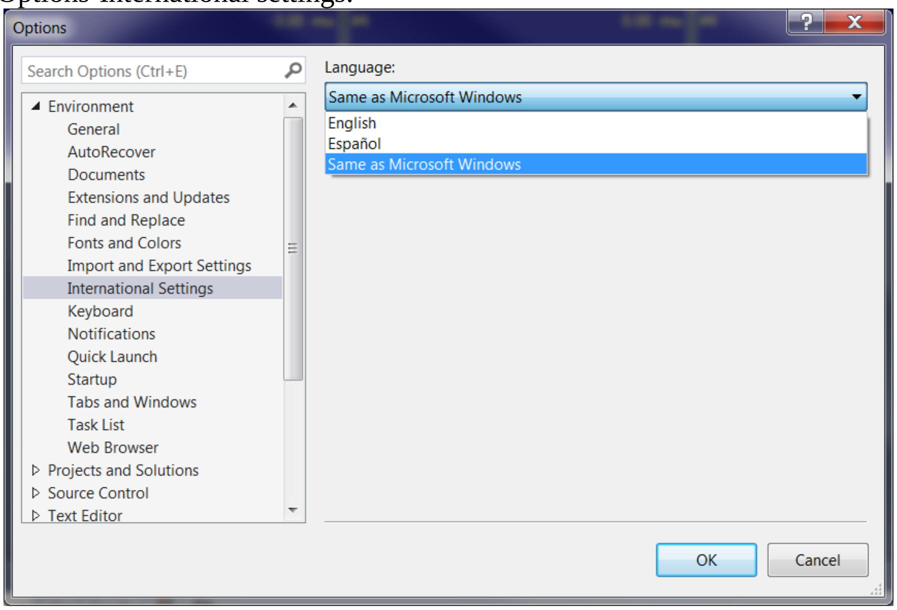
Tools

This section describes the functionality of the menu items in the Tools menu.



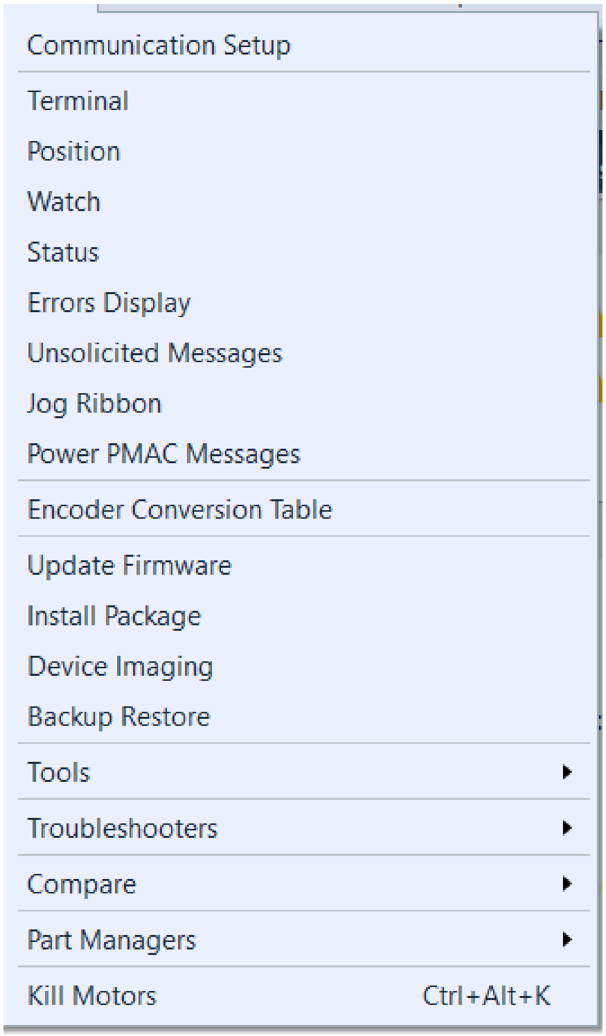
- Add additional external controls
- Add or remove commands on any menu or toolbar
- Manage the environment

Power PMAC IDE supports English, Japanese, Spanish, Korean and Simple Chinese. Language packages are installed at the time of IDE installation. The Language of the IDE can be changed from Tools-Options-International settings.



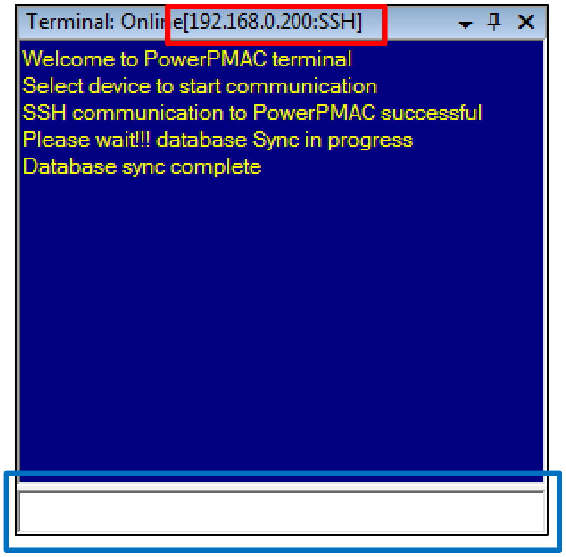
Delta Tau

All the monitoring and configuring windows pertaining to Power PMAC controller are under Delta Tau menu.



Terminal Window

The Terminal Window is a text parser into which the user can enter commands to send to the Power PMAC. The IP address of the device which this window is communicating with is displayed at the top of the window (indicated by the red box in the image below):



Type the command wanted to send into the command entry box (indicated by the blue box in the image below) and press the Enter key on the keyboard to transmit the string to the Power PMAC.

If the command produces a response from the Power PMAC, the Terminal Window will show the response.

Text can be copied from the window by highlighting it with the mouse and pressing CTRL+C on the keyboard. To select all of the text in the window click on the window and press CTRL+A and then CTRL+C to copy it.

Text can be pasted into the text parser by clicking in the command entry box and pressing CTRL+V on the keyboard.

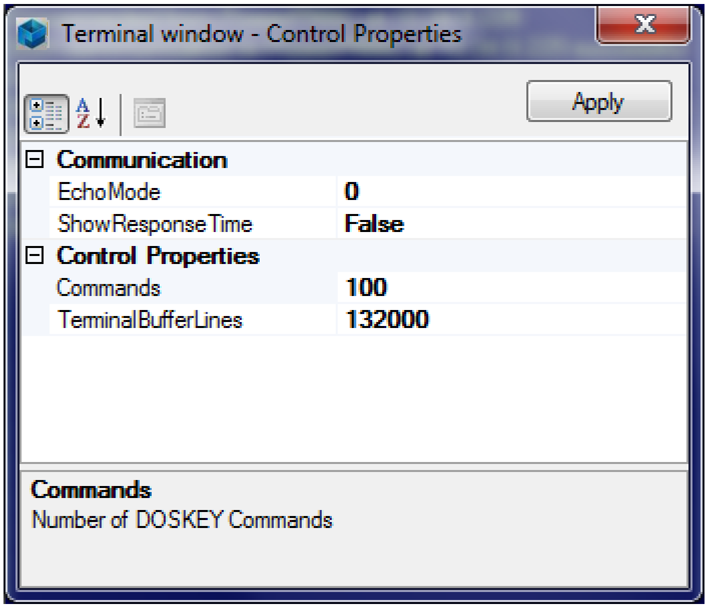
Commands can be dragged and dropped from the Editor Window or the Watch Window into the command entry box of the Terminal Window.

If more detail is needed about a command type it into the command entry box and press the F1 key on the keyboard.

Motors can be killed by clicking on the command entry box and pressing CTRL+ALT+K on the keyboard.

To save the whole contents of the Terminal Window, right-click the window and then click Properties→Control→Save Buffer to File. Contents can also be copied to the operating system's clipboard by clicking Properties→Control→Copy Buffer to Clipboard. To clear the contents of the Terminal Window, click Properties→Control→Clear Buffer.

There are more properties that can be modified by right-clicking the window and then clicking Properties→Control→General which will open this screen:



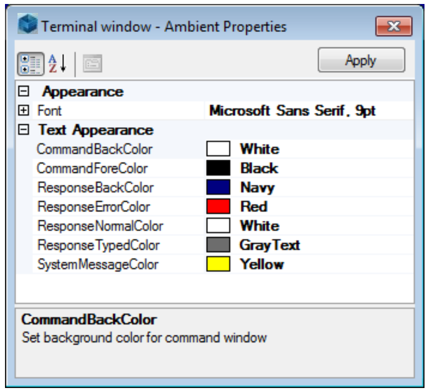
In the "Communication" box, there are two fields:

- "EchoMode" indicates how and if information is echoed back to the Terminal Window after issuing a command; see the command labeled **echo*{constant}*** in the Power PMAC Software Reference Manual for more details.
- "ShowResponseTime" [True/False], when set to True, will show how long [msec] Power PMAC took to reply after receiving a command from the host. It also lists how many characters will be received. When this option is set to False the Terminal Window uses asynchronous communications when talking to Power PMAC; that is the window sends commands to Power PMAC via one thread and receives the responses from Power PMAC on another thread. When ShowResponseTime is True the Terminal Window switches to synchronous communication sending commands to Power PMAC on one thread and then waiting, in the same thread, until Power PMAC finishes responding before the Terminal Window will show the response time.

In the "Control Properties" box there are three fields:

- The "Commands" field indicates the number of commands which were previously typed into the Terminal Window. Commands can be scrolled through using the up and down arrow keys on the keyboard.
- "LogAllMessages" [True/False], when set to True, will cause any error messages that the Terminal Window generates to report to the Delta Tau Log window (see IDE Layout section for the location of this window). These errors are from the IDE itself and not from Power PMAC.
- "TerminalBufferLines" specifies how many lines the Terminal Window will store before cycling them out; that is, the oldest commands are cleared out and the new commands are added in as they are entered.

To change the color scheme and fonts of the window right-click the window and then click on Properties→Ambient. This window will pop up:

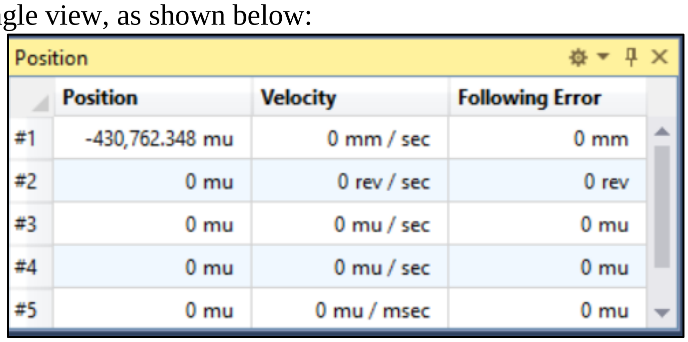


In this window the text's font and the colors of various types of commands, and responses, can be changed as desired.

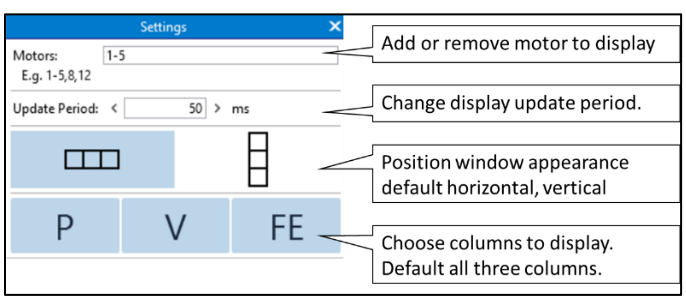
One or more commands may also be input by selecting them in a text file, whether from the Editor Window or an external program (e.g. Notepad or Microsoft Word™), and drag-dropping them into the command text box of the Terminal Window.

Position Window

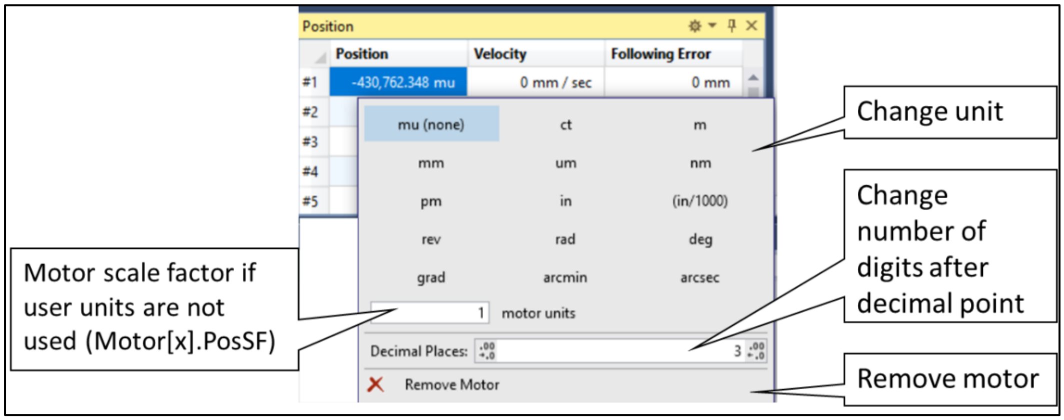
The Position Window in IDE version 4.2 and above combines the position, velocity and following error for the motors into a single view, as shown below:



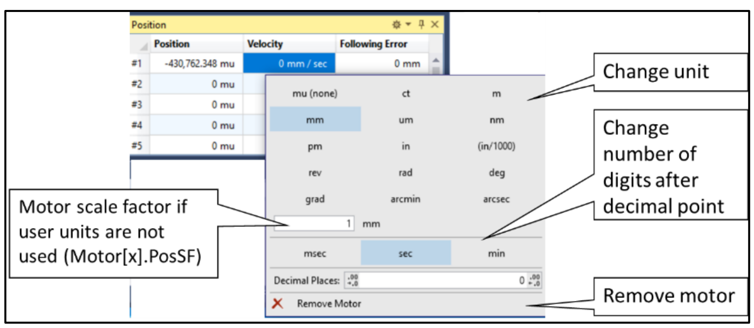
Click setting icon to change the parameters. The following image shows the possible settings.



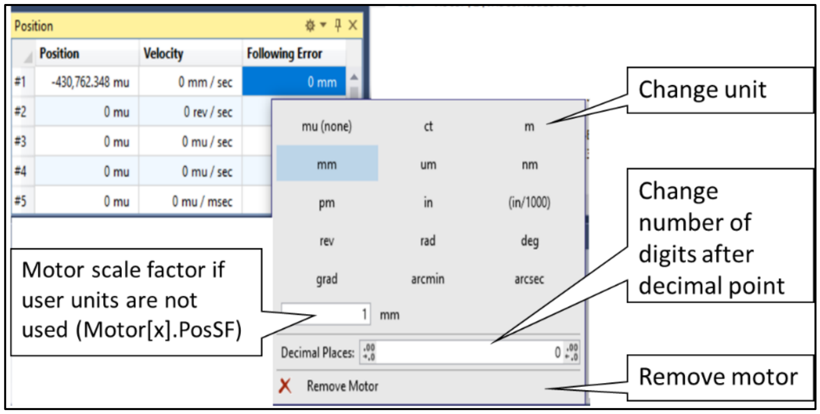
To change motor position unit, select the motor position cell and right click. See the image below.



To change motor velocity unit, select the motor velocity cell and right click. See the image below.



To change motor following error unit, select the motor following error cell and right click. See the image below.



Note

If the User Unit block from Motor topology is used to set the units then the user will not be able to change position, velocity or following error units and scale factor for that motor and it will be grayed out.

Watch Window

Commands and variables can be added into the Watch Window in order to monitor their value at the specified rate. By default the Watch Window consists of two columns as shown below:

Caution

If a valid command is input the IDE transmits the command typed into the "Command" column repeatedly. Only safe commands should be sent. To add commands to the "Unsafe Commands List" click and select Edit Unsafe Commands. Some examples of typical unsafe commands are **kill**, **$$$**, **save**, **out**, etc.

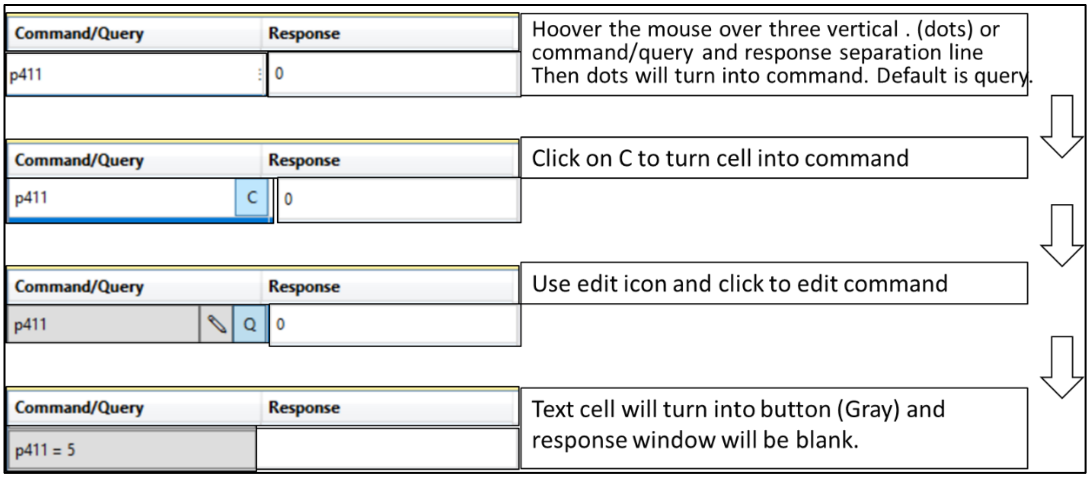
Click in the text entry box underneath "Command/Query" and type the command or variable name required to monitor and press Enter. The response, if there is one, will be shown in box underneath "Response." If the response returns an error then the command will not be sent in the next update cycle.

The Default entry in the watch table is Query.

Commands can also be sent to the Power PMAC from the Watch Window.

To change a default Query into a Command, follow the sequence shown in workflow below.

Here p411 is a default Query. Using this workflow this will be converted to command of p411 = 5.



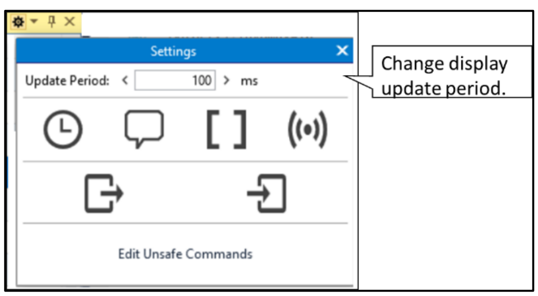
To convert back from a Command to a Query follow the same workflow in reverse.

Now that the Command is a Query remove the '= 5' from the entry.

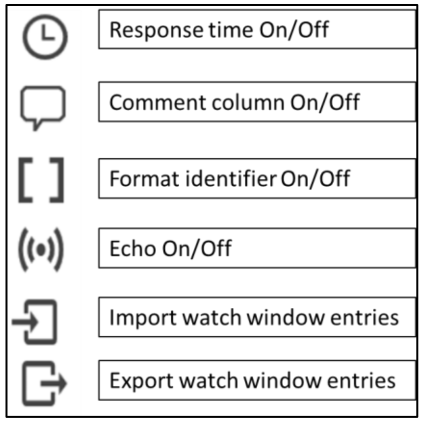
Commands can be drag and dropped from the Editor Window, the Terminal Window, or a text file from an external program (e.g. Notepad or Microsoft Word™) into the command entry box of the Watch Window.

Multiple commands may be drag and dropped into a Watch Window command row box in order to create many new entries at once.

To change watch window settings click icon to open settings window.

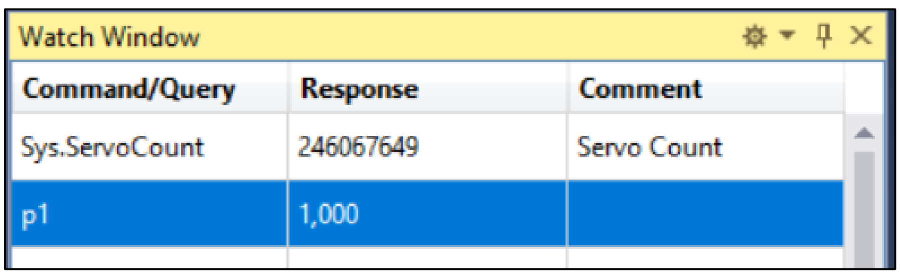


The symbol displayed represent different settings that are possible, as shown below.

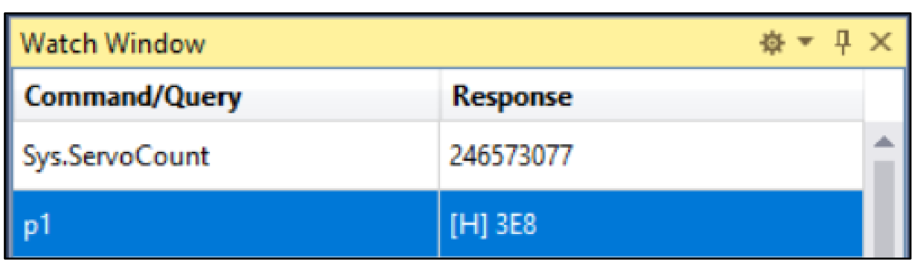


- ResponseTime On/Off: When On, this will show how long [msec] the Power PMAC took to reply after receiving a command from the host. It will also list how many characters will be received.

- Comment Column On/Off: When On, this will show an additional column in the Watch Window in which personal notes can be added to annotate that row as shown in the example screenshot below:



- Format identifier On/Off : This indicates the type of formatting on the received response, as shown below:

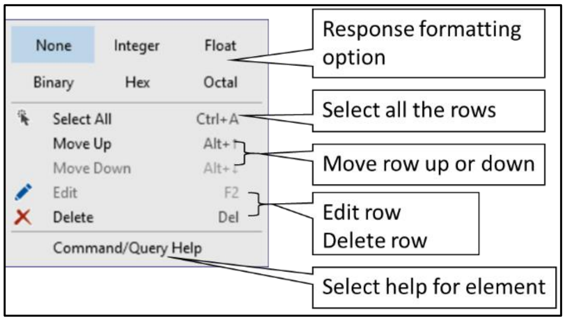


- Echo On/Off: This indicates how, and if, information is echoed back to the Terminal Window after issuing a command; see **echo{constant}** in the Power PMAC Software Reference Manual for more details.

- Import Watch Window entries: This enables the User to import the Watch table previously saved and loads it into the Watch window.

- Export Watch Window entries: This enables the User to export the Watch table current entries into a file.

Right clicking on any row will display the context menu shown below.

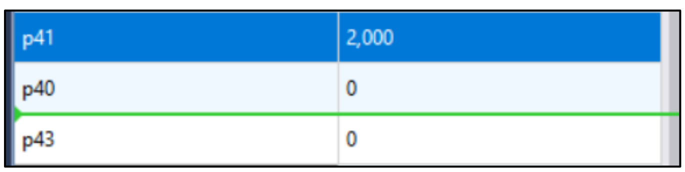


Move up/down:

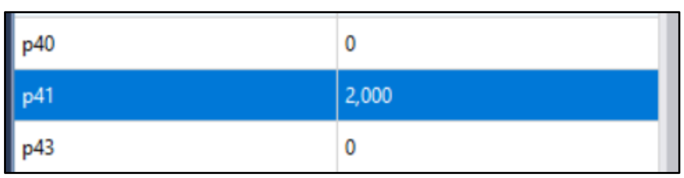
From this context menu the User can move the selected row up or down by clicking on the ‘Move Up’ or ‘Move Down’ entry in the context menu.

The User can also move a row up or down using the mouse. To do this the user needs to select the row by clicking on it, move the row by holding down the left mouse button and dragging the row to the new position. The User can also select multiple rows them in the same way.

When dragged, a Green line will show where the row/rows will be inserted. In the example below the row with P41 is selected and will be moved in between p40 and p43, as show by the green line indicator.



When the User removes their finger from the mouse button P41 is placed in between p40 and p43.



Formatting Option:

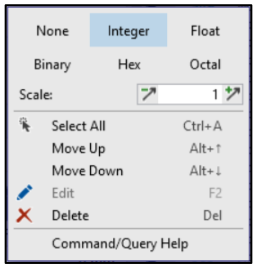
There are Five formatting options available on the context menu.

Selecting the required formatting will dynamically change the necessary formatting parameters.

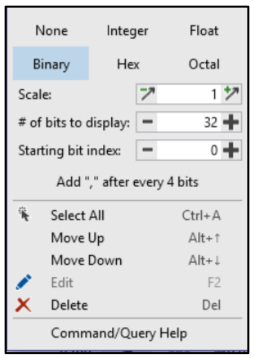
For any of the formatting options, select a row and then right click on the response section to open the context menu.

Choose the required format option from the following list:

Integer: This format will force the number to be a whole number. Enter a scale factor for the data in the "Scale" box if desired

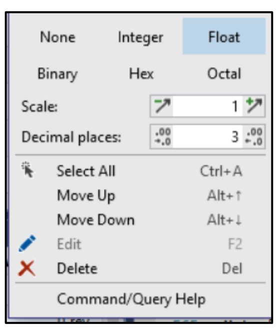


Binary: This format will show the number as a sequence of bits indicate by 0s and 1s. Enter a scale factor in the "Scale Factor" box and, if required, a numerical mask in the "Mask" box:

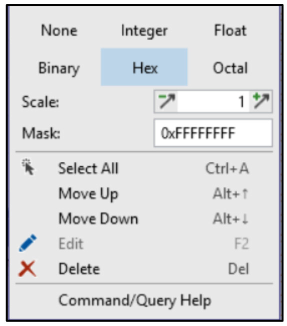


The number entered in the "Mask" box needs to be in hexadecimal format preceded by the symbol "0x" (without the quotation marks). The IDE will then bitwise "AND" this mask with the response before displaying it in the Watch Window.

Float: This will force the Watch Window to display decimal information for this number. Enter a scale factor in the "Scale" box and, if required, specify the number of decimal points in the "Decimal Places" box:

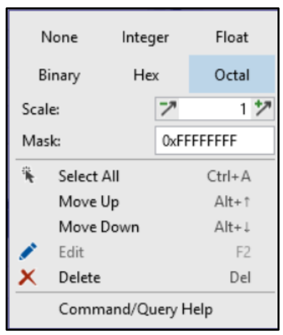


Hex: This format will force the number into a hexadecimal format. Enter a scale factor into the "Scale" box and, if required, also a mask in the "Mask" box:



The number entered in the “Mask” box needs to be in hexadecimal format preceded by the symbol “0x” (without the quotation marks). The IDE will then bitwise “AND” this mask with the response before displaying it in the Watch Window.

Octal: This format will force the number into a base-8 numerical format. Enter a scale factor in the “Scale” box and, if required, a numerical mask in the “Mask” box:



The number entered in the “Mask” box needs to be in hexadecimal format preceded by the symbol “0x” (without the quotation marks). The IDE will then bitwise “AND” this mask with the response before displaying it in the Watch Window.

Safety Notes

As previously stated, if a valid command is input the IDE transmits the command typed into the “Command” column repeatedly. Only safe commands should be sent. To add commands to the “Unsafe Commands List” click and select Edit Unsafe Commands. Some examples of typical unsafe commands are **kill**, **$$$**, **save**, **out**, etc.

If an invalid command is transmitted, the Watch Window will only transmit the command once and then the invalid response will be highlighted in red and will remain in the response area of the Watch Window. This will not be transmitted again.

Note that there is a structure called **Sys.NoShortCmds** which will force the user to input the full name of online commands.

If **Sys.NoShortCmds=0** then commands such as **#1k** can be used to kill motor 1.

If **Sys.NoShortCmds=1** the the full name of the command must be used like **#1kill** to kill motor 1. This feature can be useful; for example, if an invalid variable name is typed containing a **k** (as in the **kill** command) or **r** (as in the **run** command) or **j** (as in the **jog** command) or **a** (as in the **abort** command)

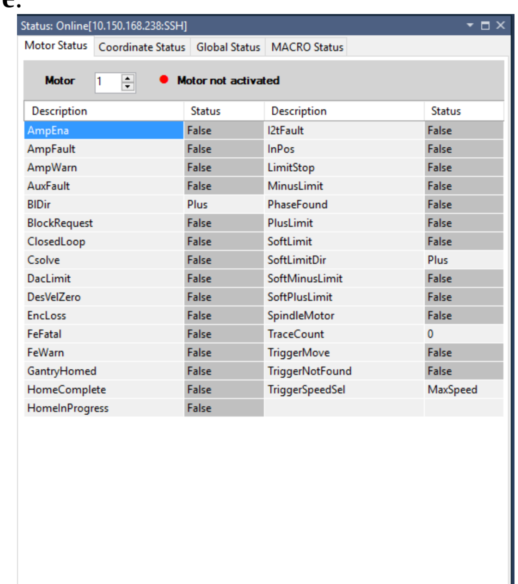
then the Watch Window will transmit that invalid variable name and the Power PMAC will parse it and try to execute whatever command it can recognize within the invalid variable name.
For example, if a invalid variable named “**MyVar**” is not declared in the entire project, or was formerly declared but is now deleted, is added to the Watch Window or transmitted in a string from the HMI program communicating with the Power PMAC, the Power PMAC will interpret this as first an **abort** command because of the **a** in **MyVar** and then as **run** command because of the **r** in **MyVar**.

Status

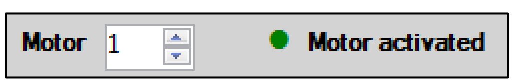
The Status Window actually contains four tabs which each give the status of a different set of information:

Motor Status

The first tab is the Motor Status tab which gives status information about motors. Each status field name listed in the Description column comes from a motor status structure. The full name of the motor status structure starts with “**Motor[x].**”, where **x** is the motor number, and ends in the name listed in the Description column of the Motor Status Window. For example, in the Description column, the first entry is TriggerMove, which corresponds to the **Motor[x].TriggerMove** structure. For example, for motor 1 this is **Motor[1].TriggerMove**.



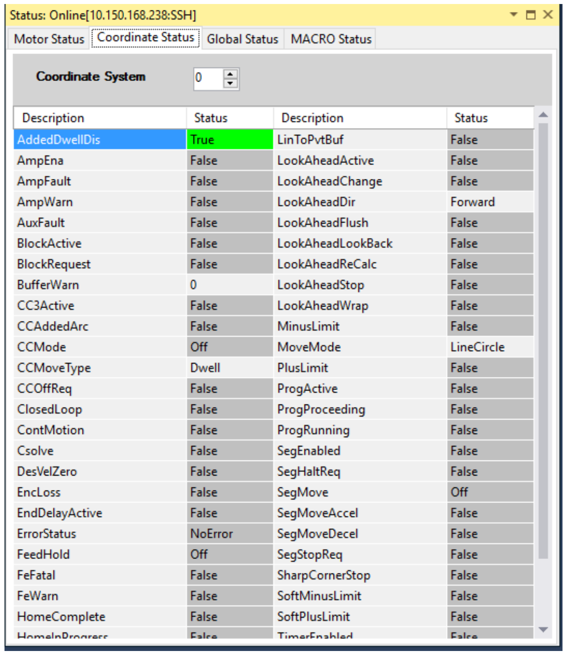
The user can select which motor to monitor the status by typing the motor number into the box next to the “Motor” label as shown below:



The dot to the right of this box shows whether the motor is activated: when green, the motor is activated; when red, the motor is not activated.

Coordinate Status

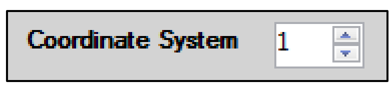
The second tab is the Coordinate Status tab, which gives status information about Coordinate Systems.



Description	Status	Description	Status
AddedDwellDis	True	LinToPvtBuf	False
AmpEna	False	LookAheadActive	False
AmpFault	False	LookAheadChange	False
AmpWarn	False	LookAheadDir	Forward
AuxFault	False	LookAheadFlush	False
BlockActive	False	LookAheadLookBack	False
BlockRequest	False	LookAheadReCalc	False
BufferWarn	0	LookAheadStop	False
CC3Active	False	LookAheadWrap	False
CCAddedArc	False	MinusLimit	False
CCMode	Off	MoveMode	LineCircle
CCMoveType	Dwell	PlusLimit	False
CCOffReq	False	ProgActive	False
ClosedLoop	False	ProgProceeding	False
ContMotion	False	ProgRunning	False
Csolve	False	SegEnabled	False
DesVelZero	False	SegHaltReq	False
EncLoss	False	SegMove	Off
EndDelayActive	False	SegMoveAccel	False
ErrorStatus	NoError	SegMoveDecel	False
FeedHold	Off	SegStopReq	False
FeFatal	False	SharpCornerStop	False
FeWarn	False	SoftMinusLimit	False
HomeComplete	False	SoftPlusLimit	False

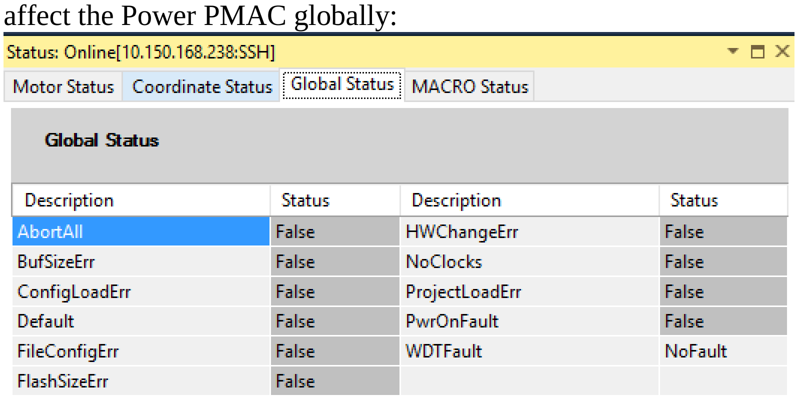
Each status field name listed in the Description column comes from a Coordinate System status structure. The full name of the motor status structure starts with "**Coord[x].**", where **x** is the Coordinate System number, and ends in the name listed in the Description column of the Motor Status Window. For example, in the Description column, the first entry is TriggerMove, which corresponds to the **Coord[x].TriggerMove** structure. For example, for Coordinate System 1, this is **Coord[1].TriggerMove**.

The user can select which motor to monitor the status by typing the motor number into the box next to the "Coordinate System" label as shown below:



Global Status

The third tab is the Global Status tab, which gives status information about configuration settings which affect the Power PMAC globally:

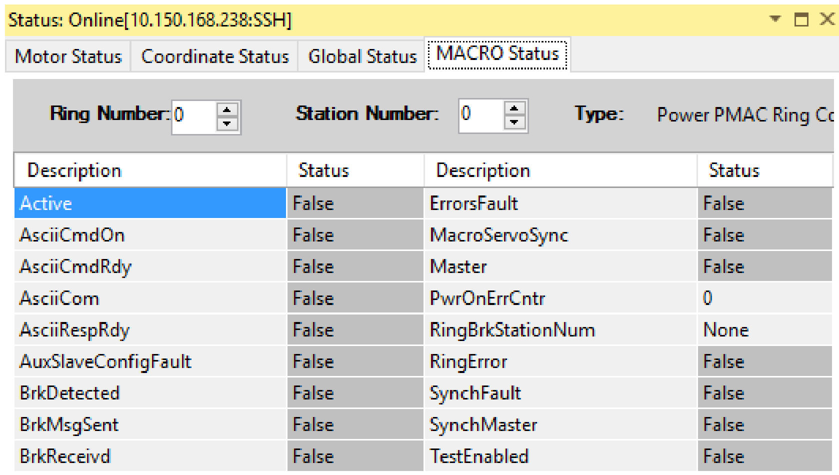


Description	Status	Description	Status
AbortAll	False	HWChangeErr	False
BufSizeErr	False	NoClocks	False
ConfigLoadErr	False	ProjectLoadErr	False
Default	False	PwrOnFault	False
FileConfigErr	False	WDTFault	NoFault
FlashSizeErr	False		

Each status field name listed in the Description column comes from a System status structure. The full name of the motor status structure starts with "**Sys.**" and ends with the name in the Description column.

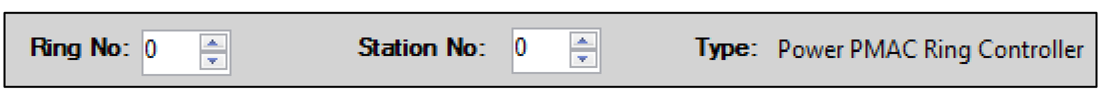
For example, the first entry in the Description column is “NoClocks,” which corresponds to the **Sys.NoClocks** structure.

MACRO Status
The fourth tab is the MACRO Status tab, which gives information about MACRO communication if MACRO is being used with this system.



Each status field name listed in the Description column comes from a MACRO status structure. All of the entries in the Description columns except for PwrOnErrCntr and RingBrkStationNum come from the **Macro.Status[x]** structure tree, where **x** is the ring number, which ranges from 0 to 3. For example, the first entry is Active, which for ring 0 corresponds to the structure **Macro.Status[0].Active**. PwrOnErrCntr and RingBrkStationNum correspond to **Macro.RingTest[x].PwrOnErrCtr** and **Macro.RingText[x].RingBrkStationNum**, respectively, where **x** is the ring number, which ranges from 0 to 3.

The user can select the ring number by typing the number into the box labeled “Ring No” as shown below:

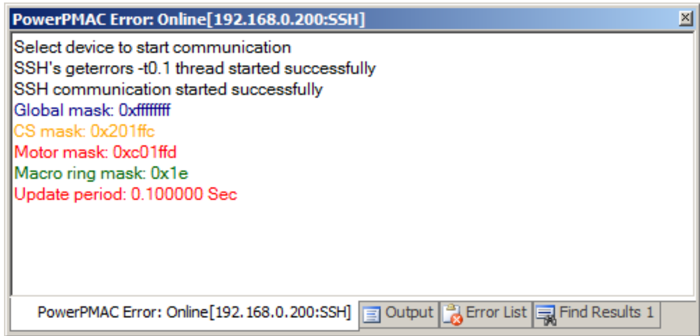


The user can select the station number by typing the station number into the box labeled “Station No” as shown above.

The “Type” label indicates the MACRO Station type of the device with which the Status Window is currently communicating. Typically, this will be a Power PMAC Ring Controller, but it can also be a Power PMAC Master and not necessarily a Ring Controller, depending on how the controller is configured.

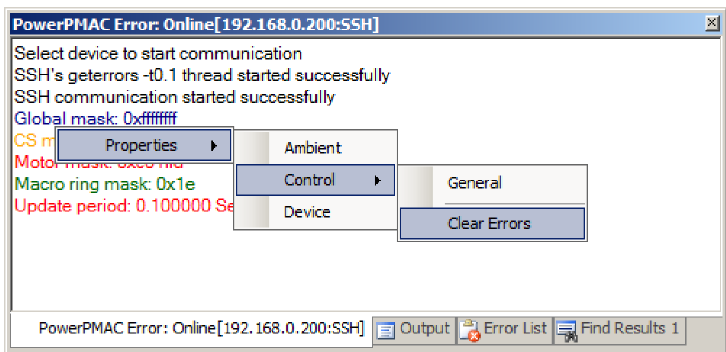
Error Display

The Error Display window displays all errors that Power PMAC reports and appears as follows:

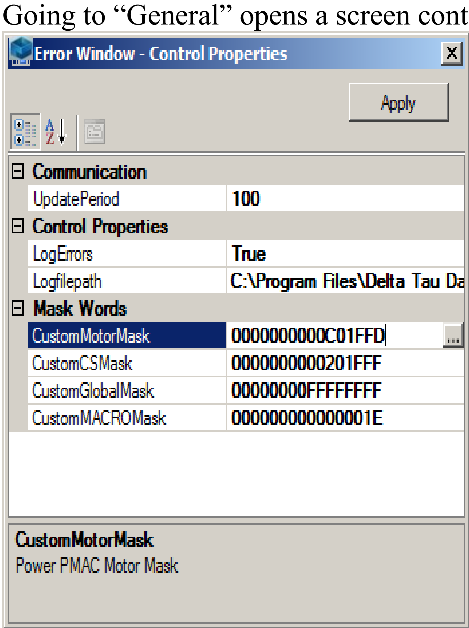


This window starts the background process "geterrors" in Power PMAC. This window reports not only errors, but also certain status updates which Power PMAC reports.

Right-clicking the window and going to Properties→Control→Clear Errors will permit the user to clear all of the information presently shown in the Error Display window:



Going to "General" opens a screen containing several properties of the Error Display window:



UpdatePeriod is the refresh period of this window in milliseconds

LogErrors, when True, starts logging errors to a file whose path is set in Logfilepath below

Logfilepath indicates where to store the Error Display's log file

CustomMotorMask indicates which motors whose errors to check. Click ... to configure.

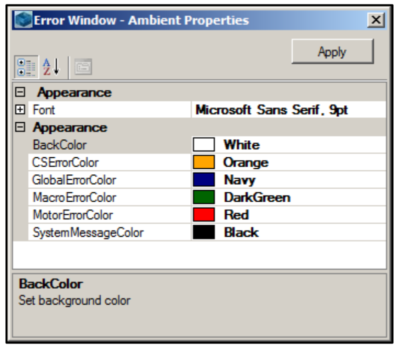
CustomCSMask indicates which coordinate systems whose errors to check. Click ... to configure.

CustomGlobalMask indicates which global errors to check. Click ... to configure.

CustomMACROMask indicates which MACRO errors to check. Click ... to configure.

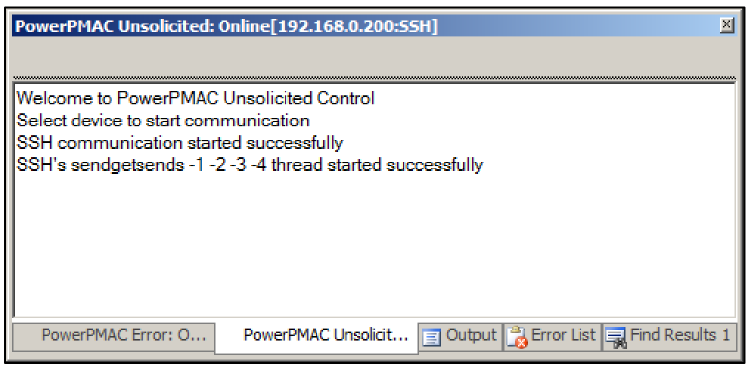
For the four masks, just click on the row containing the mask and the ... button will appear

The user can change the color scheme and fonts of the window by right-clicking it and then clicking on Properties→Ambient, which opens this screen:



Unsolicited Messages

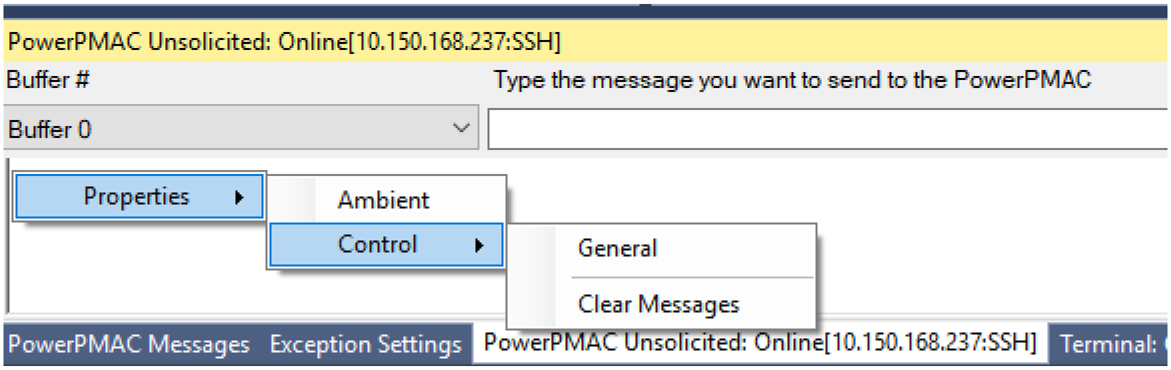
The Unsolicited Messages window displays messages sent to the host computer from Power PMAC over the eight Unsolicited Response ports (Ports 0 – 7):



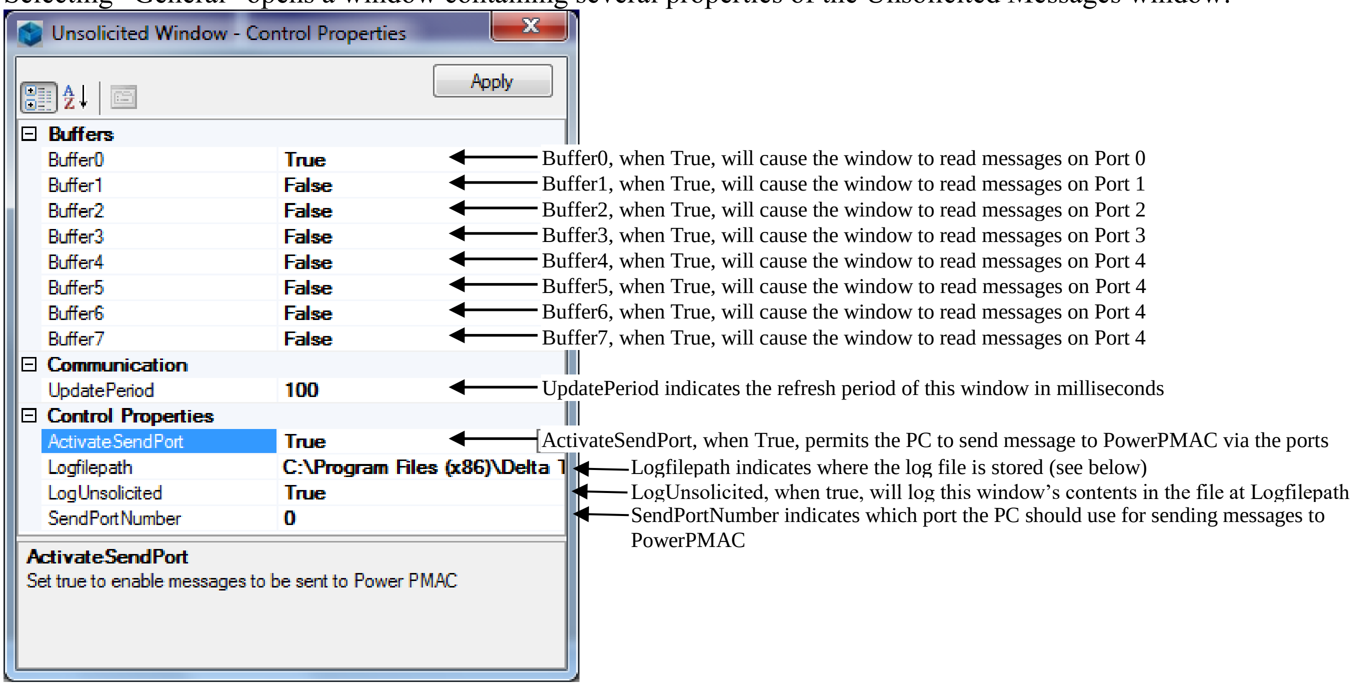
These messages can be sent from a C program using the **Send()** function or from a Script program using the **SEND** command. The host PC can also send messages to Power PMAC through these ports. Upon opening this window, the “sendgetsends” process starts on Power PMAC, which receives all of the messages. In the IDE, Port 0 is enabled at startup; Ports 1 – 7 are disabled. After sending a command from the host to Power PMAC, the status of the port must be checked. Possible status codes include the following:

- 0 means “Command sent OK”
- 1 means “Illegal Command Format”
- 2 means “Port Busy”
- 3 means “Port Full”

The user can clear the messages by right-clicking in the window and selecting Properties→Control→Clear Messages:

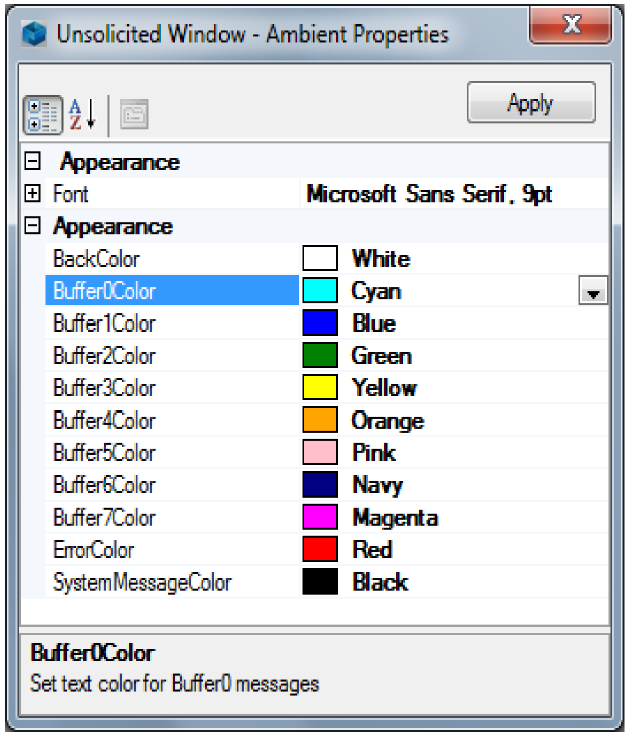


Selecting “General” opens a window containing several properties of the Unsolicited Messages window:



Each bit of the “mode word” **Sys.SendFileMode** can be set to 1 to enable or to 0 to disable sending and receiving ASCII strings on each port. Each bit in this 8-bit word represents one port. For example, to enable all ports set **Sys.SendFileMode=$FF**. To enable just Port 0 set **Sys.SendFileMode=$1**.

Selecting “Ambient” loads the following window where font and color can be chosen for each Port’s text:

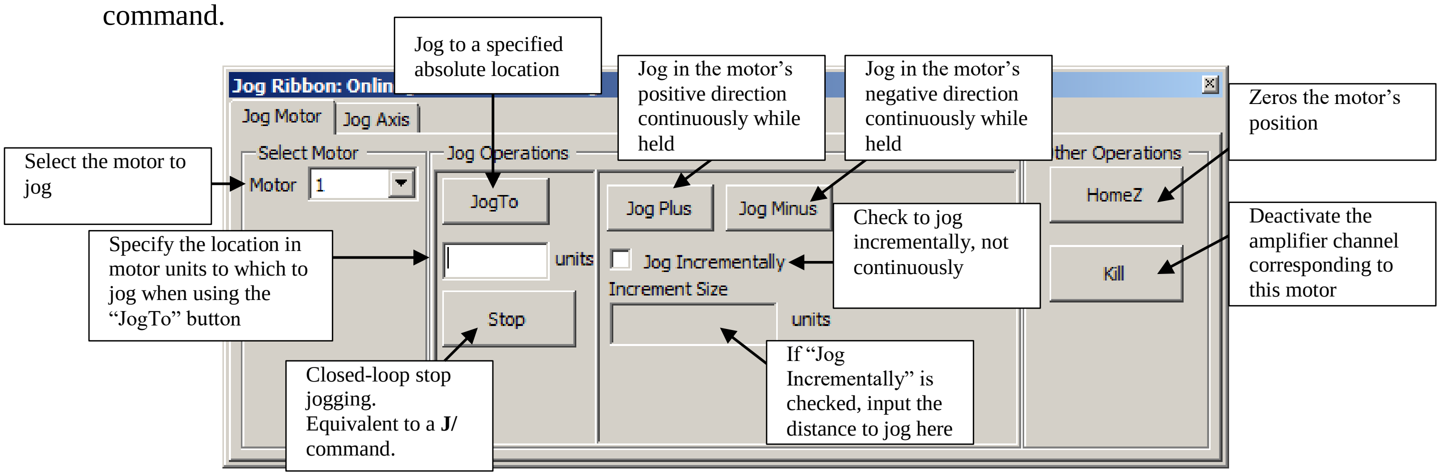


Jog Ribbon

Please make sure the selected Motor to Jog matches with Axis (If defined).
Jog Plus or Jog Minus moves motors.

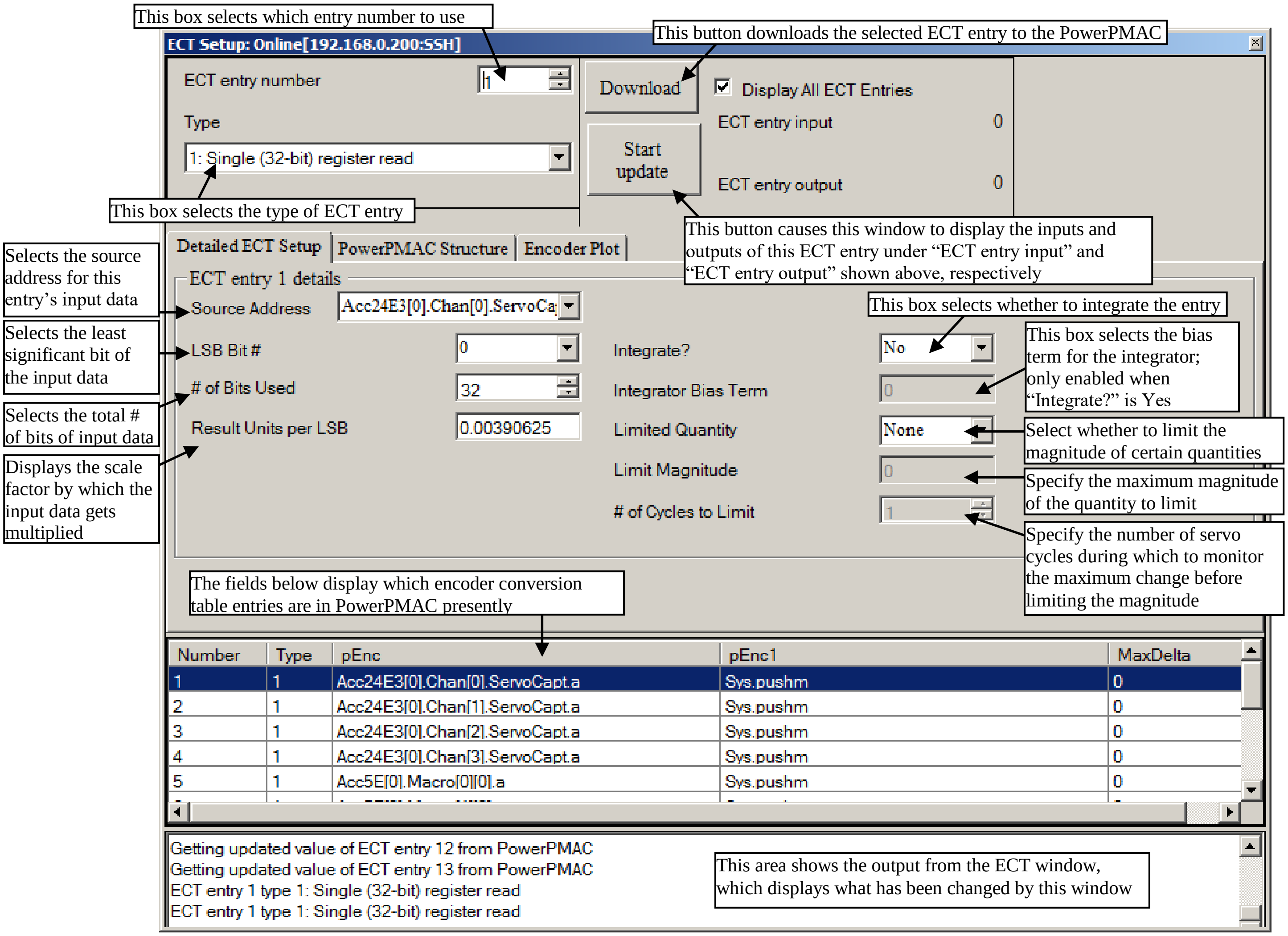
Note

The jog ribbon permits the user to jog motors or axes individually. A jog move is simply a point-to-point, constant velocity move. If the user wants to jog one motor in motor units, click on the "Jog Motor" command.

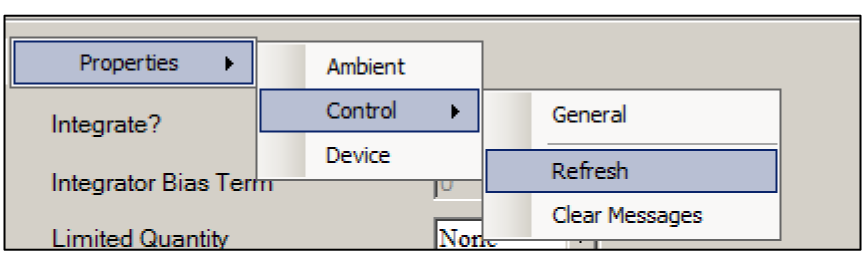


Encoder Conversion Table

The Encoder Conversion Table (ECT) window is for setup purposes and should only be used by advanced users. Its purpose is to configure the fields within the EncTable[x] structure. The main tab of the ECT window appears as follows:

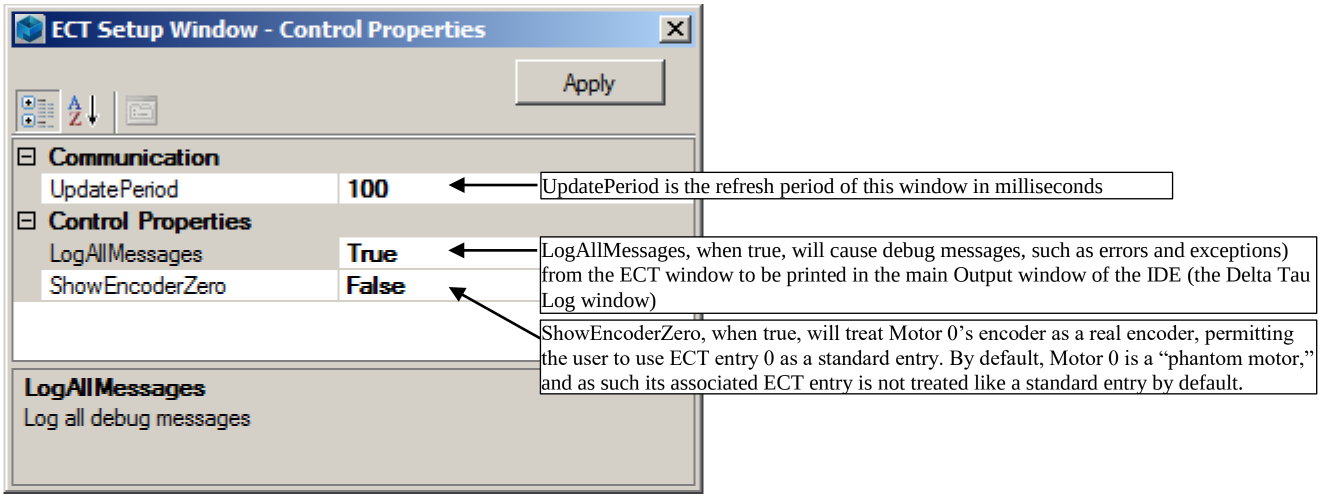


The user can refresh the information on this window by right-clicking and selecting Properties→Control→Refresh:

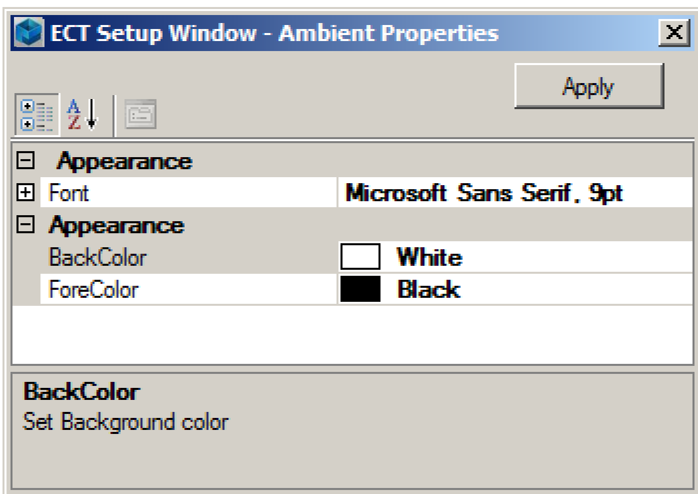


If the user selects "Clear Messages," the information in the output area at the bottom of the window will be cleared.

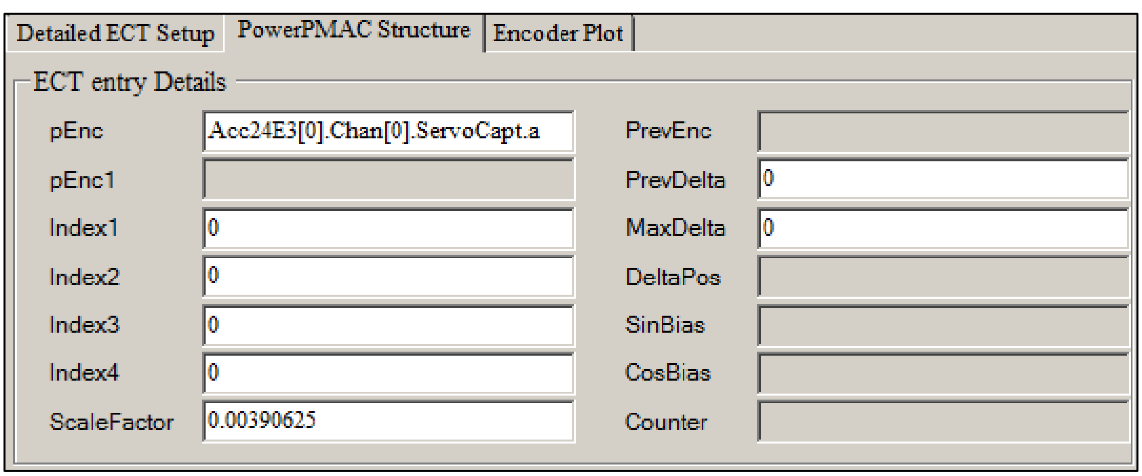
Selecting “General” will open a dialog with properties for the ECT window:



To change the font colors and sizes Click “Ambient” as shown below:

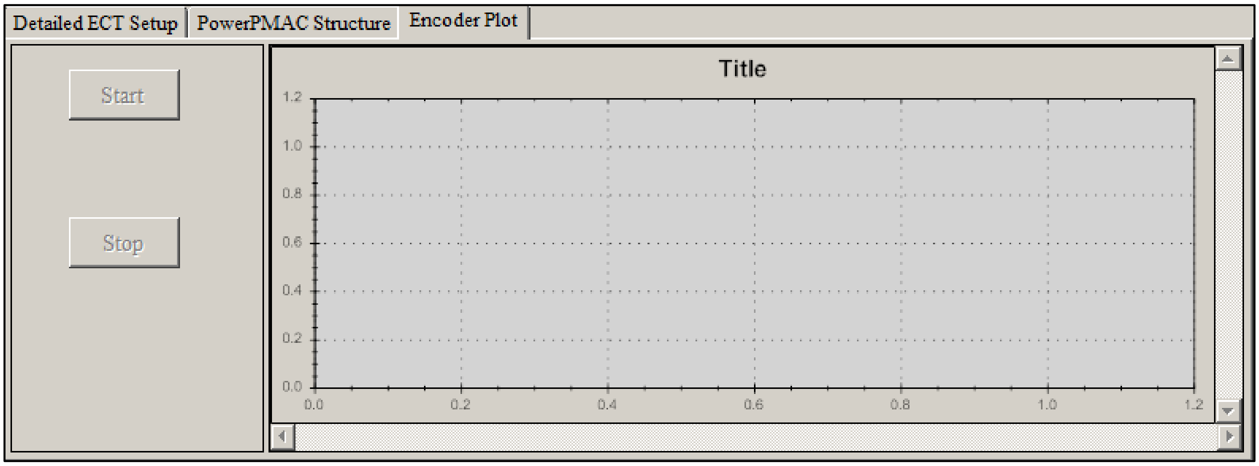


Clicking the “Power PMAC Structure” tab shows the following:

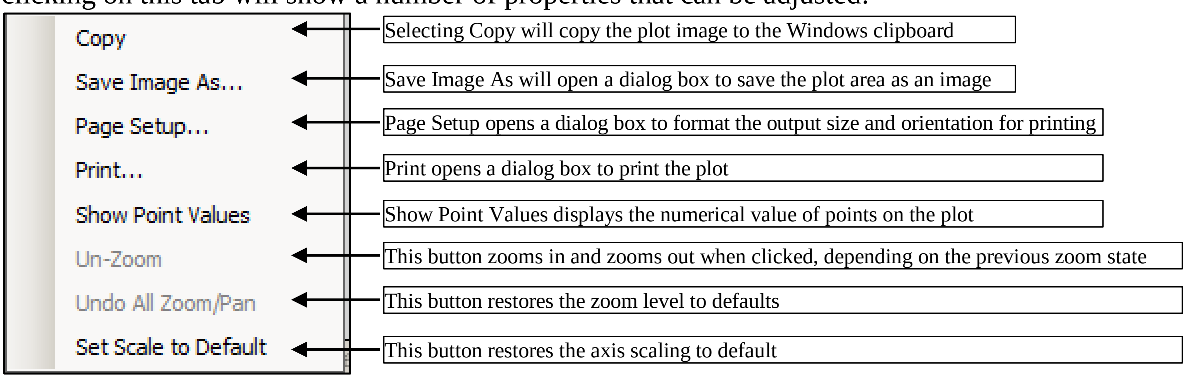


This tab displays all of the fields of the **EncTable[x]** structure which can be configured through this tab. Type the value for the modified field set. For more detail on what each field does please refer to the Power PMAC Software Reference Manual.

Clicking the "Encoder Plot" tab displays a scope of the presently selected Encoder Conversion Table entry's output (Not yet implemented):



The ECT scope can be started and stopped by clicking the Start and Stop buttons respectively. Right-clicking on this tab will show a number of properties that can be adjusted:



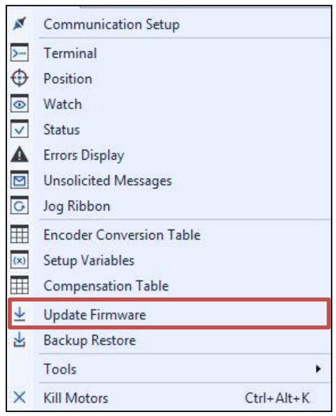
Update Firmware

Standard Firmware Download Procedure

Note

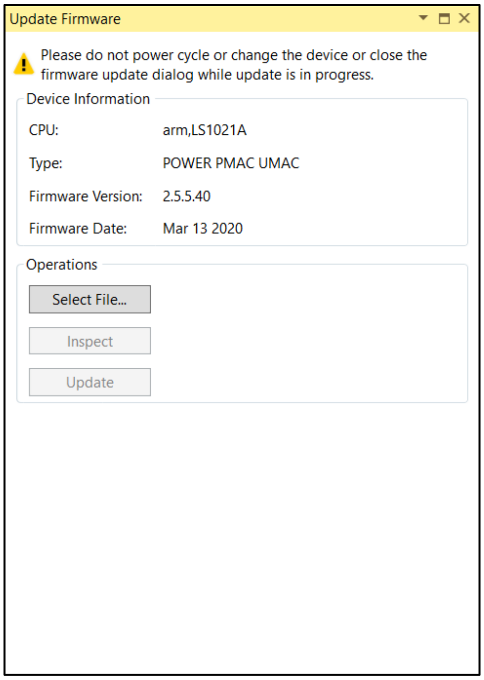
The latest released version of the Power PMAC firmware should always be used, if the application permits

To install the latest firmware, click on Delta Tau→ Update Firmware:

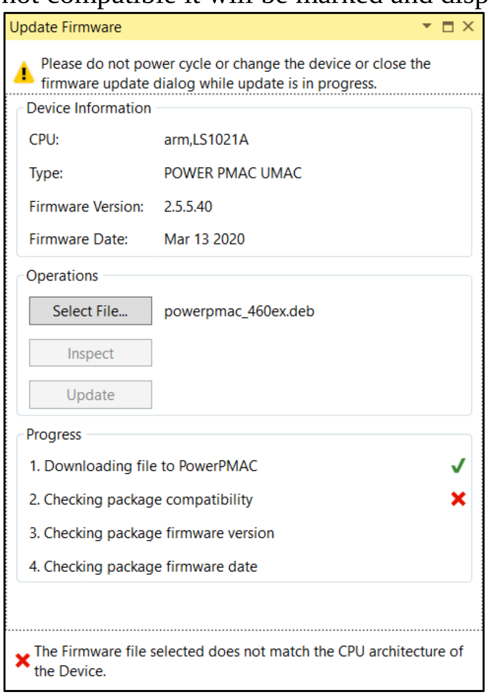


On clicking, the Firmware dialog will be opened. This is the improved view in comparison with the previous UI.
Device Information: Displays firmware information about the currently connected Power PMAC.
Progress: Displays the progress of the Inspect firmware or update firmware.

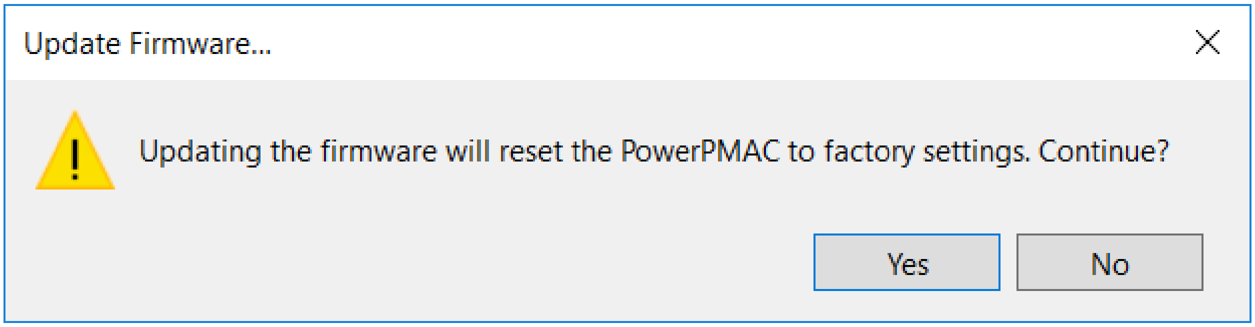
There are three buttons:
Select File: Allows the user to select the firmware file.



Inspect: This button will inspect but not update the CPU compatibility and file compatibility. If the file is not compatible it will be marked and display the appropriate error like this…



Update: On clicking the button, it will show the dialog, informing the user that it is recommended to issue the $$$*** command to bring Power PMAC to a factory reset condition.



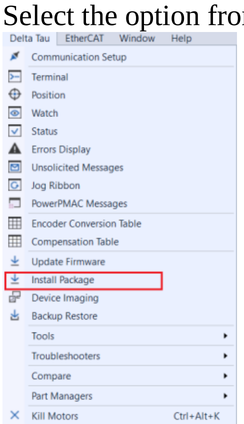
Clicking Yes will continue updating the Firmware. No, will abort the FW update and now the user can save the current Power PMAC state before updating the firmware. Please monitor the progress box for update process progress. On success the Device Information dialog will refresh to display the new firmware version and date.

Caution

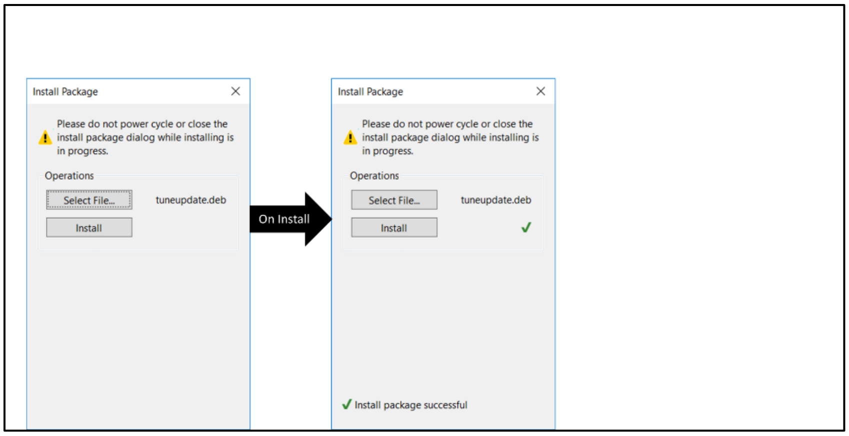
Please do not Power Cycle, Reboot, change device or close the Firmware Update dialog while the update in progress. Any attempt to do so will result in the board malfunctioning.

Install Package

As the name suggests, this dialog allows the user to update the Linux packages in case of factory recommendations.
This dialog is mostly useful if the FW requires small patch updates rather than full firmware upgrade.
Select the option from Delta Tau – Install Package, like…



On selecting, the menu Install package dialog opens as shown below…



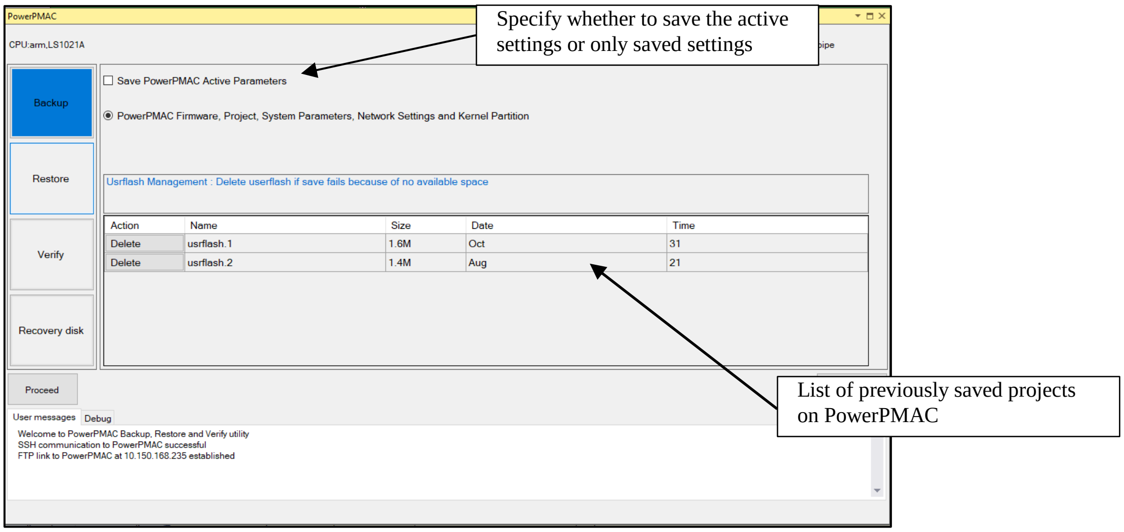
Select the packag file (.deb) and press Install to install the pckage. On success it will be marked and status will display the successful update message.

Backup Restore

The Backup Restore window has four pages: Backup, Restore, Verify, and Recovery Disk.

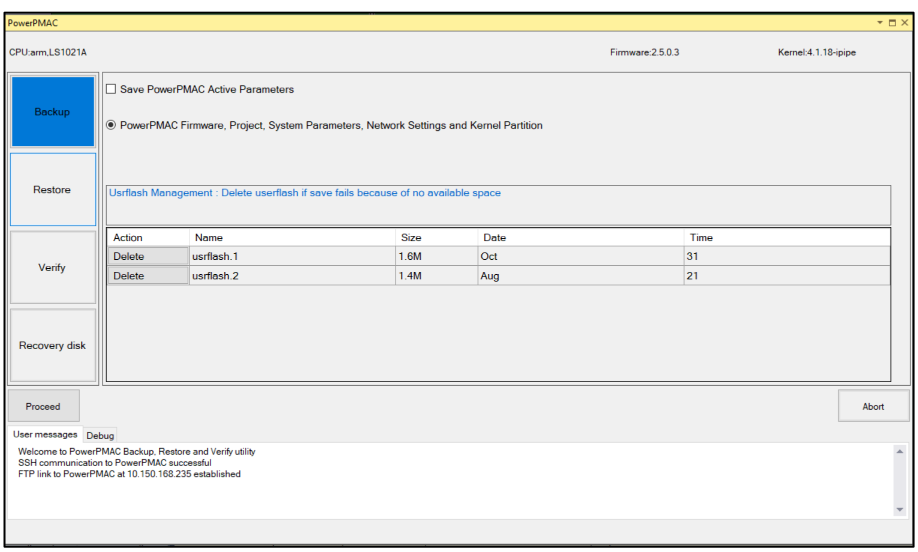
Backup page

The “Backup” screen looks like the following:



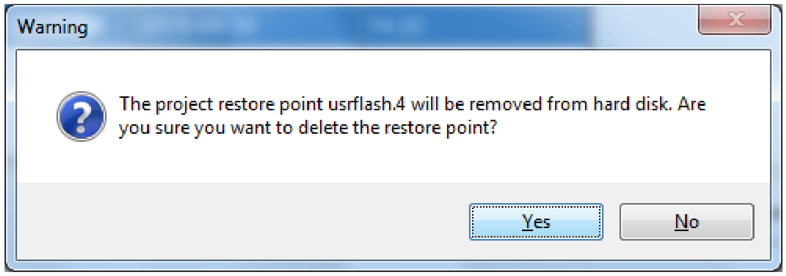
1. The first screen that appears when the Backup Restore is clicked shows possible option. This (i.e. the “Typical” backup) makes a backup of all the Power PMAC-related settings and files.

2. In addition to the typical backup options, the screen below allows user to manage previously created project restore points “usrflash.x” like shown below:



If the previous restore points are occupying large space in Power PMAC and the **save** process fails due to lack of space on the Hard Disk, then the user has the option to delete previous restore points to free space on the Hard Disk by clicking the Delete button next to the restore point. Note that Power PMAC automatically creates a restore point every time the **save** command is issued.

Pressing delete displays the following dialog:

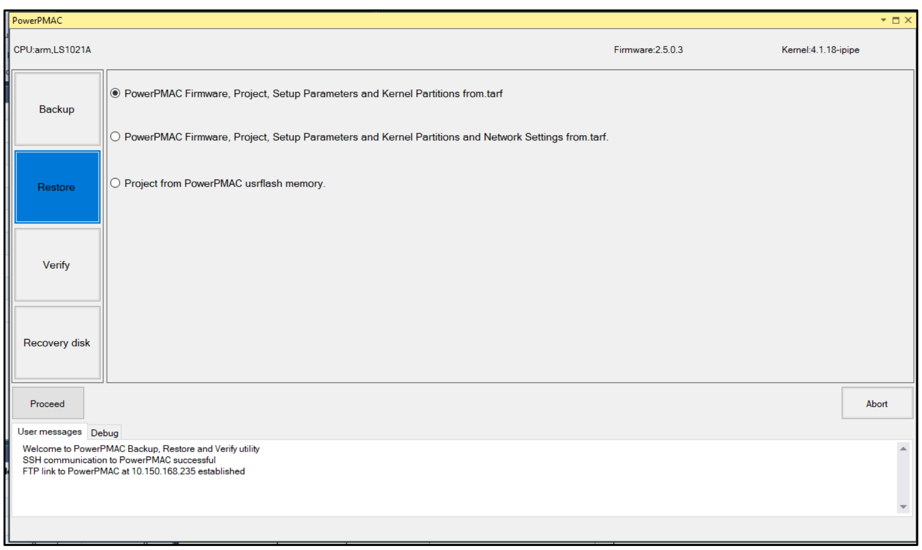


Note:

1. For the CPU type “x86, Hypervisor” there is no need for previous restore points and therefore are not listed in the above screen.
2. The Disk Image option is not needed and therefore is hidden from the main screen when communicating to a CPU type “x86, Hypervisor.”

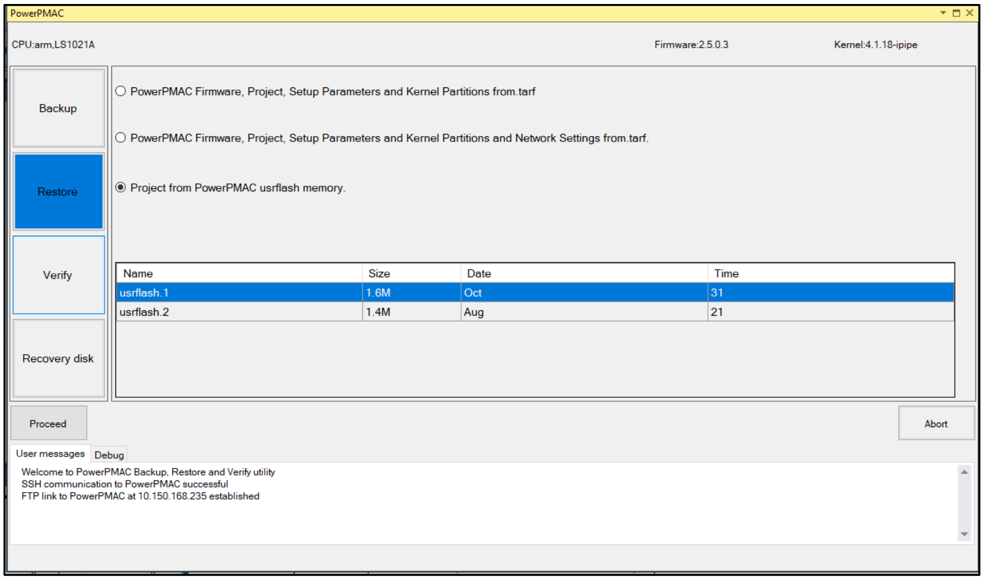
Restore page

Clicking on the “Restore” button shows this screen:



The Restore screen has different choices for different CPU types. For CPU types “PowerPC,460EX” and “PowerPC, XPM86xxx”, this screen offers 3 choices:

1. Firmware, project, setup parameters and kernel restore from a .tar file backup.
2. Regular restore, plus restores network settings. If selected this will replace the IP Address, Subnet Mask and Default Gateway settings with those from the backup configuration in addition to firmware, project, setup parameters and the kernel.
3. The third choice allows the restore of project only files from previously saved restore points. These restore points are already stored on Power PMAC:



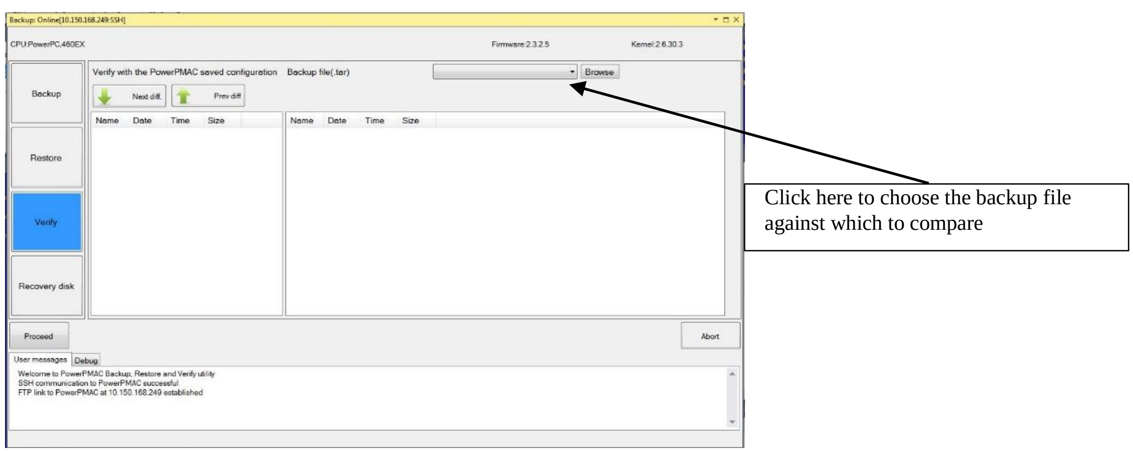
Note: Similar to the Backup screen, for the CPU type “x86, Hypervisor”, there are no previously saved restore points and therefore 2nd and 3rd options are disabled for this type of CPU.

If the selected drive is a shared folder on the host computer, then it requires login credentials for that drive to mount that folder on Power PMAC before it can restore the image from the source disk.

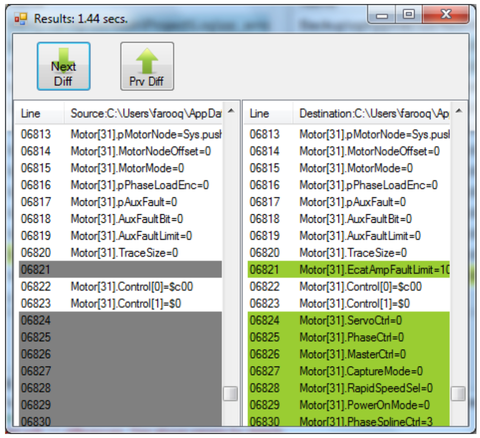
Clicking “Proceed” will cause the program to prompt to browse for the backup file. It wants a “.tar” backup file that was generated using the Backup Restore program previously.

Verify page

Clicking on the “Verify” button shows this screen:



This screen is used to compare either the Active or the Saved configuration in the Power PMAC against a backup file that has been previously created. Clicking “Proceed” will show the differences between these two configurations highlighted in blue as shown below:
This page now has a program implemented to generate the difference in the files.



Note: Verify only compares projects and related files.

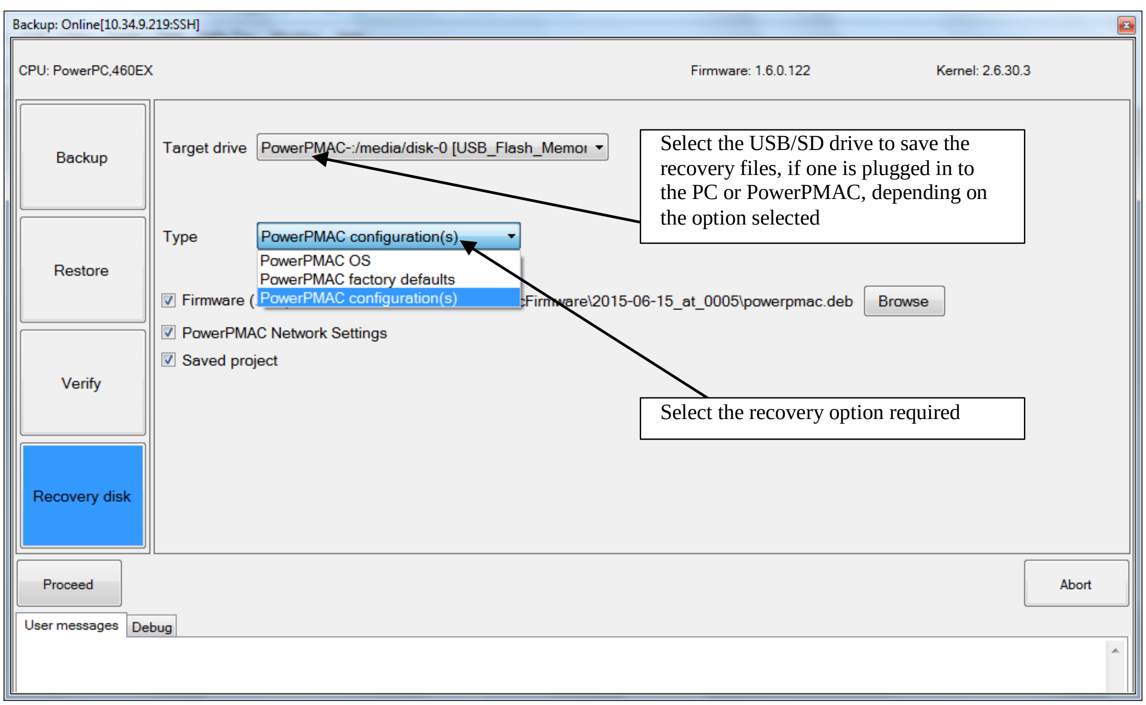
Recovery Disk page

Clicking on the “Recovery Disk” button shows the screen below:

This is used to create a recovery disk that can be saved to a USB or SD drive for restoring various settings on Power PMAC. The functionality of this tab depends on the recovery option selected under the “Recovery Option” field.

Selecting “Power PMAC Firmware Install,” requires a USB or SD drive to be entered into the PC operating the IDE and the selection of a firmware file (with .deb extension) on the PC to be installed into the Power PMAC.

For all other options a connection must be made to a Power PMAC and the USB or SD drive should be entered into the Power PMAC.



The various recovery options available are listed below:

Recovery Option	Description
Power PMAC OS	This option creates a recovery disk for the Operating System without restoring the Power PMAC's firmware.
Power PMAC Factory Defaults	This option creates a recovery disk for Power PMAC to start up in factory default mode ($$$***).
Power PMAC Configuration(s)	This option makes a recovery disk for Power PMAC based upon which checkbox is selected. Any one, two or all three options can be selected from firmware, the Power PMAC's configuration (Active or Saved) (stored on Power PMAC's disk) and Power PMAC's network settings.

Device Imaging (Backup & Restore)

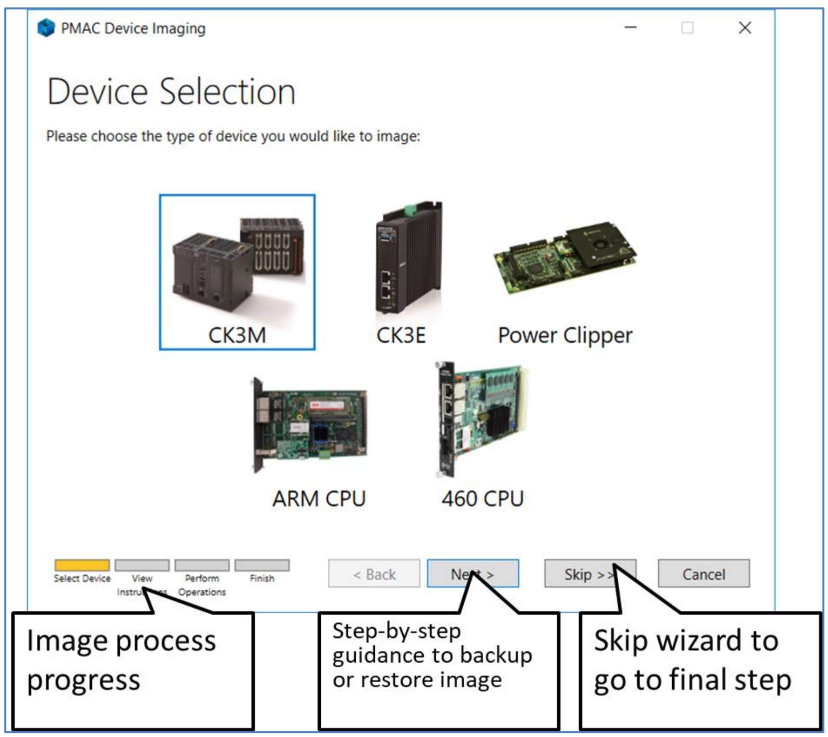
The “Device Imaging” option is available from Delta Tau Menu.

The User can use this to backup or restore the Power PMAC image.

The User is given guided instructions on how to connect to and image Power PMAC devices using a USB cable.

The User will be able to change the IP address at the time of restoring the image. This process will also retain EtherCAT license information if the option is present.

When User selects “Device Imaging” a wizard style dialog will be launched and will walk the User through the full process. This launch view is shown below:

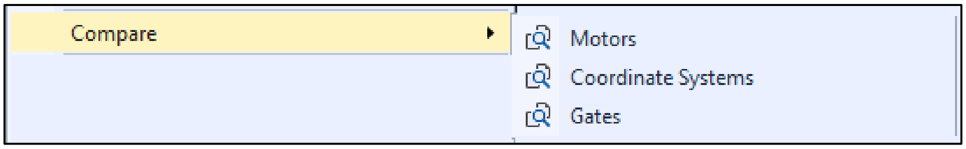


Imaging requires the Power PMAC power to be switched off. The User needs to issue a Save command if needed.

Note

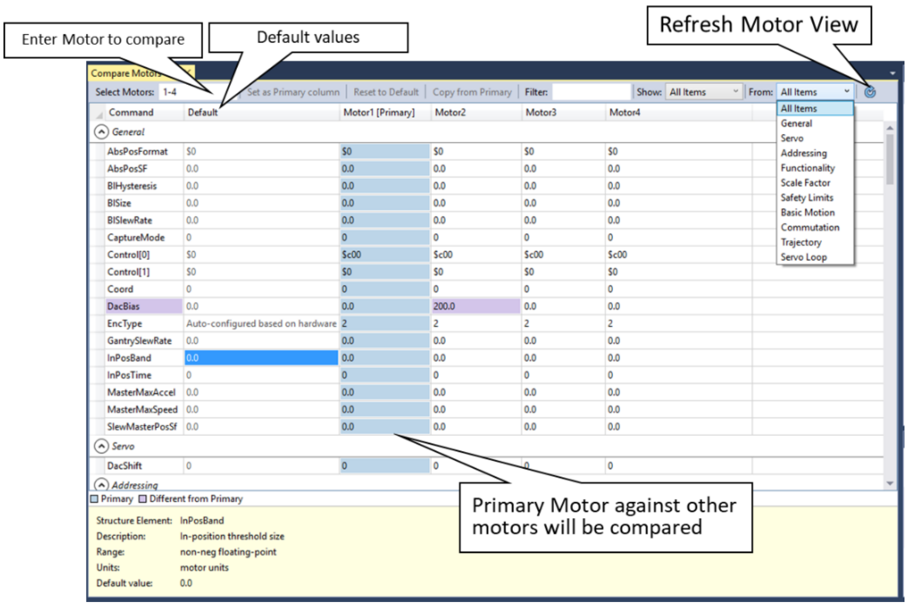
Compare

Compare Motor or Coordinate System options are available from the Delta Tau Compare Menu.
On clicking the menu, the user will have the choice of comparing motor or coordinate systems as shown below…



Motors

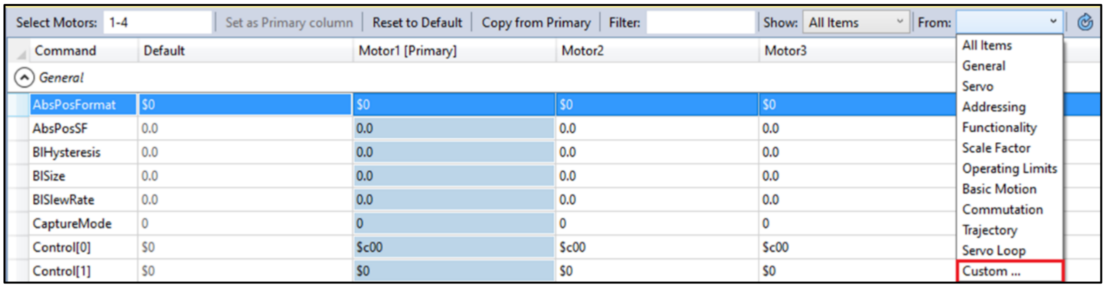
Clicking this option will open the motor compare dialog as shown…



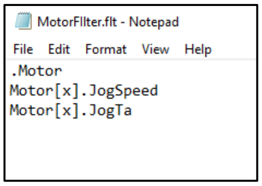
Only the saved motor structure elements are compared.

User can...

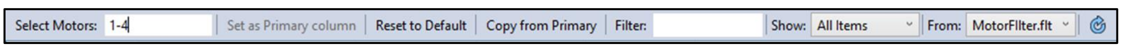
1. View saved structure elements.
2. View the current motor structure elements against the factory default ($$$***) motor structure elements.
3. Set any motor as primary and the other motor structure elements are compared against the it.
4. Visually identify the differences in motor structure elements between primary and regular motors.
5. Edit the motor structure elements and updates the Power PMAC on entering the value.
6. Copy and paste single/multiple motor structure element cells from the primary motor or default.
7. Reset the motor structure element values to factory default with a Reset to Default command available from the dialog.
8. Quickly search for an element by either typing the text in the filter or picking a category from the drop-down list.
9. Supports special custom filter. This feature allows user to customize the elements most commonly used. The custom filter selected as shown below…



On clicking custom filter it will look for .flt file. This is simple ini file format. Typical file looks like this…



User can add any motor saved structured element as shown above and save the file as .flt.
.Motor is the category and it is mandatory to add .Motor on top of the Motor element filter file.
The benefit of this feature is when you have multiple motors say 10, of same type and same feedback then you can fully setup one motor and then copy all the setting across for selected structure element using custom filter. Typical case will be EtherCAT motor.
The custom file will be displayed in the Filter view and will be part of the list too. This will retain for the current IDE session. It will look like this..



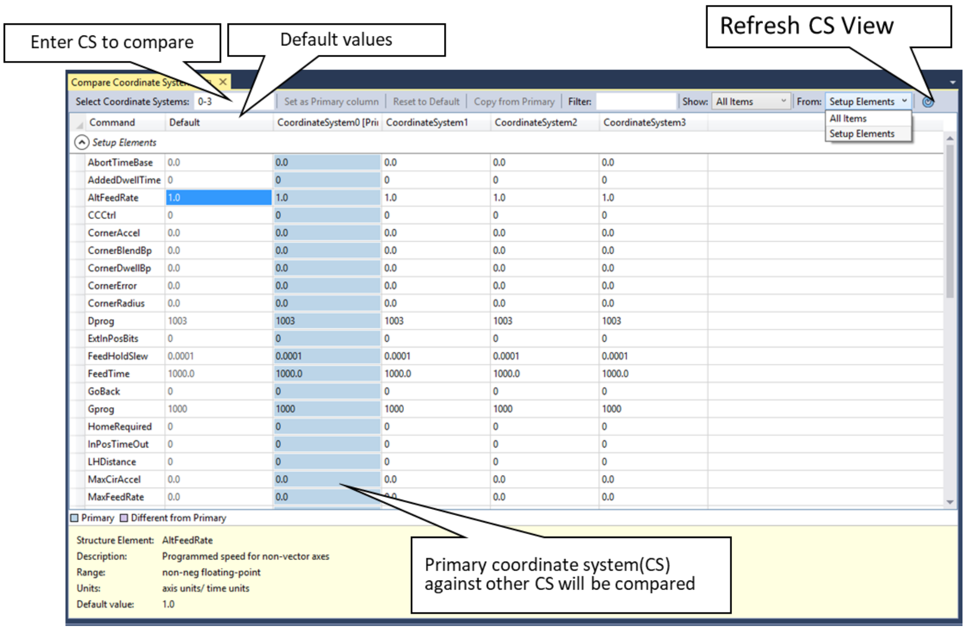
10. Refresh the motor compare dialog if the values have been changed after downloading the project.
11. View the description of every motor structure element, such as description, range, unit and default value.

Note

Motor compare shows only Power PMAC saved motor structure elements.

Coordinate Systems

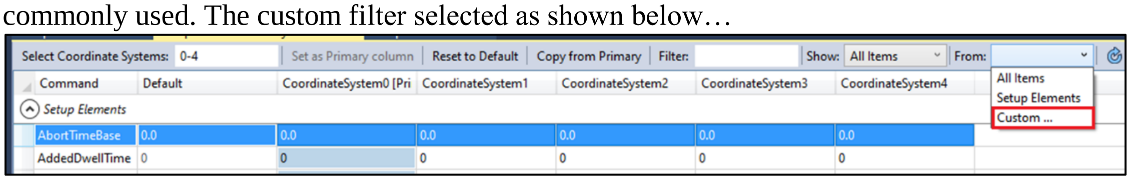
Clicking this option will open the Coordinate system compare dialog as shown…



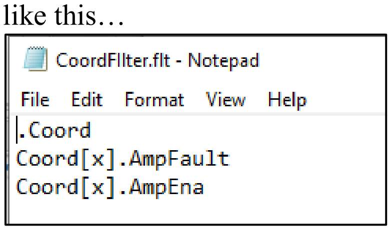
Only the saved coordinate system structure elements are compared.

User can…

1. View saved structure elements.
2. View current coordinate system structure elements against factory default values ($$$***).
3. Make any coordinate system the primary coordinate system and other coordinate system structure elements are compared against the primary column.
4. Visually identify the different coordinate system structure elements between primary and regular coordinate system.
5. Edit the coordinate system structure elements and updates the Power PMAC on entering the value.
6. Copy and paste single/multiple coordinate system structure element cells from primary or default.
7. Reset the motor structure element values to factory default with a Reset to Default command available from the dialog.
8. Quickly search for an element either typing the text in the filter or picking a category from the drop-down list.
9. Supports special custom filter. This feature allows user to customize the elements most commonly used. The custom filter selected as shown below…



On clicking custom filter it will look for .flt file. This is simple ini file format. Typical file looks like this…



User can add any coordinate saved structured element as shown above and save the file as .flt.

.Coord is the category and it is mandatory to add .Coord on top of the coordinate element filter file. The custom file will be displayed in the Filter view and will be part of the list too. This will retain for the current IDE session. The filter file name will displayed similar to picture from Motor compare view.
Refresh the coordinate system compare dialog if the values have been changed after downloading the project.

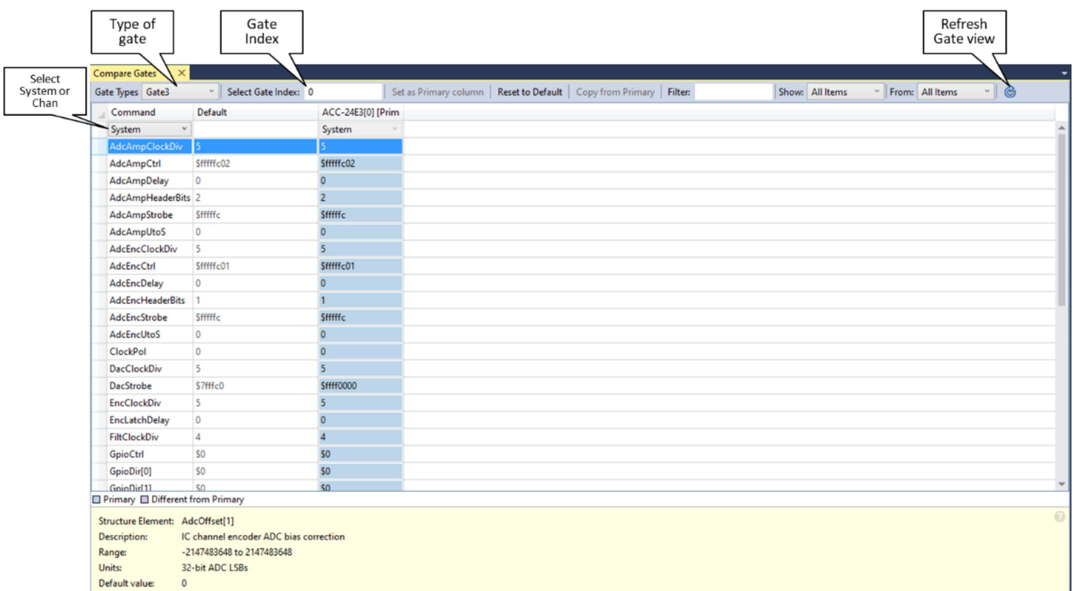
10. View the description about every coordinate system structure element, such as description, range, units and default values.

Note

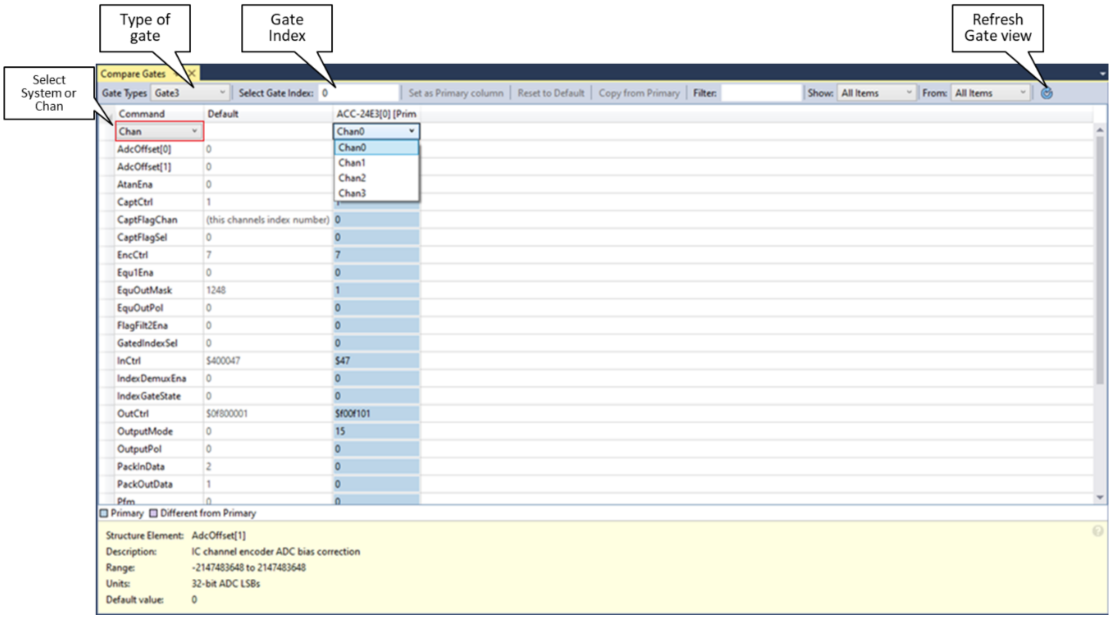
Coordinate system compare shows only Power PMAC saved coordinate system structure elements.

Gate structure element

This supports Gate1 and Gate3 structures. Depending on the detected hardware the type of gate drop down will automatically populated. User can select type of gate and enter the index. Gate index can be found from Project Hardware mode. Hoovering the mouse on the text box will guide how to enter the index number. If you multiple gates then this feature very useful in comparing gate saved structure element. Choose the chan structure elements from drop down under command column.
Clicking this option will open the Coordinate system compare dialog as shown…

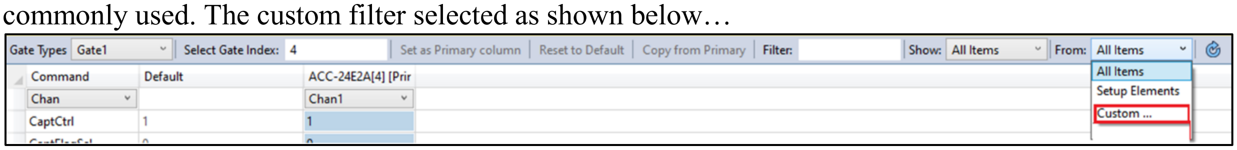


To view the Gate-channel user need to select Chan from the Select System or Chan drop down list. Marked Red Square.

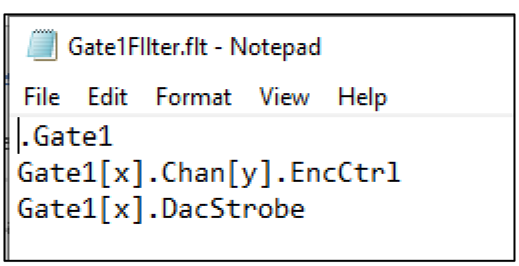


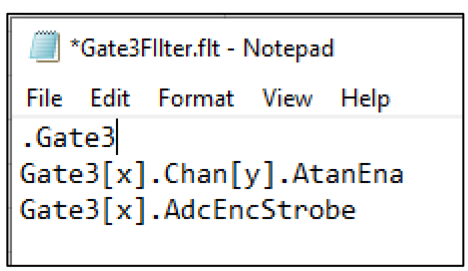
User can…

1. View saved structure elements.
2. View current Gate structure, Gate-Chan elements against factory default values ($$$***).
3. Make any Gate system or Gate-Chan the primary column and other Gate system or Gate-Chan structure elements are compared against the primary column.
4. Visually identify the different Gate system or Gate-Chan structure elements between primary and regular Gate system or Gate-Chan elements.
5. Edit the Gate system or Gate-Chan structure elements and updates the Power PMAC on entering the value.
6. Copy and paste single/multiple Gate system or Gate-Chan structure element cells from primary or default.
7. Reset the Gate system or Gate-Chan structure element values to factory default with a Reset to Default command available from the dialog.
8. Quickly search for an element either typing the text in the filter or picking a category from the drop-down list.
9. Supports special custom filter. This feature allows user to customize the elements most commonly used. The custom filter selected as shown below…



On clicking custom filter it will look for .flt file. This is simple ini file format. Typical file looks like this…





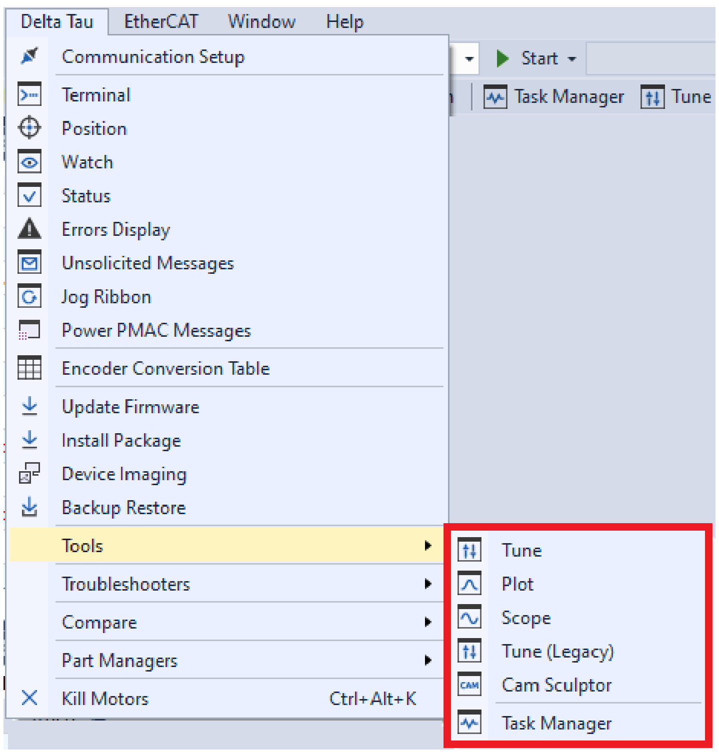
User can add any coordinate saved structured element as shown above and save the file as .flt.

.Gate1 is the category and it is mandatory to add .Gate1 or Gate3 on top of the Gate filter file. The custom file will be displayed in the Filter view and will be part of the list too. This will retain for the current IDE session. The filter file name will displayed similar to picture from Motor compare view.

10. Refresh the Gate system or Gate-Chan compare dialog if the values have been changed after downloading the project.
11. View the description about every Gate system or Gate-Chan structure element, such as description, range, units and default values.

Tools

The Tools submenu from Delta Tau menu shows Cam Sculptor, Advanced Tunng, Plot, Scope and Task Manager. This is shown below inside the red box:

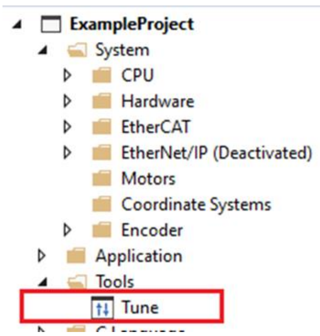


Tune

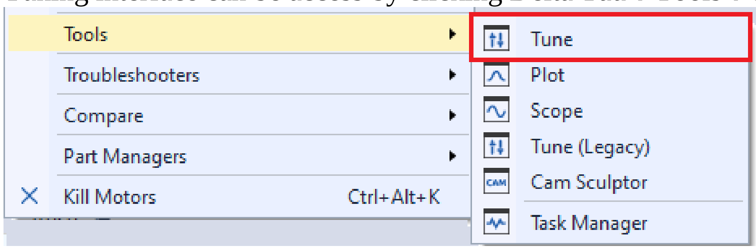
The Tuning tool can be used to tune current loops and position (servo) loops the motors. “Tuning” refers to the process of adjusting the gains in the control loop until the desired performance level is achieved. This is a complete new tuning interface developed considering the usability and clean and clear selection option.

New tuning interface is integrated with project. The tuning can be open from the project by double clicking the Tune from Tools node from project as shown below. It is our recommendation to the user to use the Tuning from Project. Benefit of using the tuning from project …

1. Fully integrated to the project
2. On accepting the gain settings from tuning are written to the motor setup file.
3. Tuning settings are stored in the project. For example the move size, dwell time, filter frequency values etc. will be per project.
4. Power PMAC IDE will continue to enhancing integration of tuning in the project.



For the user who does not want to open project can open the tuning from Delta Tau menu, though it is not recommended.
Tuning interface can be access by clicking Delta Tau→Tools→Tune as shown below.

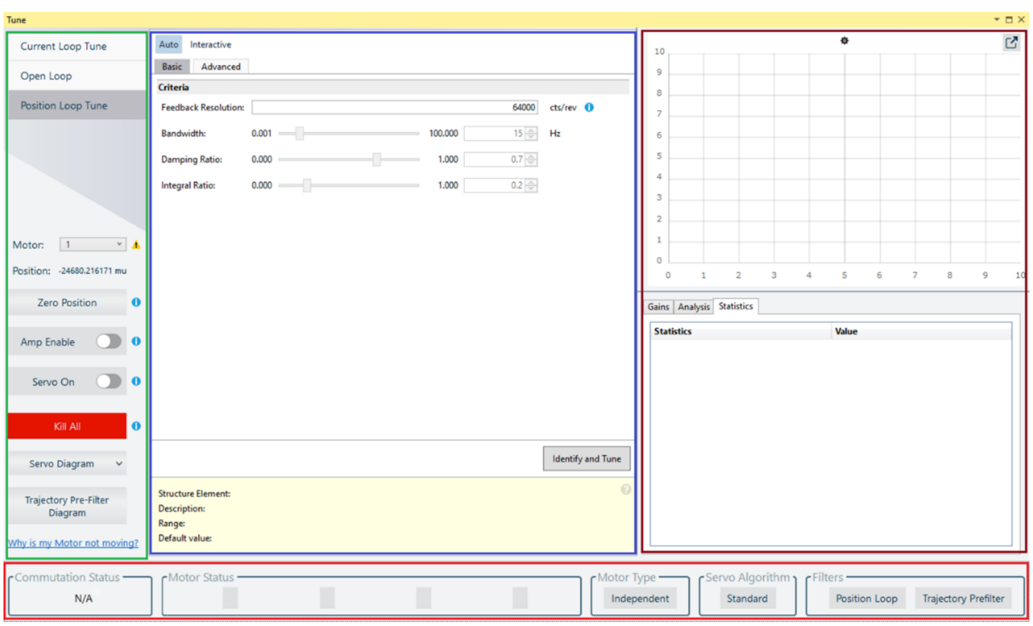


Note

We recommend to use Tuning from Project menu and not from Delta Tau menu.

Tuning Window Layout

As shown below when Tuning is open from Project or from Delta Tau menu it will look like…



There are four section as marked by different color. In following section each colored box will be explained. Resizing of Graph section is possible and can have bigger size.

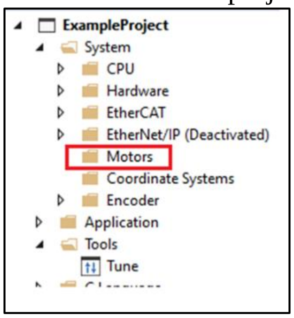
Common convention

Info icon on hoovering the mouse will display additional help regarding the setup parameter.

Tuning mode, Motor section

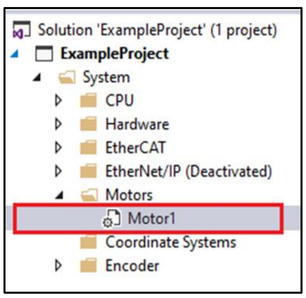
The left panel marked by Green box allows user to select type of tuning and on which motor.
In the picture you can see the yellow warning icon next to the motor number drop down. This Yellow warning sign, warns user that the motor that is used for tuning is not part of the Project. There are two possibilities ….

1. Motor is not in the project as shown below.. there is no motor under Motor Node.

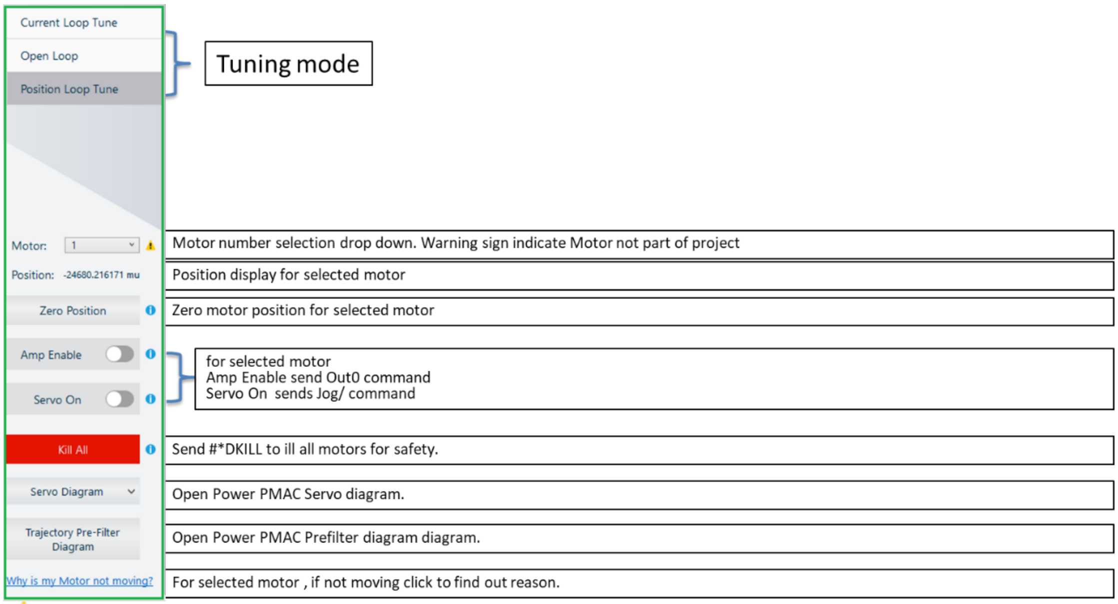
   Message will be: Motor <number> is not in the project.

2. Motor is under Motor Node but not fully configured. As shown below …

   In this case the warning sign will say
   Message: Motor <number> has not been fully configured through System Setup.
   If the user has added the motor but not completed going through Topology blocks.
   Message: Motor <number> has not completed Basic Tuning in System Setup.
   If the user has added the motor and partially completed Topology block but not completed Basic tuning.

Following picture shows more details about left panel.



Warning sign near the Motor drop down will give following warning on hoovering the mouse… This motor was set up outside of the system setup environment. Tuning gains and Motor elements will not be saved automatically. It is the user's responsibility to update/save those in their pmh files.

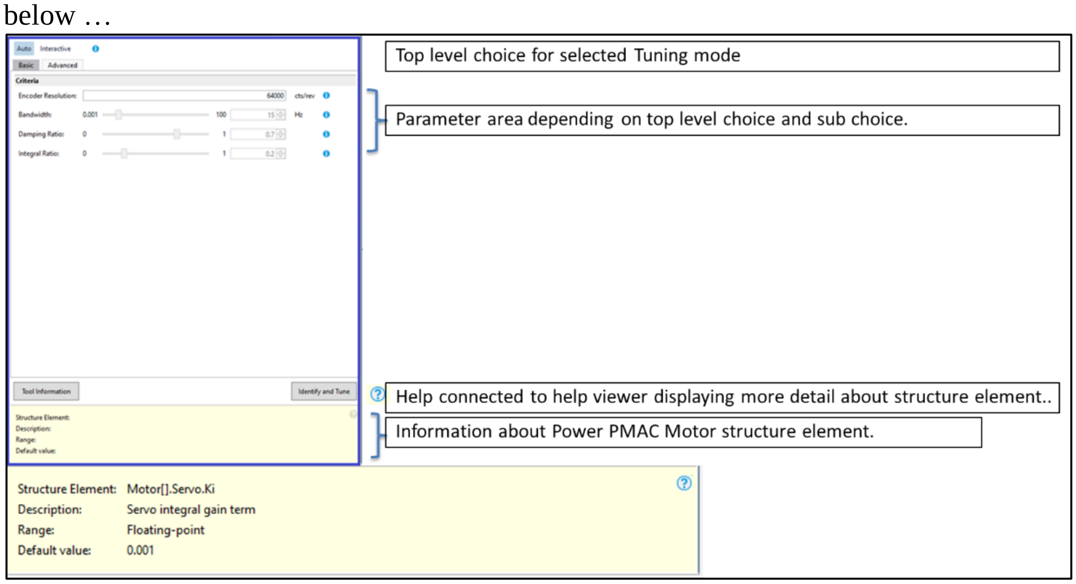
Note

We recommend user to use System Setup to setup Motor and initially tune using Basic Tuning Topology Block. This Basic tuning identifies system and comes up with gain settings. Using Advance tuning after Basic tuning will reduce tuning time.

.

Tuning parameter and performing tuning moves section

The middle section Blue area is for setting tuning parameters and performing tuning move. As shown below …



All the different Top level choices and sub level choices are self-explanatory.
Common control from this middle sections are explained below…

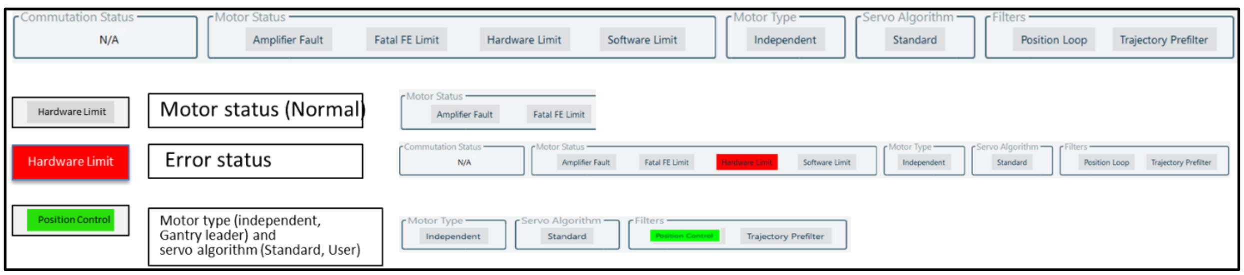
Tool Information	More information about the choices selected by user for Tuning
Identify and Tune	Automatically identifies and come up with basic set of gain limits.
Single Move	Makes a single tuning move based on user choices
Live Tune	Continuous tuning move. User can change the gain and see the result at next move.

Note

Since the servo loop gains change as they are altered in the Tuning window only safe gains must be entered in order not to damage the motor or cause it to go unstable, potentially damaging equipment or people.

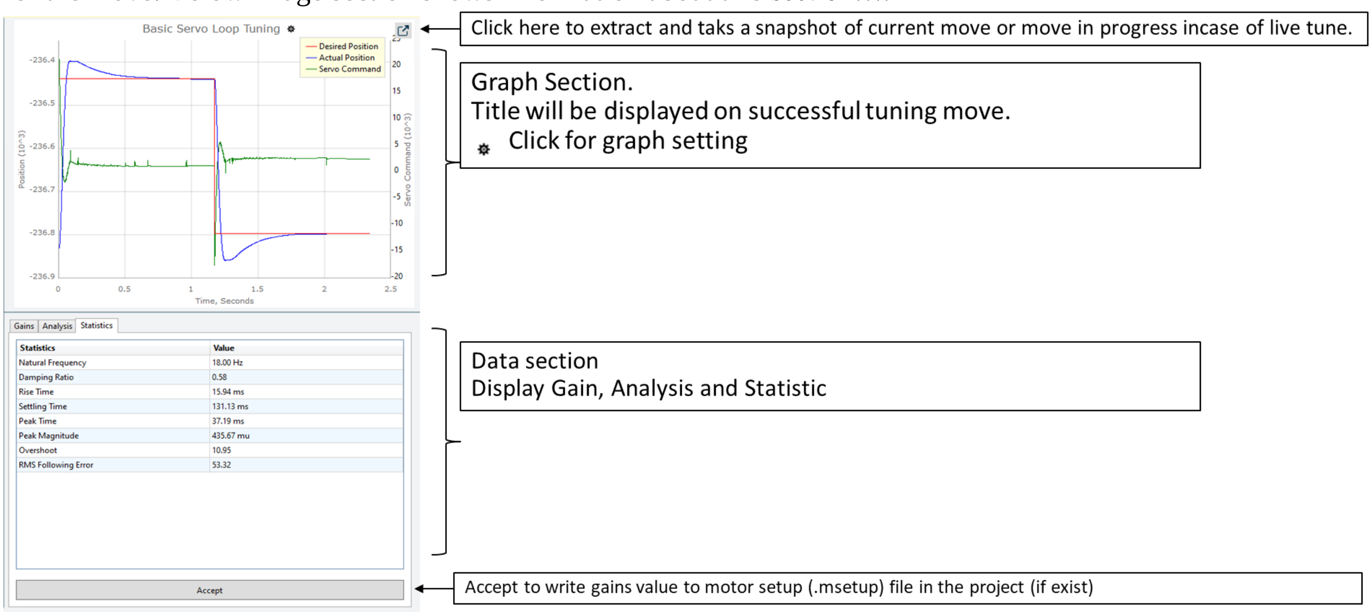
Tuning status section

The bottom Red square shows the Motor status of currently selected motor. The statuses are grouped in logical way. Status is continuously updated. The status area looks like this..



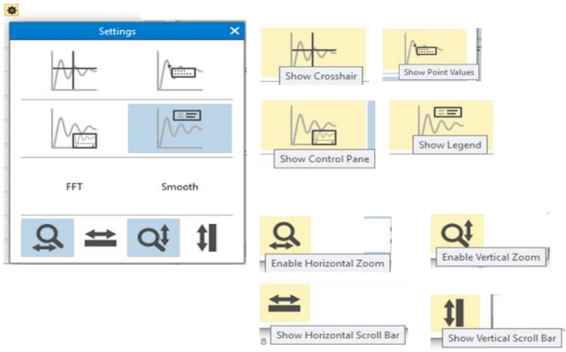
Tuning Result section

After successful tuning move the result will be displayed as a graph in the tuning result section. It is marked with Brown color. The bottom part of Graph shows result, analysis and Statistic in tabular form for the move. Below image section shows information about this section….

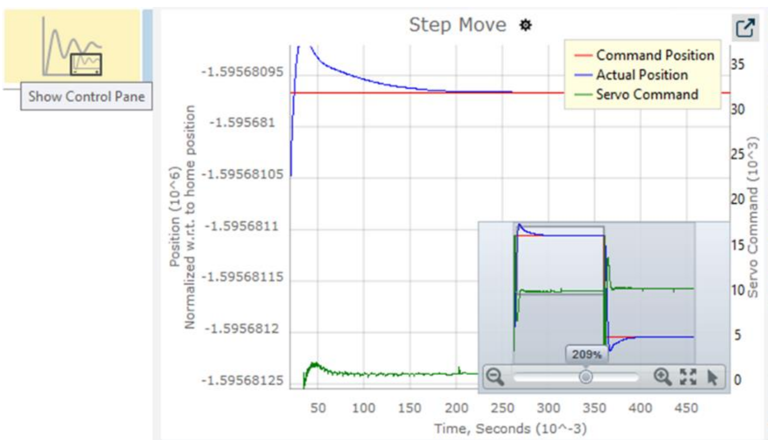


Statistics	Value
Natural Frequency	18.00 Hz
Damping Ratio	0.58
Rise Time	15.94 ms
Settling Time	131.13 ms
Peak Time	37.19 ms
Peak Magnitude	435.67 mu
Overshoot	10.95
RMS Following Error	53.32

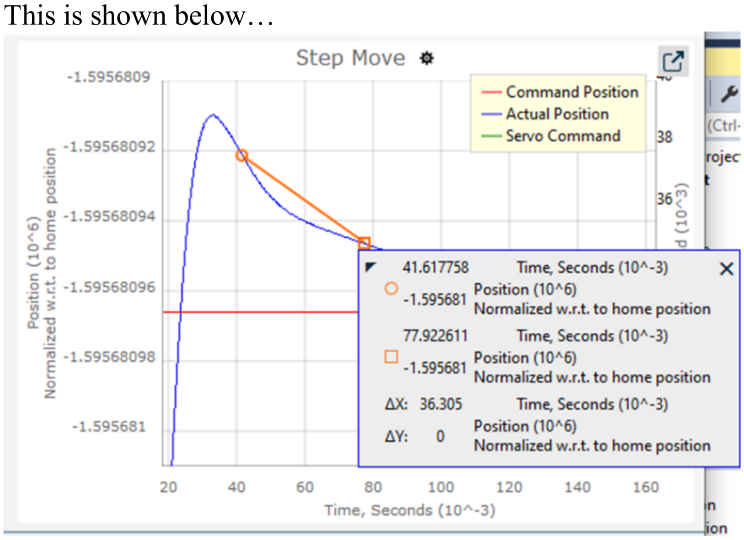
Graph settings are set of visual icons and on hoovering the mouse will display what are those settings. As shown below….



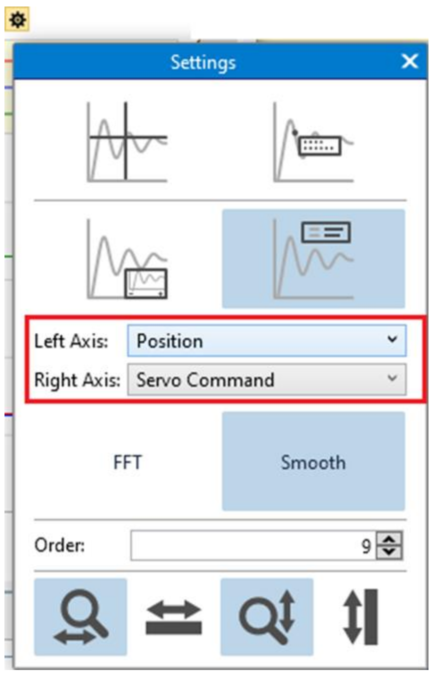
To Zoom in or out use mouse wheel or you also use Zoom pane as shown below. This will allow user to choose area to zoom.



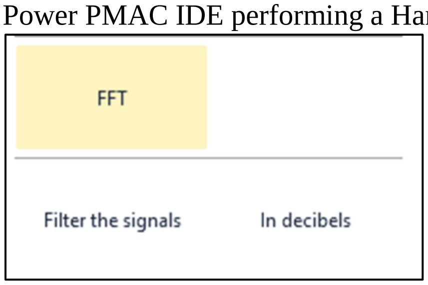
The graph also supports measurement. Click the point on the graph to see measurement between two points. The measurement is available all the time. Clicking the same point will remove the point.
This is shown below…



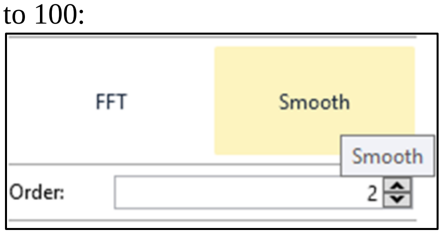
On successful tuning move the settings option dynamically changes and support user selecting Left and Right axis



FFT (Fast Fourier Transform) will perform (FFT) of the data. . Choose whether to filter the signals or not or whether to plot the vertical axes in units of decibels (dB). Choosing to filter the data will result in the Power PMAC IDE performing a Hanning window filter on the data



Smooth option will filter the signals chosen to plot with a moving average whose order can be set from 0 to 100:

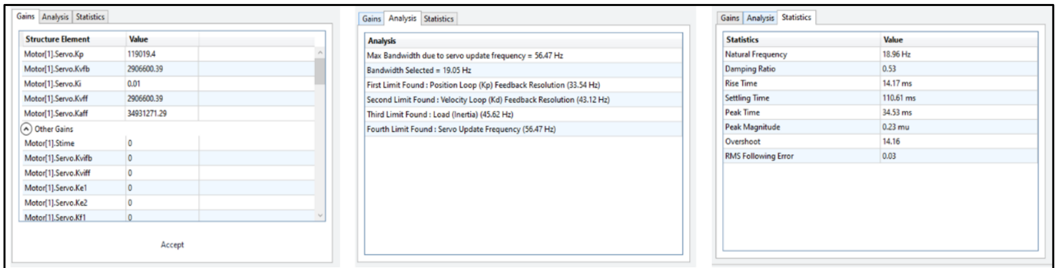


The default filter order is 2. The filter sums groups of points (the number of points in the sum is equal to the order of the filter) and then divides by the order of the filter. The equation of the filter is

$$p_i = \frac{1}{N}\sum_{k=0}^{N} a_k,$$

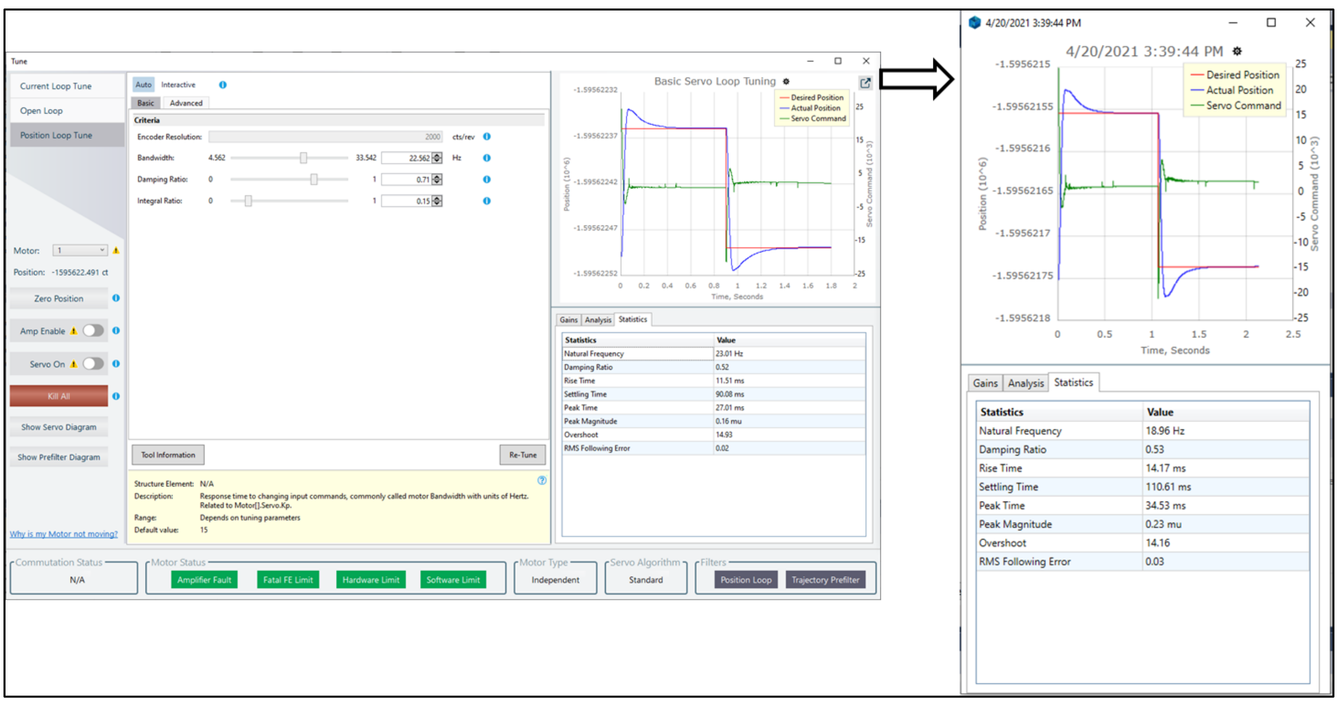
where p_i is a point on the plot, where i runs from 0 to the total number of points on the plot, N is the order of the moving average filter, and a_k is a point of data in the group of points of size (N + 1) presently being processed by the filter.

On successful tuning move data will be displayed under the graph. It will look like …



Accept only available on Gain Tab. On Accept the gains are written to motor setup file That is used in building systemsetup.cfg file on build and download.

 Clicking this icon on the graph will allow user take a snapshot of the current tuning performance and compare. The snapshot is static picture and the regular Tuning will continue. This is helpful if the user likes some tuning performance then a snapshot can be taken and user can keep adjusting tuning parameters to see if the performance improves or not.



Tuning Moves

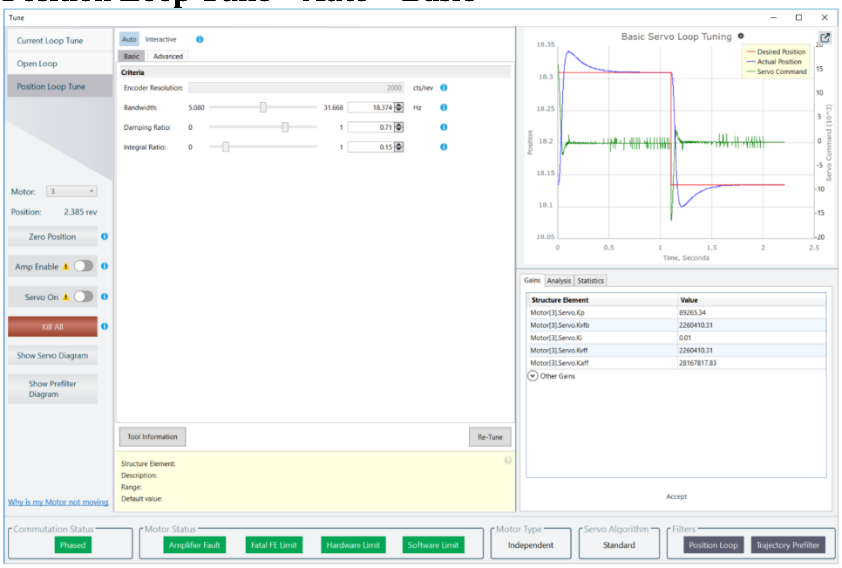
Auto Tune Moves

Please make sure that it is safe to do Tuning moves. Tuning moves, like Step moves can vibrate the machine.

It is recommended external Emergency Stop switch connected that will kill the amplifier power in case of motor runaway or loss of communication.

Note

Position Loop Tune – Auto – Basic



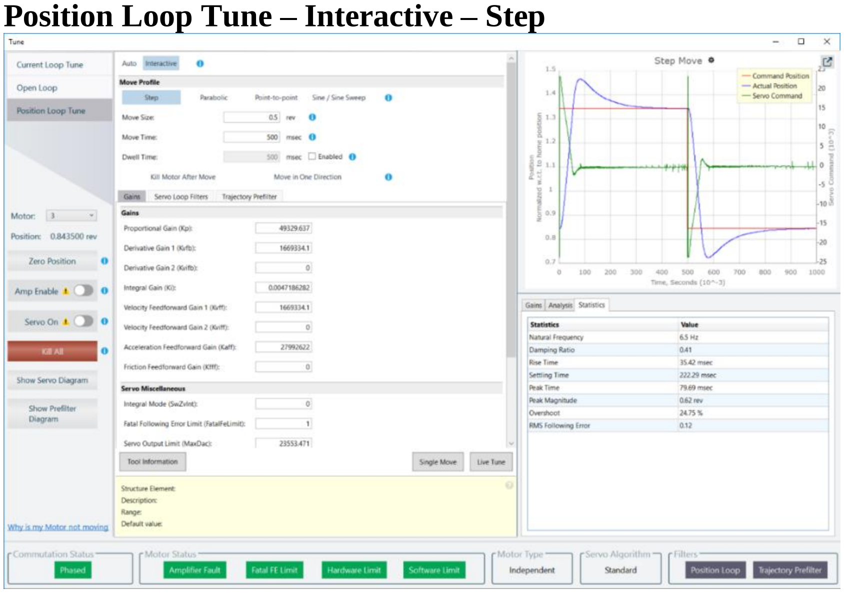
Interactive Tuning Moves

Please make sure that it is safe to do Tuning moves. Tuning moves can vibrate the machine resulting in machine damage.

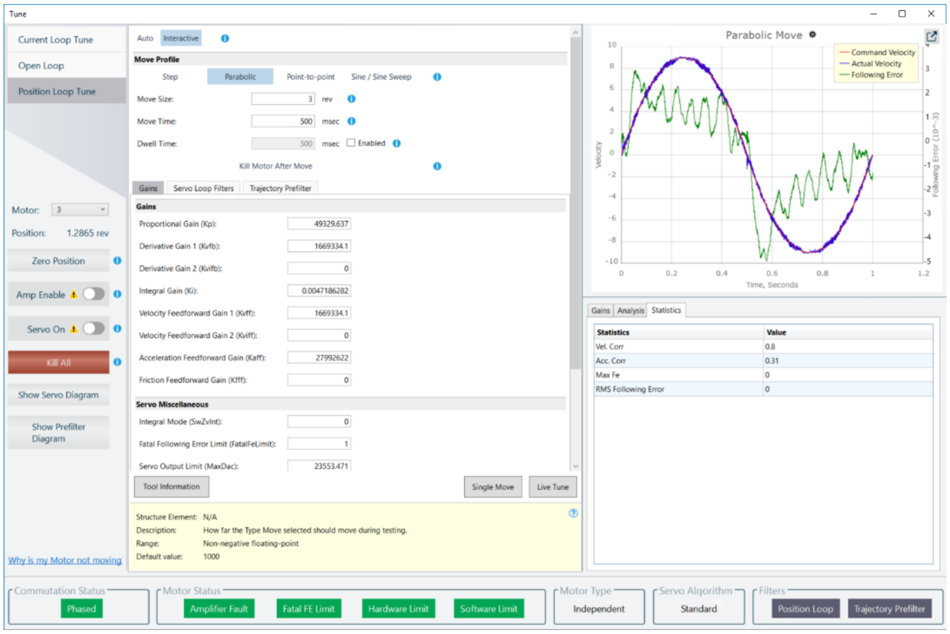
It is recommended external Emergency Stop switch connected that will kill the amplifier power in case of motor runaway or loss of communication.

Note

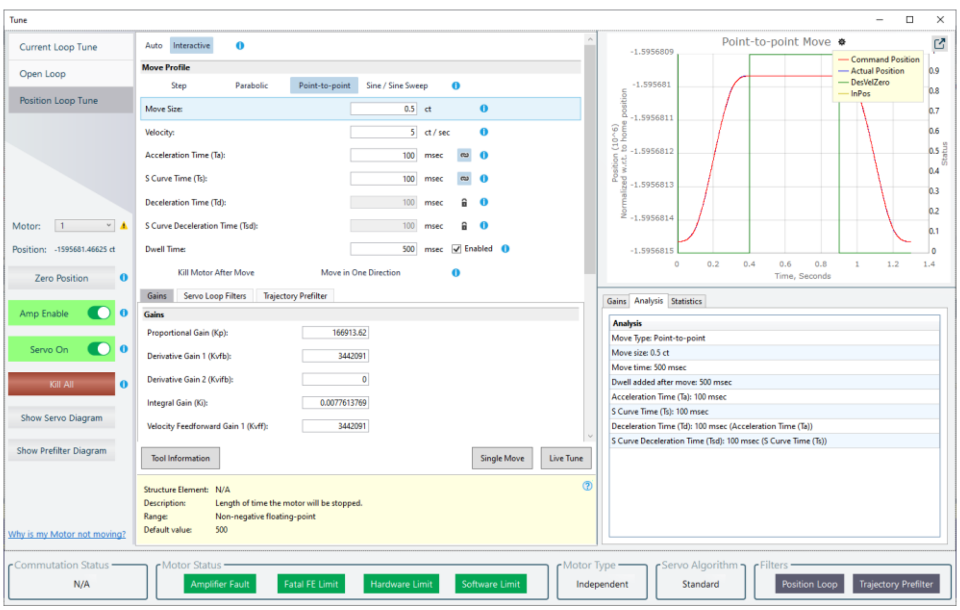
Position Loop Tune – Interactive – Step



Position Loop Tune – Interactive – Parabolic



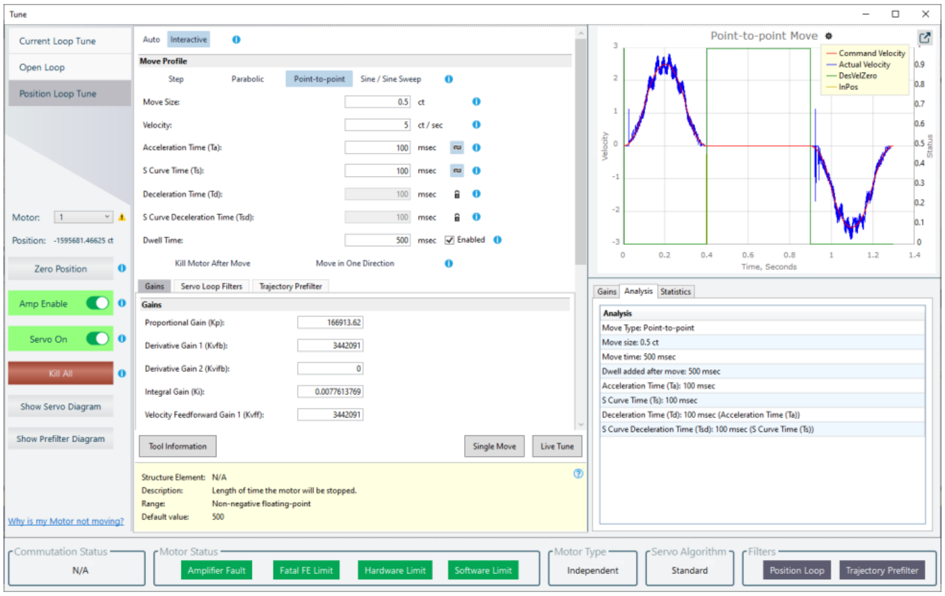
Position Loop Tune – Interactive – Point-to-point



Above plot showing Position with Desired Velocity and InPos.

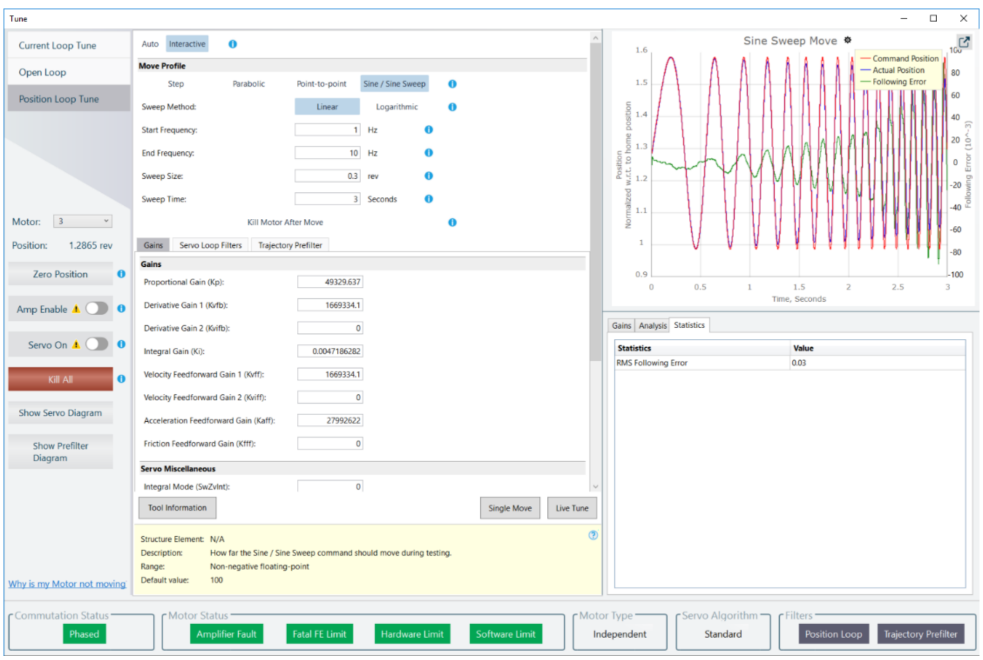
Note

To plot In Pos bit successfully user must set the In position band Motor[n].InPosBand.
Point-to-point move requires updating tuning package using Package installer.



Above plot showing Velocity with Desired Velocity and InPos

Position Loop Tune – Interactive – Sine/SineSweep



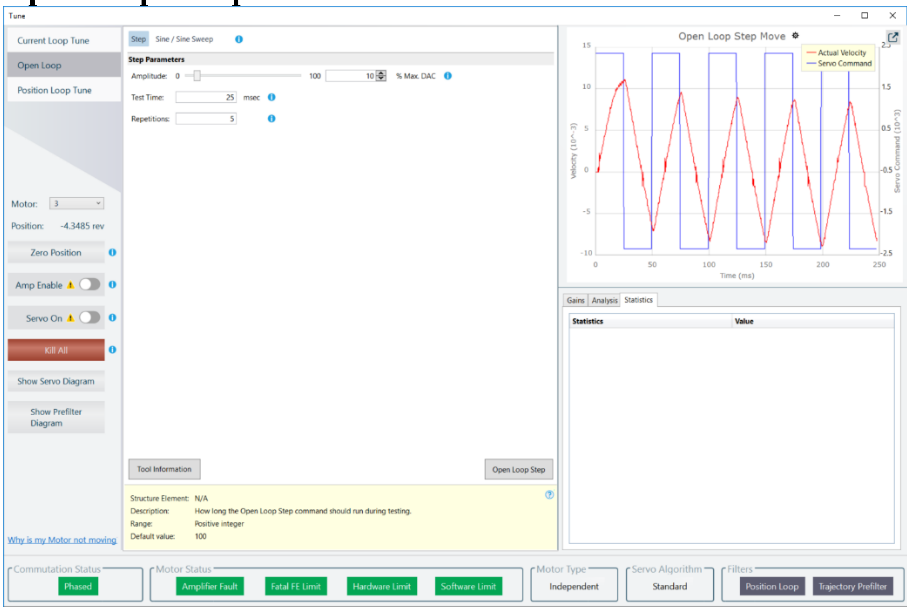
Note

Open Loop Moves

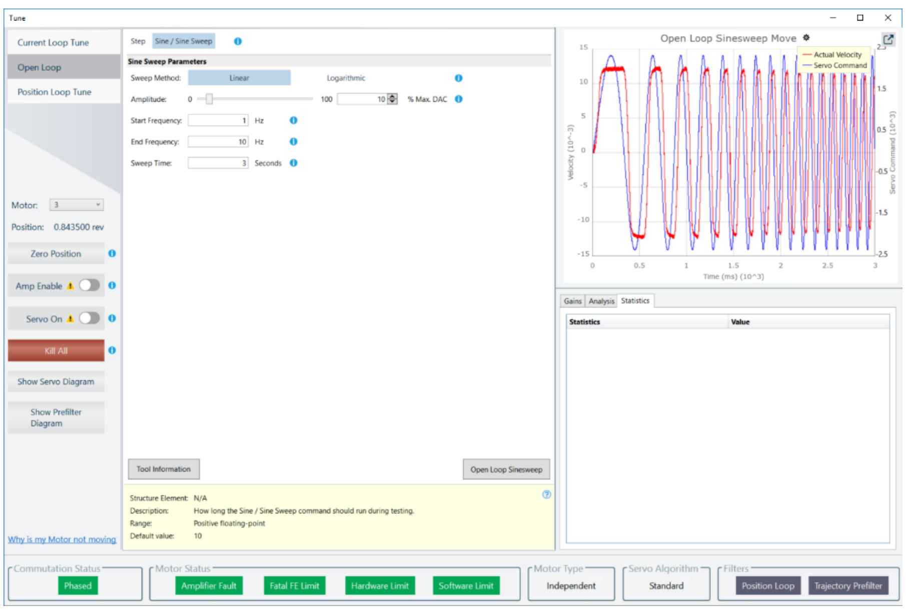
Please make sure that it is safe to do Tuning moves. Open loop Tuning moves, like Step or Sine/Sine Sweep moves can runaway in case of loss of feedback cable or communication

It is recommended external Emergency Stop switch connected that will kill the amplifier power in case of motor runaway or loss of communication.

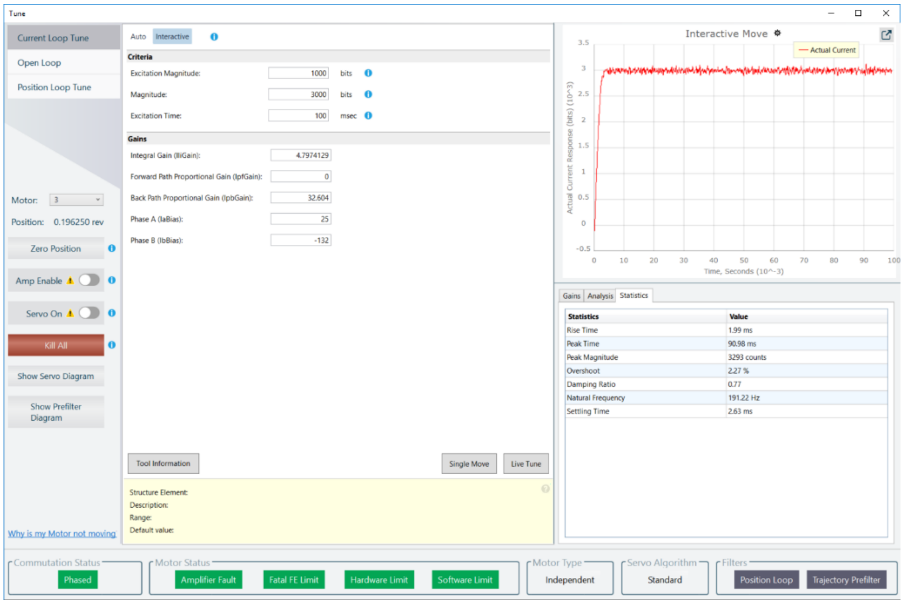
Open Loop – Step



Open Loop – Sine/SineSweep

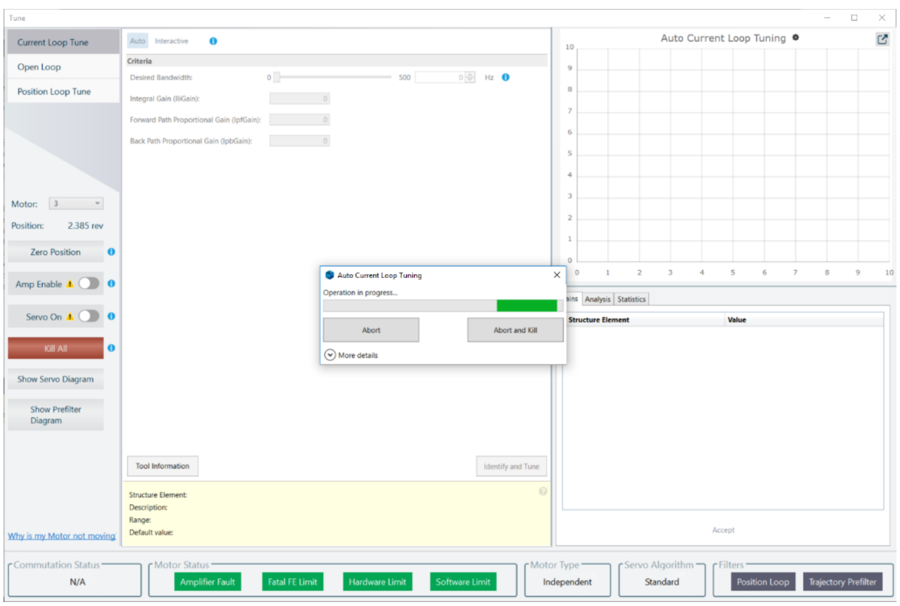


Current Loop Tune – Interactive

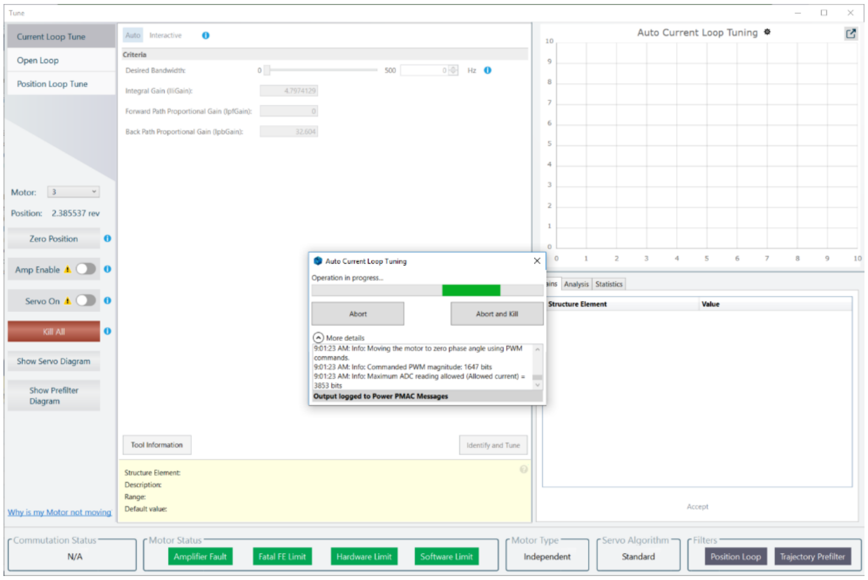


Current Loop Tune – Auto

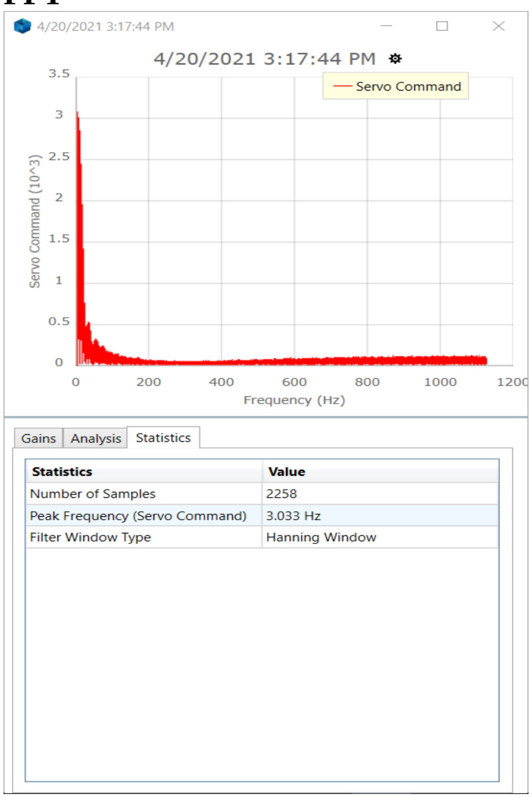
While in move progress bar is displayed.



Click on More details to see the messages coming out of the move. The messages are also logged in Power PMAC Message window.



FFT



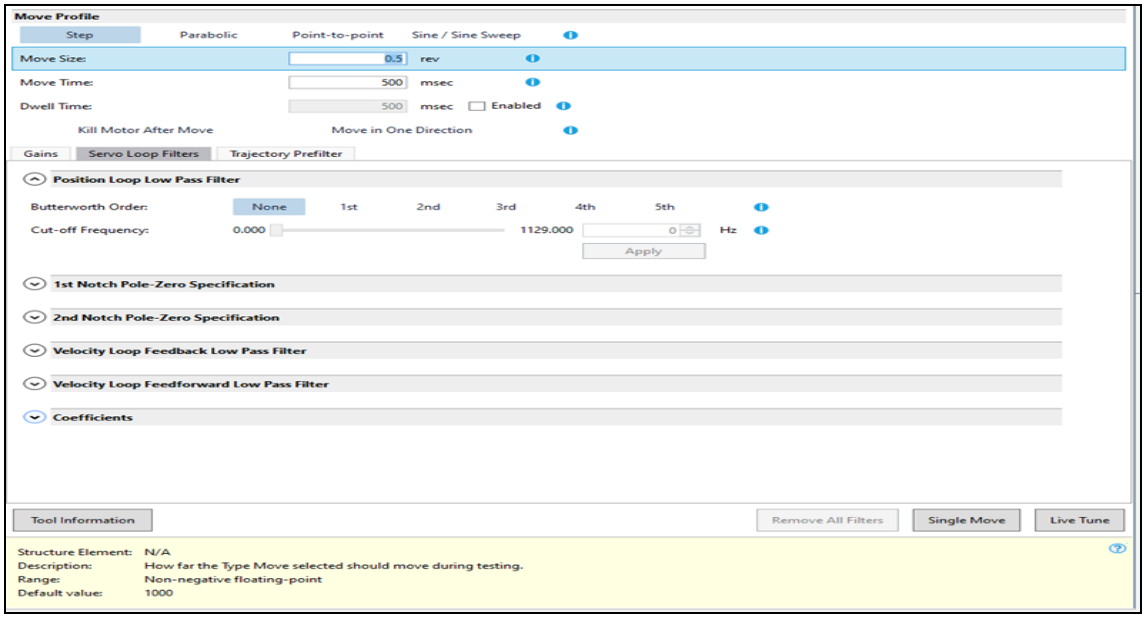
Filter options

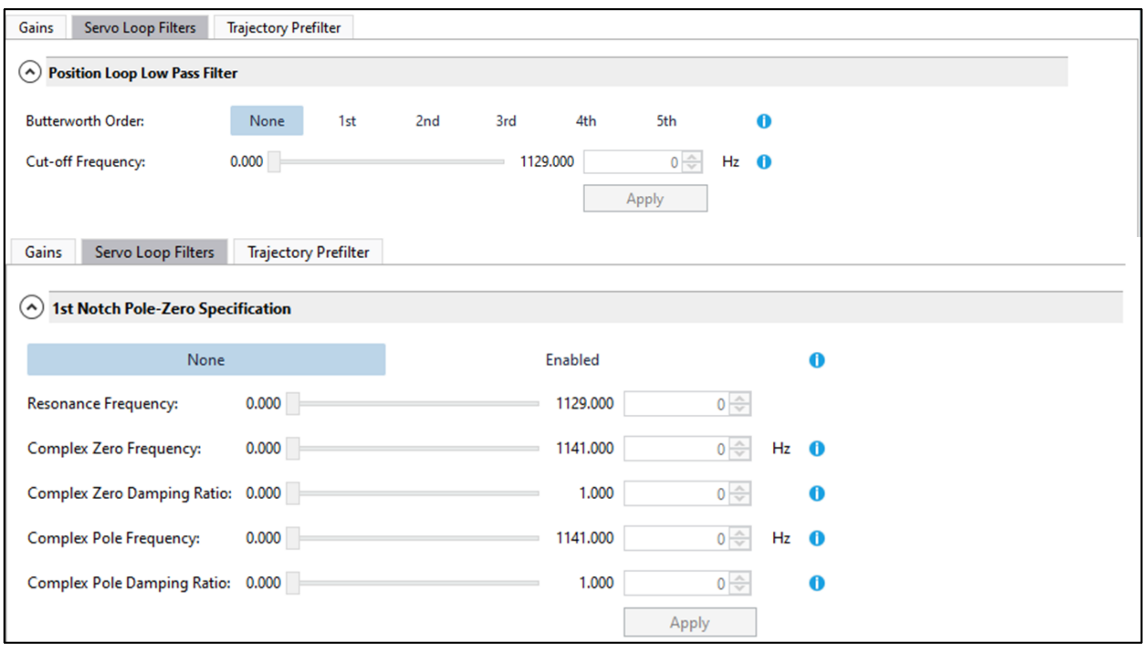
All the filter values are stored in the project, whenever project is loaded the previously set filter values are restore. Filter values are stored on the Power PMAC when the project is build and download so if the user upload the project from Power PMAC the filter values will be restored.

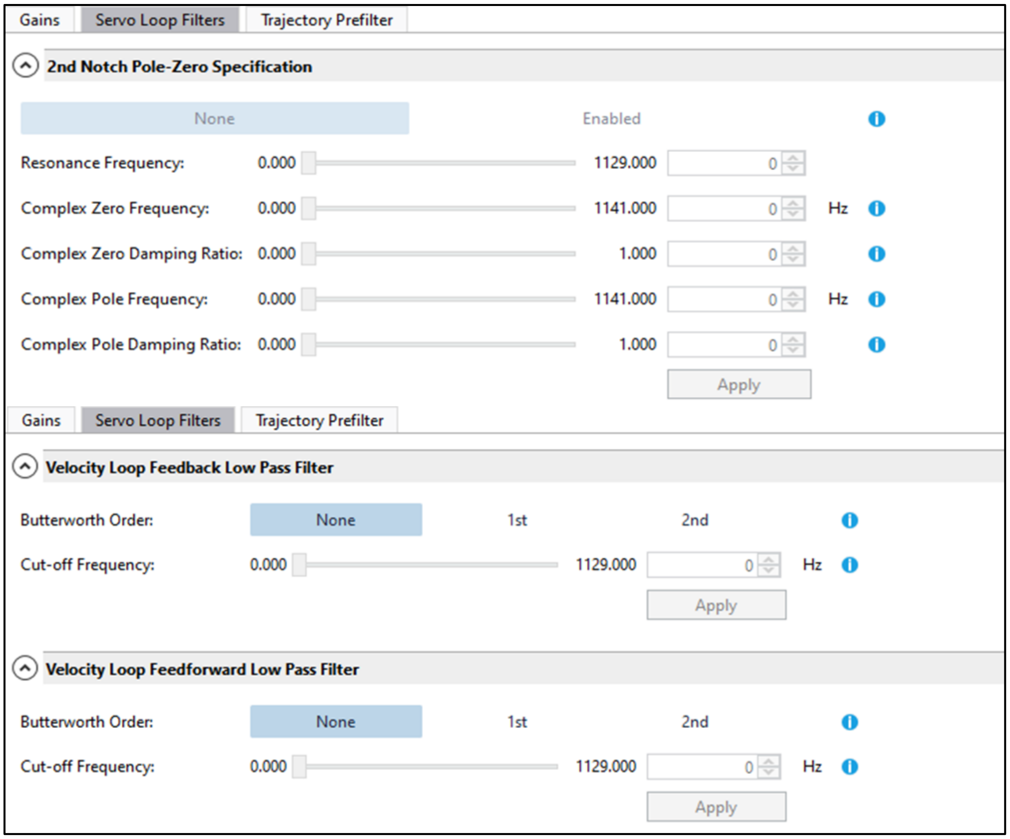
Two types of filter available with any type of interactive move, either single move or live tune.

Servo loop filter

Following are configuration screen for setting Servo Loop filter.


Move Profile
Step
Parabolic
Point-to-point
Sine / Sine Sweep
Move Size:
0.5
rev
Move Time:
500
msec
Dwell Time:
500
msec
Enabled
Kill Motor After Move
Move in One Direction
Gains
Servo Loop Filters
Trajectory Prefilter
Position Loop Low Pass Filter
Butterworth Order:
None
1st
2nd
3rd
4th
5th
Cut-off Frequency:
0.000
1129.000
0
Hz
Apply
1st Notch Pole-Zero Specification
2nd Notch Pole-Zero Specification
Velocity Loop Feedback Low Pass Filter
Velocity Loop Feedforward Low Pass Filter
Coefficients
Tool Information
Remove All Filters
Single Move
Live Tune
Structure Element: N/A
Description: How far the Type Move selected should move during testing.
Range: Non-negative floating-point
Default value: 1000

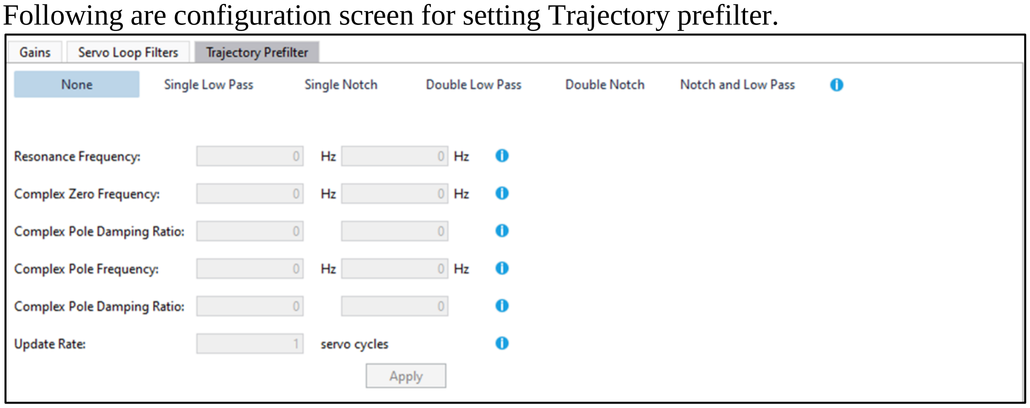

Gains
Servo Loop Filters
Trajectory Prefilter
Position Loop Low Pass Filter
Butterworth Order:
None
1st
2nd
3rd
4th
5th
Cut-off Frequency:
0.000
1129.000
0
Hz
Apply
Gains
Servo Loop Filters
Trajectory Prefilter
1st Notch Pole-Zero Specification
None
Enabled
Resonance Frequency:
0.000
1129.000
0
Complex Zero Frequency:
0.000
1141.000
0
Hz
Complex Zero Damping Ratio:
0.000
1.000
0
Complex Pole Frequency:
0.000
1141.000
0
Hz
Complex Pole Damping Ratio:
0.000
1.000
0
Apply



Trajectory Prefilter

The Trajectory Prefilter Setup is used to enable the Trajectory Prefilter feature of Power PMAC and configure whether to use it as a Notch Filter, a Low Pass Filter or both as there are two filters available which can be applied to the trajectories. The Trajectory Prefilter filters any trajectory that the Power PMAC generates before commanding it to the motor in order to prevent low frequency oscillations from occurring at the machine's end effector.

Following are configuration screen for setting Trajectory prefilter.



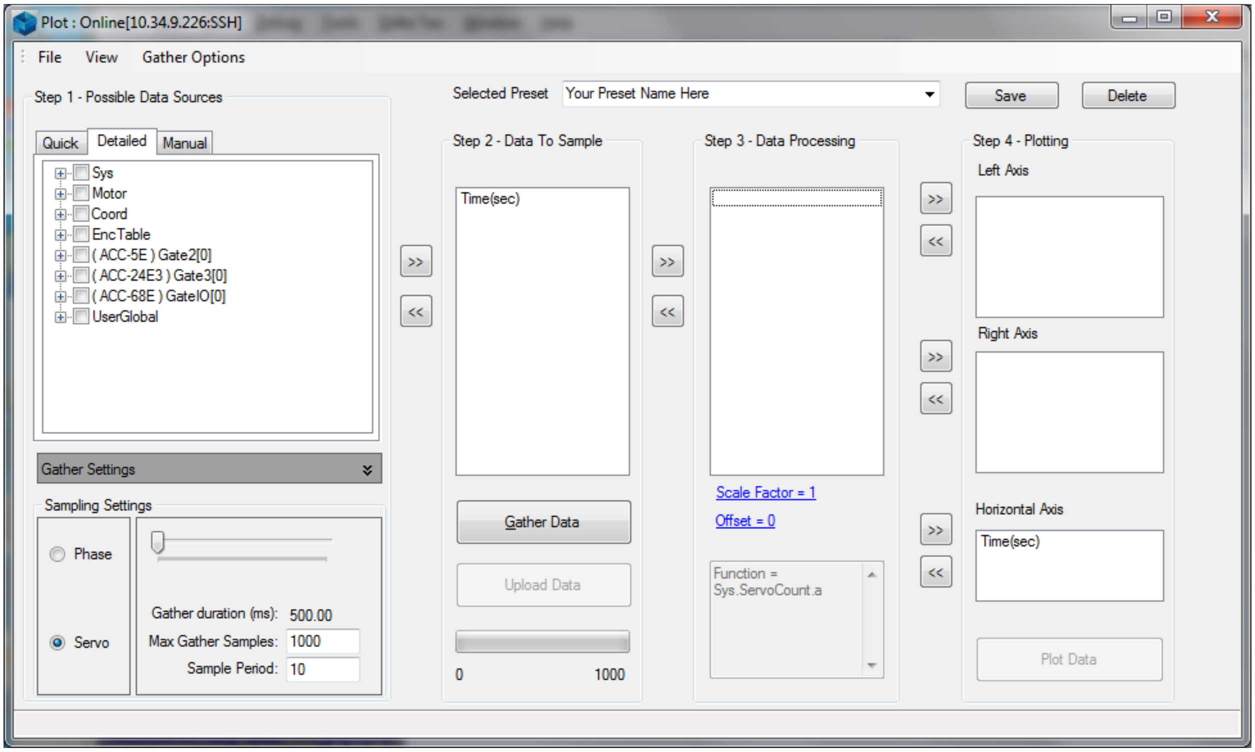
Plot

The Plot window can be used for gathering data from within Power PMAC and plotting it. This tool cannot be used for real-time plotting; for this case the Scope tool should be usedt. The Plot window can be configured through four steps:

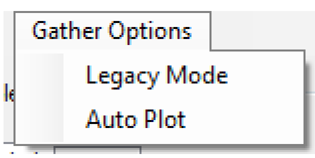
1. Possible Data Sources
2. Data to Sample
3. Data Processing

4. Plotting

These steps are outlined at the top of each pane in the Plot window as shown below:



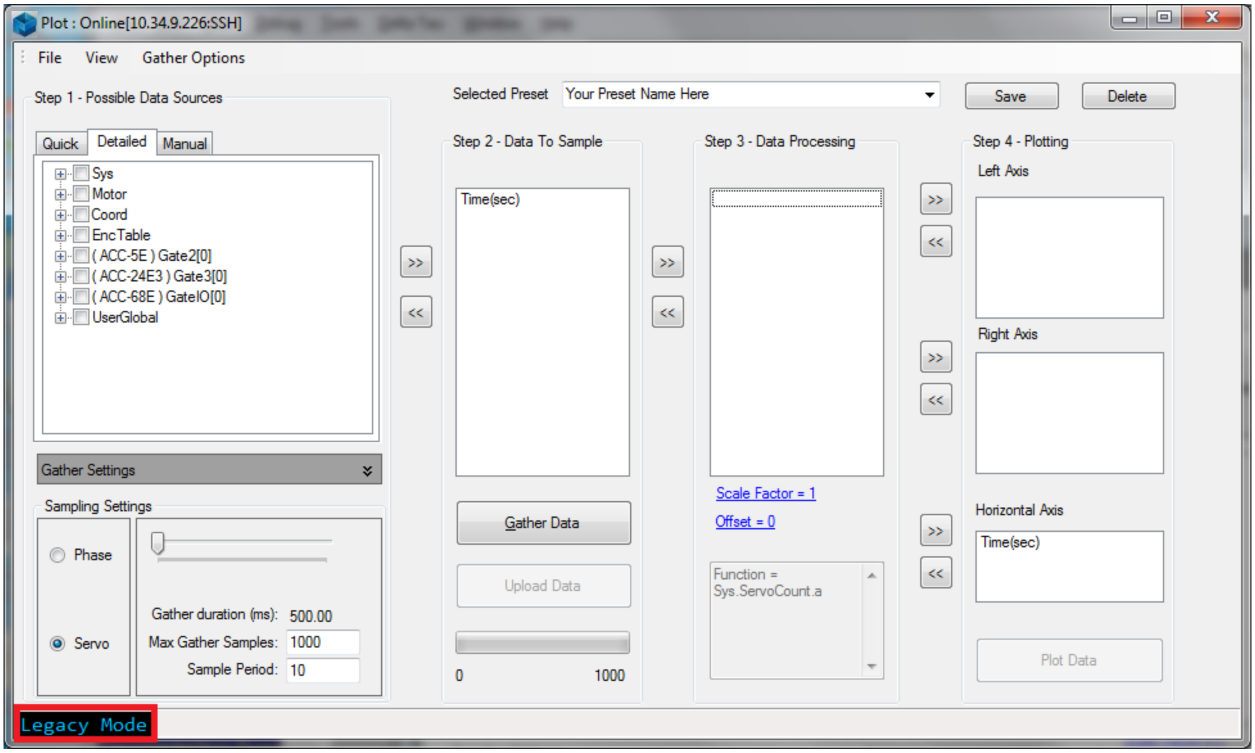
In the main plot window clicking on the Gather Options menu will list some options to change how a gather is performed.



Legacy Mode

Legacy Mode causes the Plot to gather data in the same method used prior to IDE version 2.1. Data begins to be stored in a buffer on Power PMAC when the Gather button is pressed until the Gather Max Samples is reached. When the Upload button is pressed, the data is stored in a file on Power PMAC and then transmitted to the user's PC and formatted for plotting.

Legacy Mode will be automatically enabled when the Plot control is connected to a device with firmware older than 2.0.2.64, or when the device is detected as being under heavy load. Devices detected as being under heavy load may have Legacy Mode disabled, but they are at increased risk of the gather being interrupted or data being lost. If either condition occurs the user will be notified. The screenshot below demonstrates the indication that legacy mode is enabled:



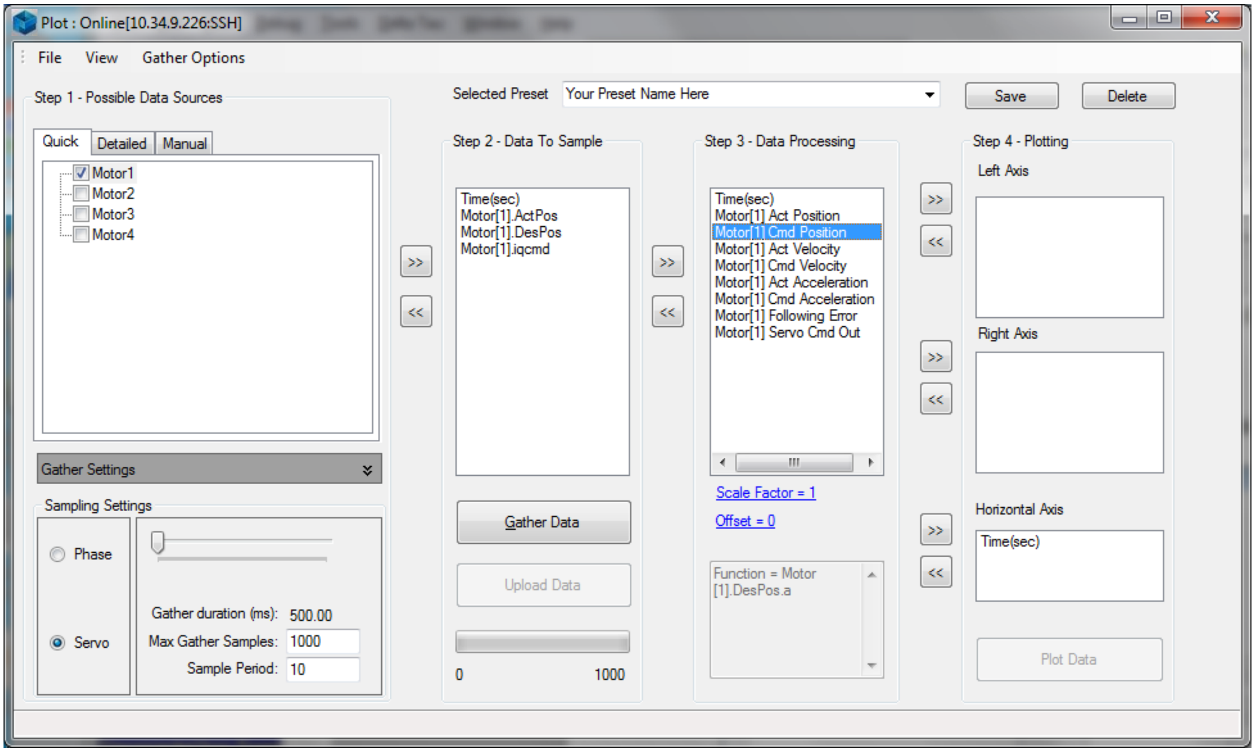
Auto Plot

Auto Plot saves the user from needing to press the Upload Data and Plot Data Buttons. When **Gather.Enable** changes from a value of 3 or 2 to 1 or 0 a plot is generated using the current settings in the Plot Control. Auto Plot may only be enabled while Legacy Mode is disabled.

Step 1 – Possible Data Sources
There are three tabs in the Plot Window underneath the heading "Step 1 – Possible Data Sources": Quick, Detailed, and Manual.

Quick Plot

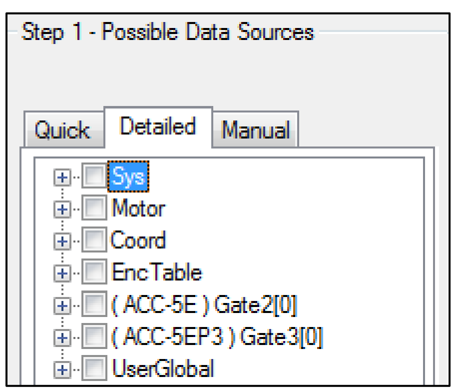
The Quick tab only displays motors that have been enabled (i.e. **Motor[x].ServoCtrl** > 0).



Selecting the enabled motor automatically puts commonly used motor structures into Step 2's and Step 3's panes.

Detailed Plot

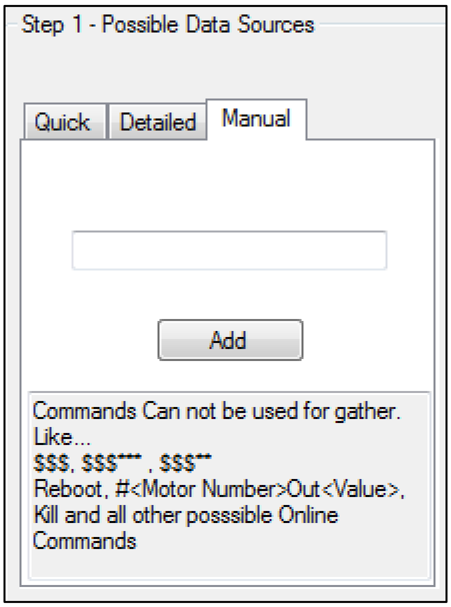
The Detailed tab shows all of the available structure trees whose structures can be plot. Click the plus button (⊞) next to each structure tree's name in order to display all of the elements or substructures within that tree:



Click the check box to the left of the structure or element to be included in Step 2 as a data source to sample.

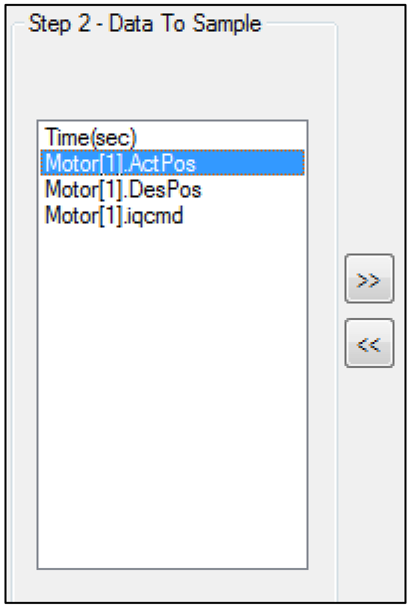
Manual Plot

Clicking the Manual tab allows the structure name to be entered if the exact structure name is known:



For example, **Motor[1].ActPos** could be entered to gather that structure directly.

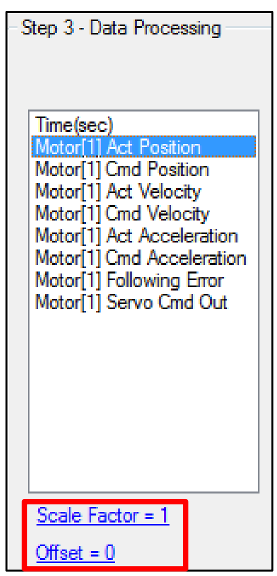
Step 2 – Data to Sample
Select the data source to be sampled by clicking the data source and then clicking the double right arrow (») button to add the data source to the Data Processing field for use in Step 3:



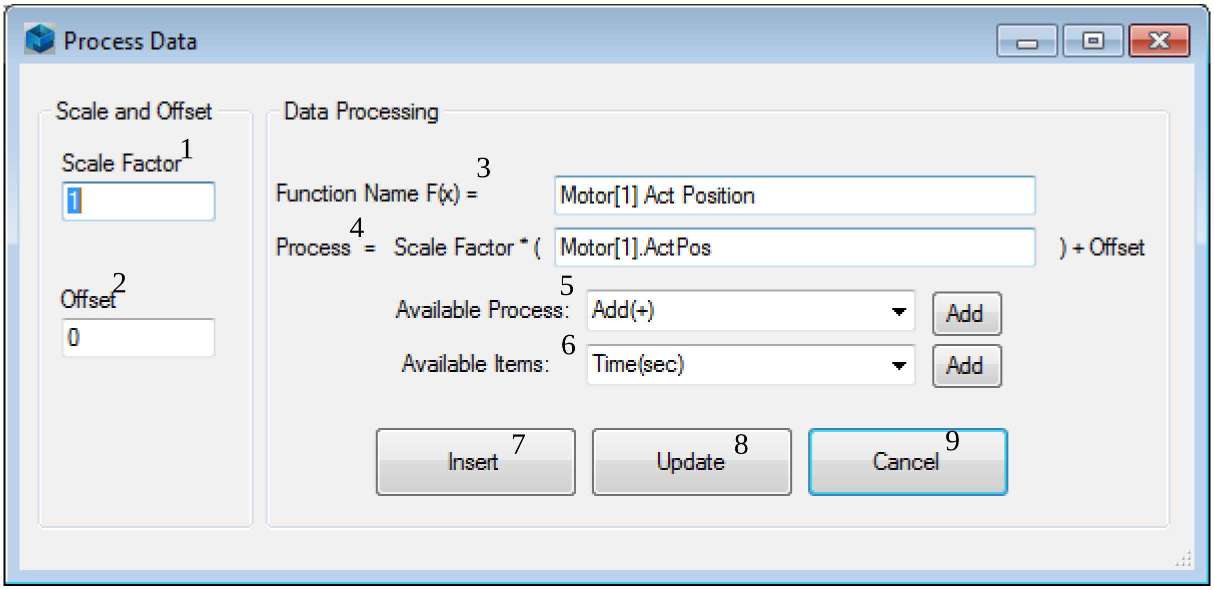
Clicking the double left arrow («) button will remove an item from Step 3 and put it back into Step 2.

Step 3 – Data Processing
To choose how to offset and scale the data multiply the raw data by a constant and/or add a constant to it before plotting the data.

All data is, by default, not scaled (i.e. it is multiplied by a scale factor of 1) and has an offset of 0. In order to modify the scale factor or the offset first select the data source required and then click “Scale Factor” or “Offset” as shown in the red box below:



This opens the “Process Data” window as seen below:



The various elements of the Process Data screen are described below. Each description corresponds to the superscripted red numbers superimposed in the screenshot above:

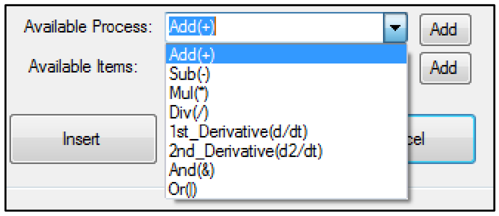
Scale Factor (**1**):
This number multiplies the data item as shown in the equation in Process (**4**).

Offset (**2**):
This number will be added to the product of the data item and the scale factor as shown in Process (**4**).

Function Name (**3**):
This is the name of the result of the data processing as listed in the box under “Step 3 – Data Processing” on the main Plot screen.

Process (**4**):
Process is the result/output of the data processing. In other words “Process” is equal to the expression shown to the right. By default the process is simply to multiply the data item selected by the Scale Factor (**1**) and then add the Offset (**2**).

Available Process (**5**):
This dropdown menu shows all of the processes which can be included in the Process (**4**) equation:



The functionality of each process in the list is described in the table below:

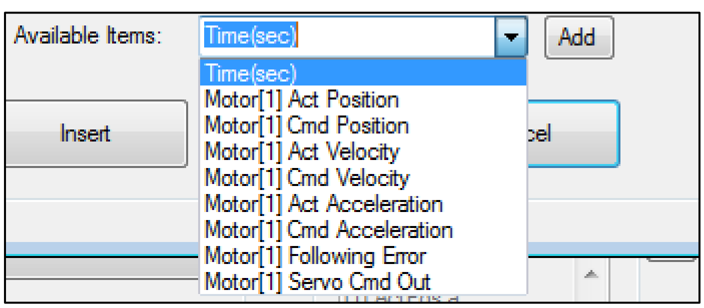
Process Symbol	**Functionality**
Add(+)	Adds a number[1] specifed in the Process equation to the Data Item
Sub(-)	Subtracts a number[1] specifed in the Process equation from the Data Item
Mul(*)	Multiplies a number[1] specified in the Process equation by the Data Item
Div(/)	Divides the Data Item by a number[1] specified in the Process equation
1st_Derivative(d/dt)	Performs the numerical 1st derivative of the Data Item
2nd_Derivative(d^2/dt^2)	Performs the numerical 2nd derivative of the Data Item
And(&)	Performs a bitwise AND with the data and a number[1] specified in the Process equation
Or(|)	Performs a bitwise OR with the data and a number[1] specified in the Process equation
[1]This number can either be a hard-coded constant or another Data Item selected in Available Items (**6**).	

After selecting the process to use click the 

button to the right of the dropdown menu to add that process to the Process (**4**) equation.

Available Items (**6**):
Other Data Sources can be incorporated into the Process (**4**) equation. To do this click the "Available Items" dropdown menu to select the Data Source required to be added to the Process equation and then click the 

button:



Insert (**7**):
Click this button to insert the data processing entry into Step 3's list of items back on the main Plot screen.

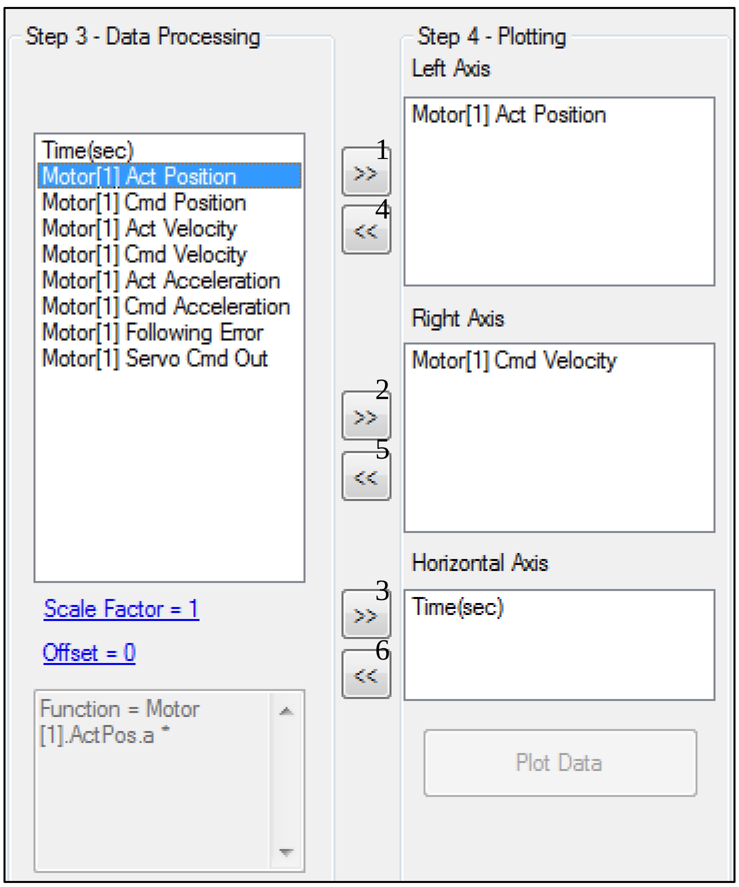
Update (**8**):
Click this button to update this entry, if it already exists, with the settings selected.

Cancel (9):
Click this button to cancel modifying this entry.

Step 4 – Plotting
In Step 4 the items to be plot can be confiured to specific Axis. The axes available are the Left Axis, the Right Axis and the Horizontal Axis.

To add an item to an axis select the Data Source required to be added from Step 3's list of items and then click the double right arrow button (») next to the list box:

- Click the arrow (») indicated by the superscripted red "1" shown in the image below to add the data source to the Left Axis.
- Click the arrow (») indicated by the superscripted red "2" shown in the image below to add the data source to the Right Axis.
- Click the arrow (») indicated by the superscripted red "3" shown in the image below to add the data source to the Horizontal Axis.



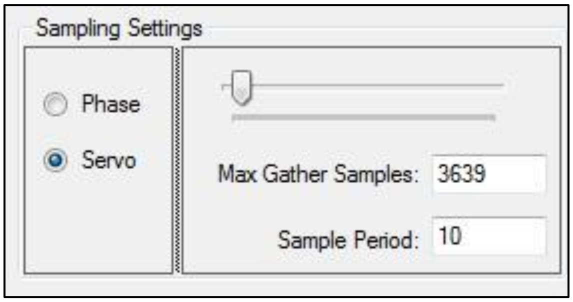
Click the double left arrow button («) next to each axis's list box to remove that item from the axis:

- Click the arrow («) indicated by the superscripted red "**4**" shown in the image above to remove the data source from the Left Axis.
- Click the arrow («) indicated by the superscripted red "**5**" shown in the image above to remove the data source from the Right Axis.
- Click the arrow («) indicated by the superscripted red "**6**" shown in the image above to remove the data source from the Horizontal Axis.

Gathering and Plotting
The final step is to gather, upload and plot the data.

Sampling Settings

The sampling settings controls in Step 1 sets how many samples are gathered per data source and the sampling period for gathering:



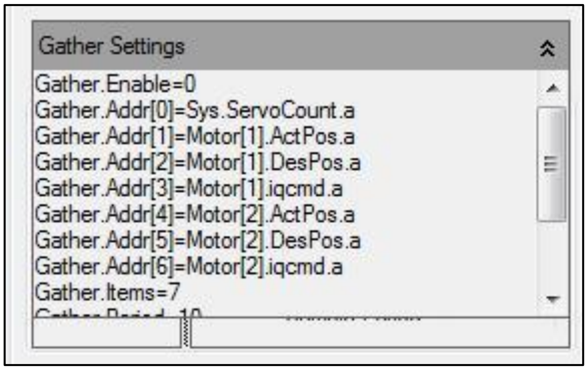
The "Sample Period" is in units of servo periods. For example, if the sample period is set to equal to 1 then this will sample every servo period.

"Max Gather Samples" specifies the maximum number of data points to sample per source.

Selecting the slider underneath "Sampling Settings" will show how many seconds of data will be gathered based upon the "Max Gather Samples" and the "Sample Period" settings chosen.

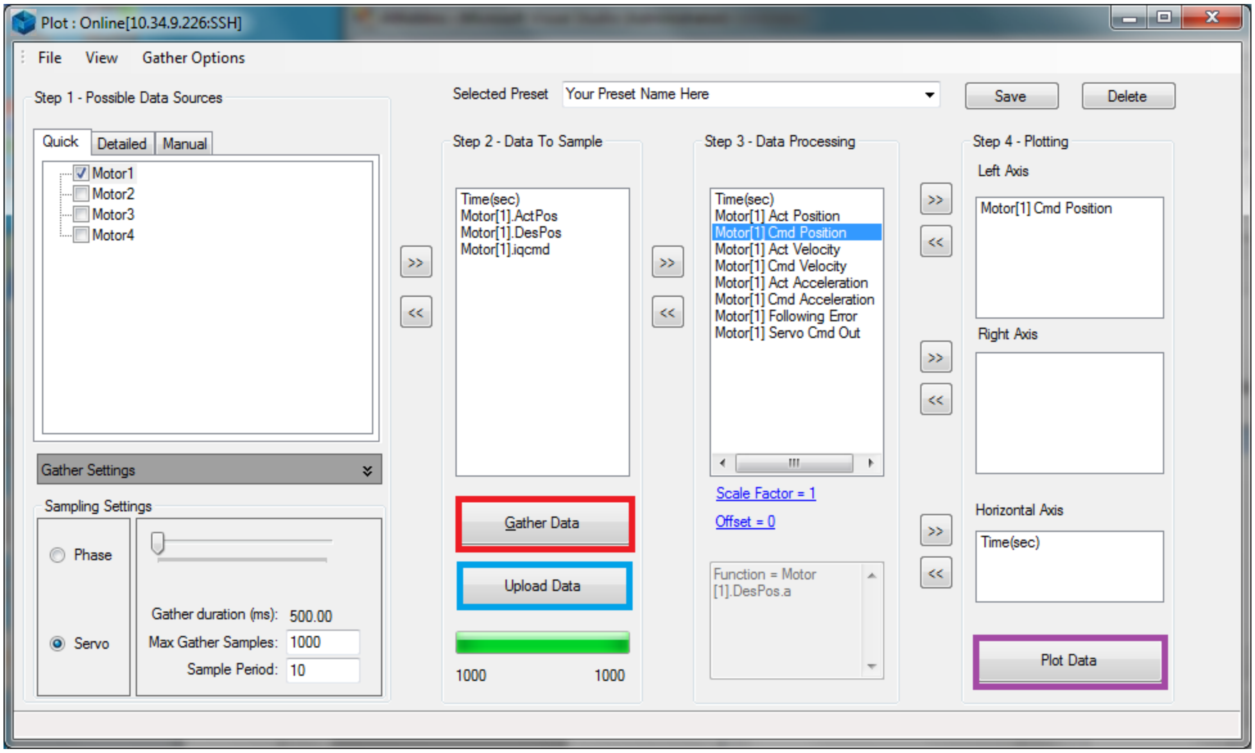
The plot program supports gathering at the Phase rate as well. The settings are similar to Servo rate sampling settings.

The Gather Settings window in the lower left corner of the Plot window (shown on the right) shows settings describing the sources to gather, whether to use servo period or phase period, etc., that will be used for gathering.



Gathering

To start gathering the data click the "Gather Data" button as shown in the red box below:

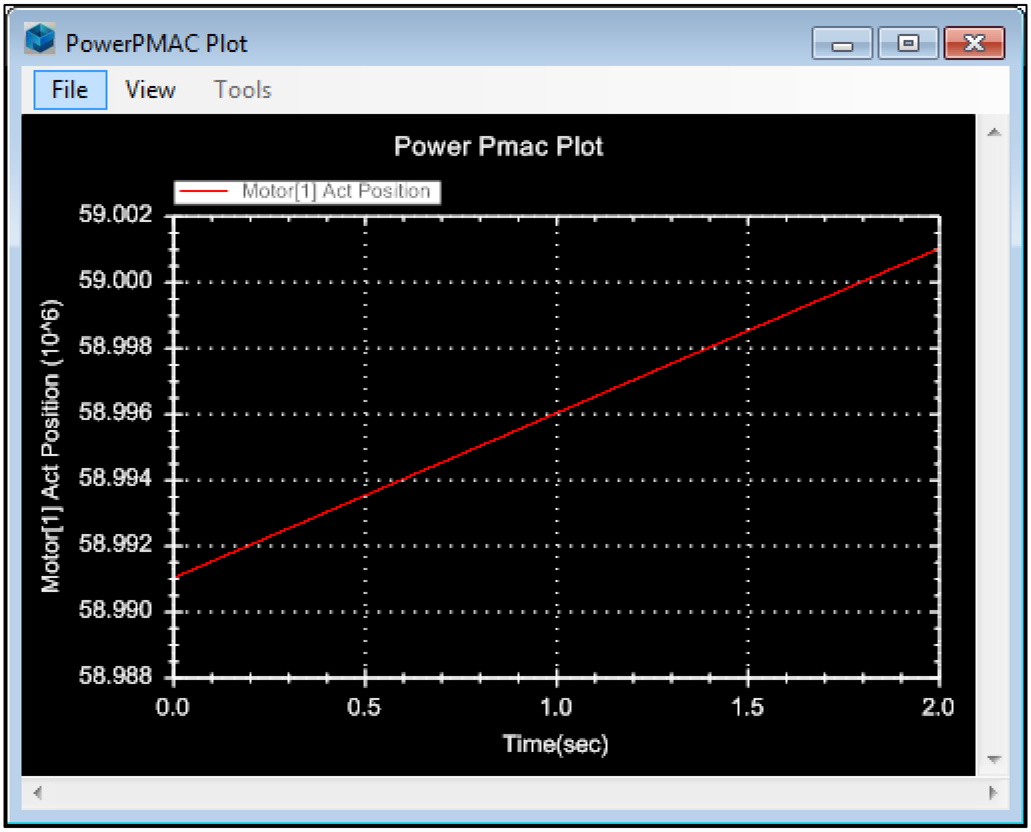


The progress meter, which is located beneath the "Upload Data" button (surrounded by a blue box in the image above), will fill up with green as the data is being gathered. Once the meter is full with green click "Upload Data". The process can be stopped while gathering data by clicking the "Stop" button - this replaces the "Gather Data" (surrounded by a red box in the image above) while data is being gathered.

After clicking "Upload Data" and the data has been uploaded click "Plot Data" (surrounded by a purple box in the image above). This button will be grayed out until the data has been successfully uploaded.

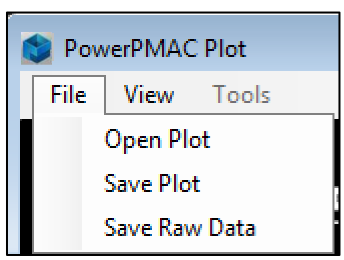
Plot Tools

Clicking the "Plot Data" button opens a plot for the selected data sources. In this example the actual position of motor 1 is being plotted on the Left Axis as a function of time on the horizontal axis:



Tools for Saving and Exporting Plots and Raw Data

Clicking the File menu shows tools for loading and saving plots:



Open Plot

Opens a plot saved previously with the "Save Plot" command.

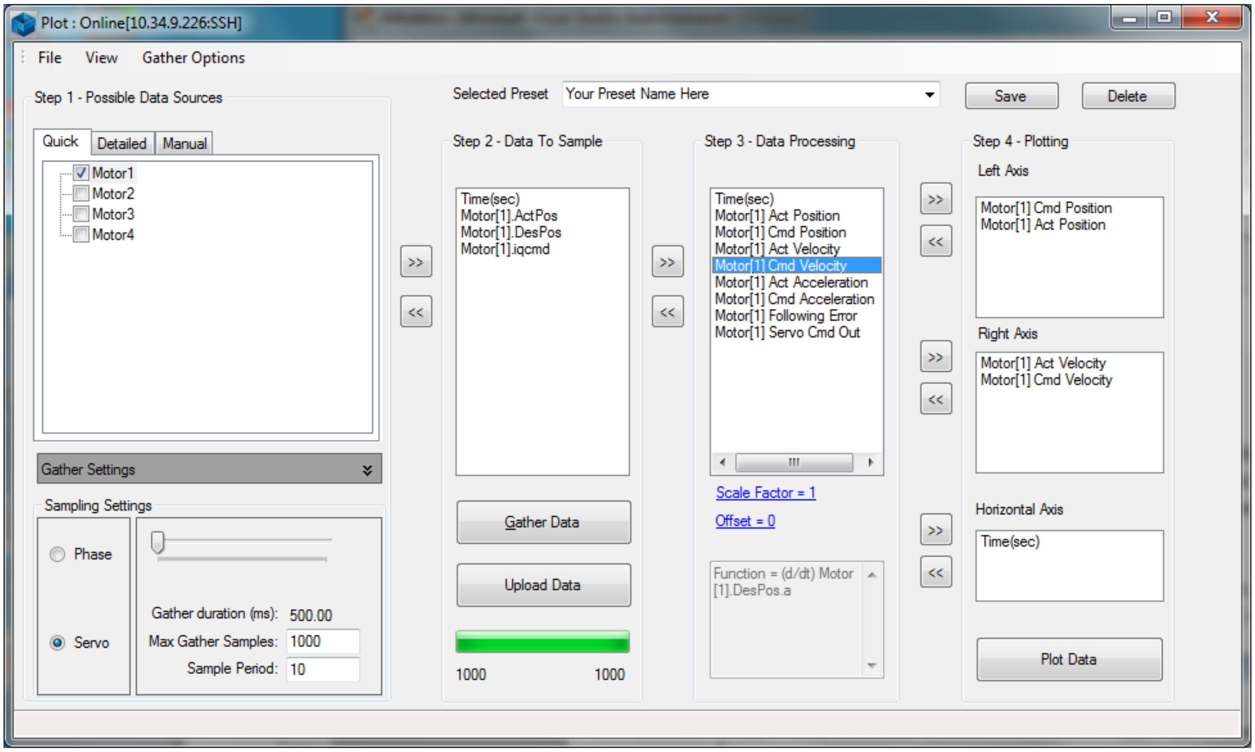
Save Plot

Saves the contents of the current plot and the plot formatting settings in the "*.ppp" file format.

Save Raw Data

Saves the raw data contents of the plot without the plot formatting settings. This file is in the "*.txt" file format. This file consists of tab-delimited columns. The first row is the name of the data source. Subsequent rows contain the data points in double precision. The leftmost column is the first data source for the Horizontal Axis selected. The next column is the next data source for the Horizontal Axis. After that, subsequent columns consist of the Left Axis data sources in order and then the Right Axis data sources.

For example, motor 1's actual and commanded position are on the Left Axis, the actual and commanded velocity on the Right Axis, and Time on the Horizontal Axis as shown below:



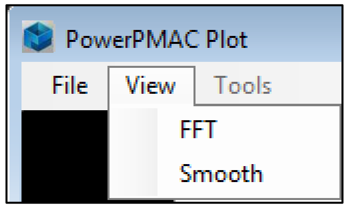
Then the exported data file will appear as such (only the first few rows are being shown):

```
Time(sec)  Motor[1] Act Position  Motor[1] Cmd Position  Motor[1] Act Velocity  Motor[1] Cmd Velocity
0.000000 73153818.000000 73153818.000000 5000.000000 5000.000000
0.002000 73153828.000000 73153828.000000 5000.000000 5000.000000
0.004000 73153838.000000 73153838.000000 5000.000000 5000.000000
0.006000 73153848.000000 73153848.000000 5000.000000 5000.000000
0.008000 73153858.000000 73153858.000000 5000.000000 5000.000000
0.010000 73153868.000000 73153868.000000 5000.000000 5000.000000
0.012000 73153878.000000 73153878.000000 5000.000000 5000.000000
0.014000 73153888.000000 73153888.000000 5000.000000 5000.000000
```

This can easily be imported into, for example, Microsoft Excel™ for further processing if desired.

Tools for Filtering Data and Creating Power Spectra

Clicking the View menu will show some tools for filtering the data and plotting power spectra:



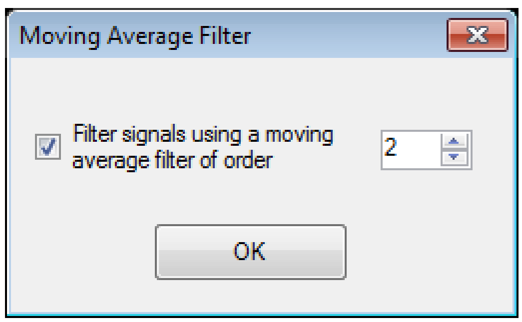
These tools work as follows:

FFT

This tool will perform a Fast Fourier Transform (FFT) of the data. It is possible to choose whether to filter the signals or not or whether to plot the vertical axes in units of decibels (dB). Choosing to filter the data will perform a Hanning window filter on the data. If not, it will use a Uniform/Rectangular window. The Horizontal Axis will not be logarithmic.

Smooth

This tool will filter the chosen plot signals with a moving average whose order can be set from 0 to 10:

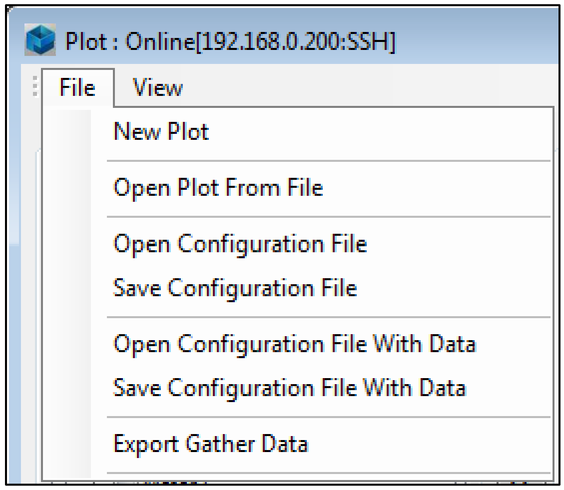


The default filter order is 2. The filter sums groups of points (the number of points in the sum is equal to the order of the filter) and then divides by the order of the filter. The equation of the filter is as follows:

$$p_i = \frac{1}{N}\sum_{k=0}^{N} a_k,$$

where p_i is a point on the plot, where i runs from 0 to the total number of points on the plot, N is the order of the moving average filter, and a_k is a point of data in the group of points of size $(N + 1)$ presently being processed by the filter.

Saving and Loading Plot Configurations
In the main Plot window clicking on the File menu will list several tools:



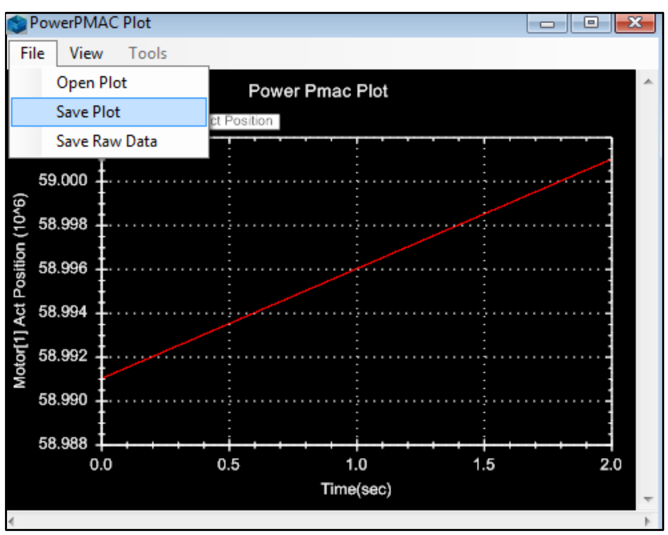
These tools work as follows:

New Plot
This tool wipes this Plot window of all settings and shows a default, blank Plot window.

Note

The Plot Window will retain the plot settings chosen for this project until a New Plot is clicked to wipe it clean.

Open Plot From File
This tool opens a plot previously saved by clicking File→Save Plot from within a plot of data as in the screenshot below:



Open Configuration File
Opens a configuration file containing the plot settings previously saved by clicking "Save Configuration File" in this Plot Window.

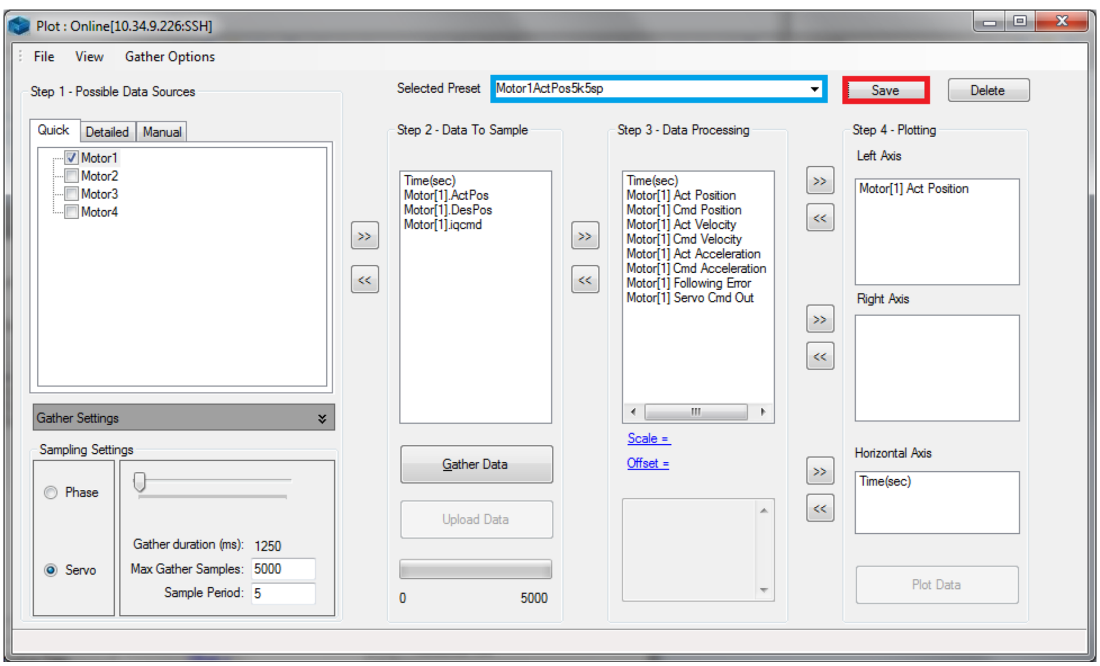
Save Configuration File
Saves a configuration file containing the plot settings for this present instance of the Plot Window in the "*.cfg" file format.

Open Configuration File with Data
Opens a configuration file containing the plot settings previously saved, along with the data previously saved by clicking "Save Configuration File with Data."

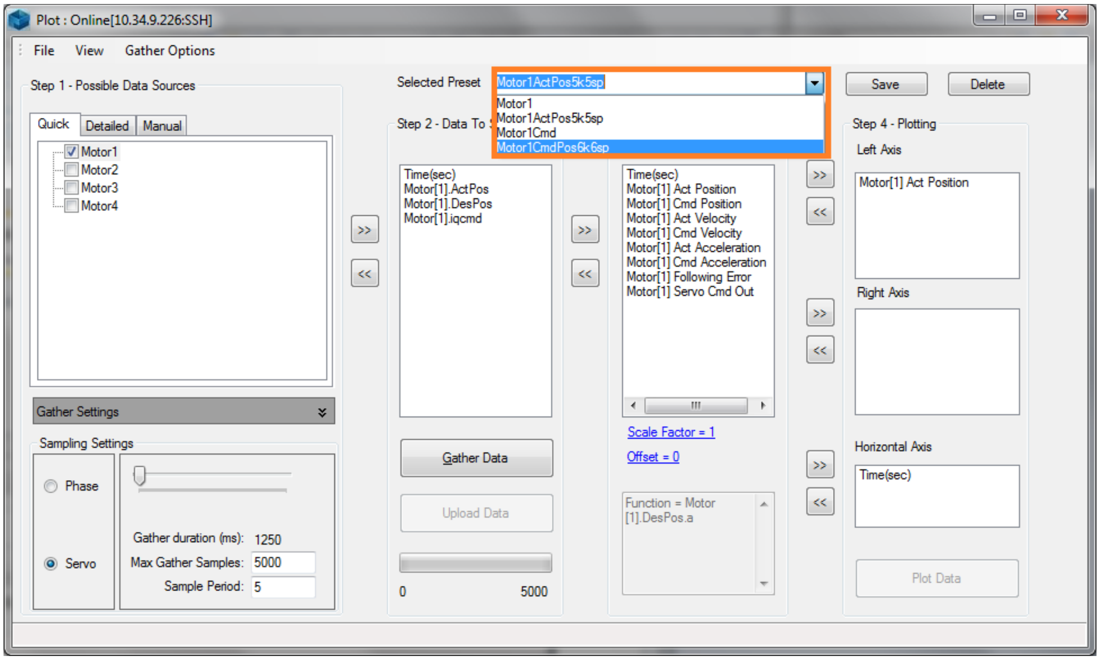
Save Configuration File with Data
Saves a configuration file containing the plot settings for this present instance of the Plot Window along with any data presently uploaded to the PC from Power PMAC in the "*.prj" file format.

Export Gather Data
Exports any data presently uploaded to the PC from the Power PMAC in the "*.gat" file format.

Selected Presets
Selected Presets allow for rapidly switching the gathered and plotted items to previously saved selections. Once the plot contains the setup to be saved, type a name for the setup in the "Selected Preset" field (boxed in blue in the image) and press the "Save" button (boxed in red):



To switch to a different preset, select the item in the “Selected Preset” field’s dropdown menu:

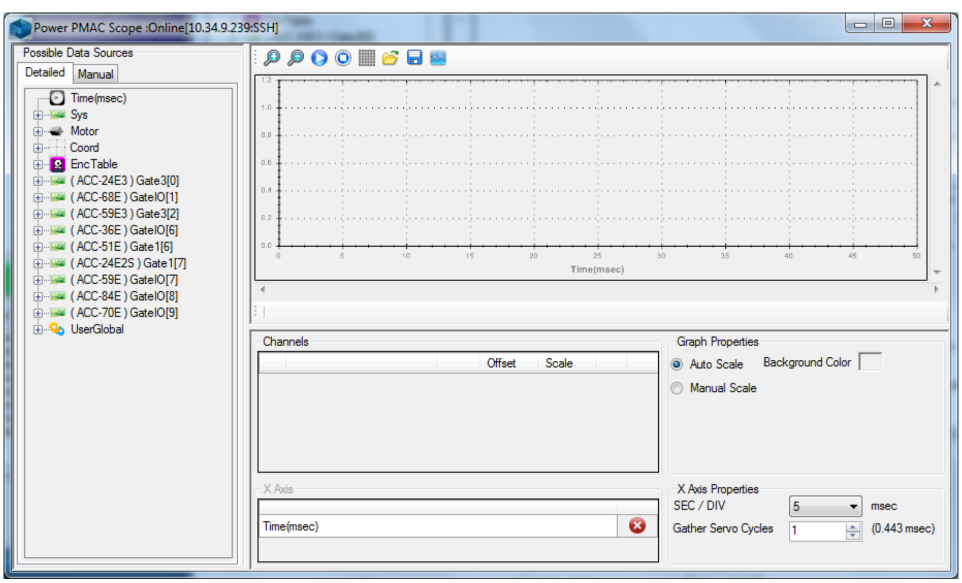


Alternatively pressing the Enter key, while the cursor is in the “Selected Preset” field and the field contains the name of a previously saved preset, will load that preset. In order to delete a preset type its name in the selected preset field or select it from the dropdown and press the delete button. To overwrite an existing preset with different options select it from the dropdown menu or type its name in the Selected Preset field, change the Sampling, Gather, Processing, and/or Plot options, and press the Save button.

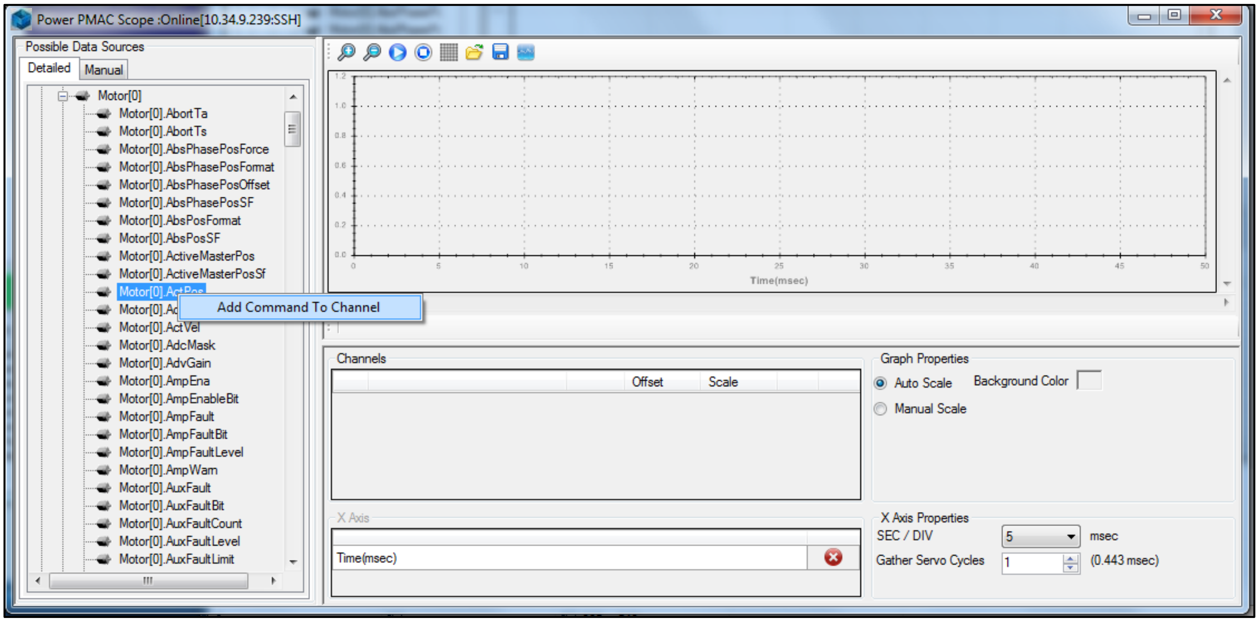
Scope

The scope tool enable the plotting of data in near real-time. The interface looks like this:

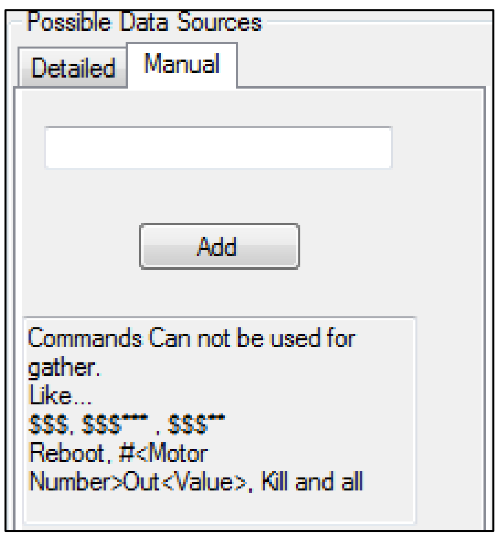
Selecting the Data to Scope



Under the “Detailed” tab on the left select the structure to scope. Click the plus button (⊞) to expand the structure tree, right-click the structure and then click “Add Command to Channel” as shown below:



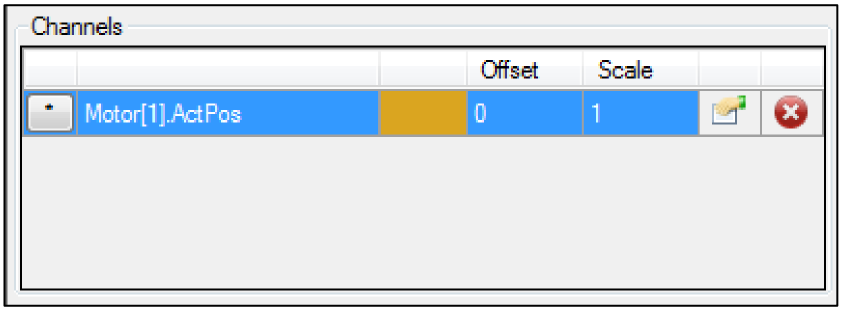
Alternatively click the “Manual” tab and type in the command to add to the channel:



The exact structure name must be entered. After typing the command click “Add” to add it to the channel.

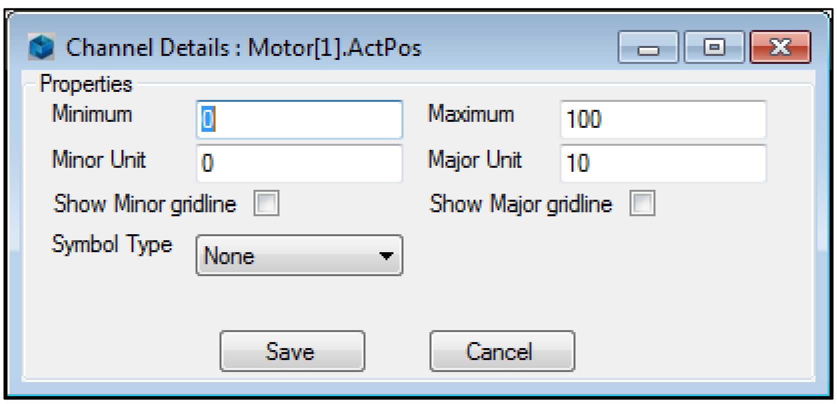
Changing Vertical Axis Settings

The offset, to add to the data, and thescale factor, by which to multiply the data, can be modified before plotting by changing the Offset and the Scale fields, respectively, in the box underneath "Channels" as shown below:



To delete the command from the channel click the Delete button ( ). To select this channel as the primary vertical axis click the button. To change the properties of this channel click the Properties button ( ).

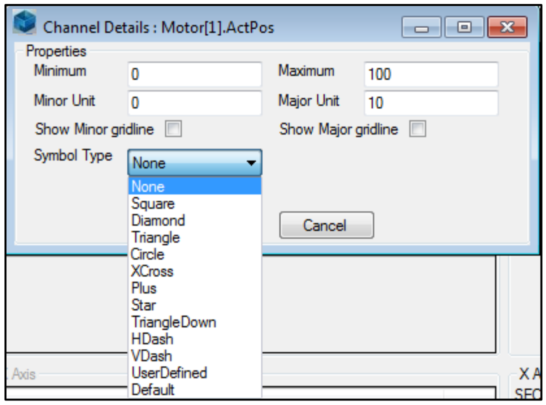
Clicking the Properties button will show the Channel Details dialog box as shown below:



The minimum and maximum limits and the minor and major units for the vertical axis which this command occupies can be set here. Note that these scales will only be used if "Manual Scale" is selected under Graph Properties on the main Scope window.

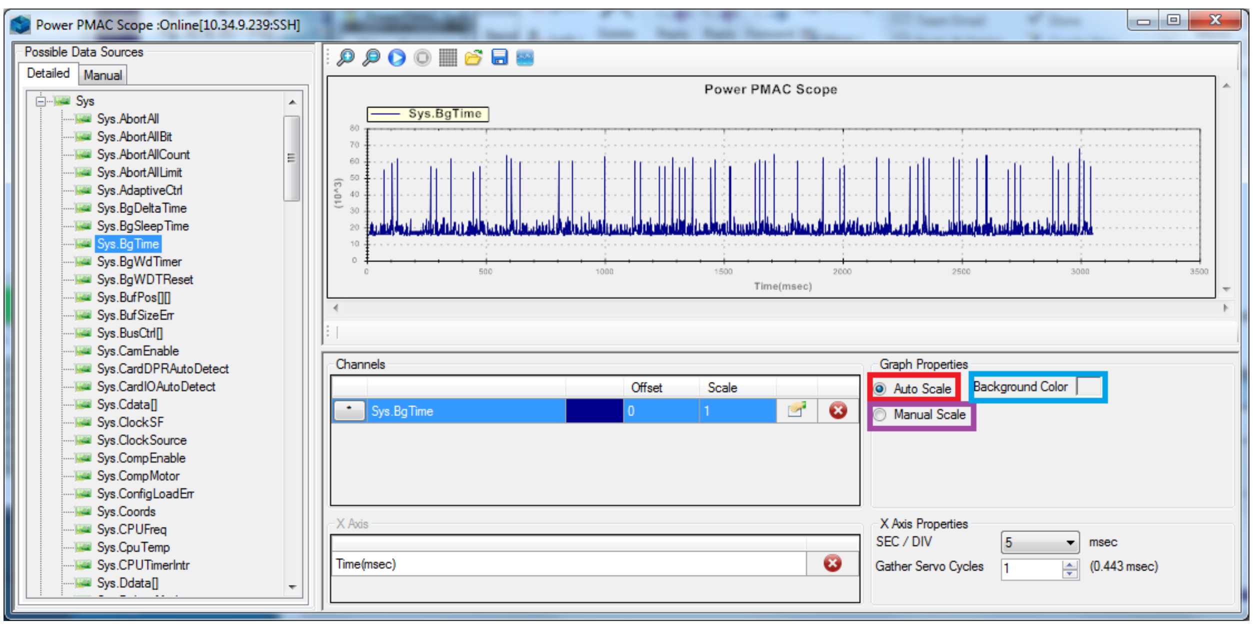
To show or hide the minor and major gridlines select the Check boxes.

The symbol type to represent each data point can be selected. The symbol types that can be chosen are shown below:



Click the “Save” button to save the settings and leave the Channel Details window.

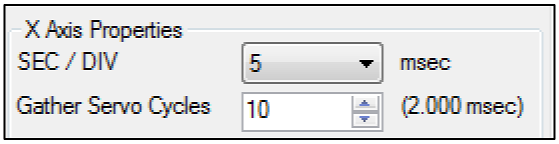
On the the main Scope window choose whether to have the IDE automatically scale the Scope window’s limits based upon the size of the data by clicking on “Auto Scale” (surrounded by a red box in the image below) under Graph Properties:



To choose Manual scaling instead select “Manual Scale” (surrounded by a purple box in the image above) and set the limits in the Properties menu which is opened by clicking on the  button for the channel whose limits are to be modified. To change the color of the Scope’s background click on the Background Color button (surrounded by a blue box in the image above).

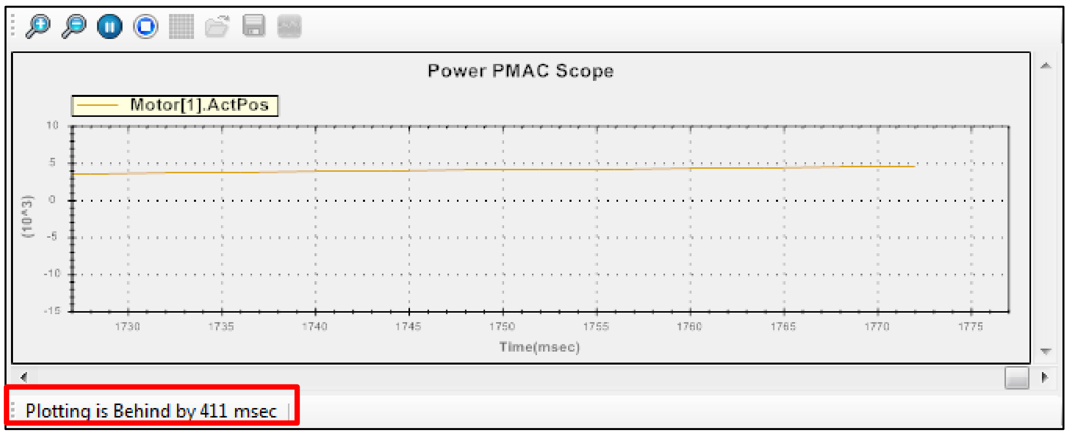
Changing Horizontal Axis Settings

The horizontal axis Time is in units of milliseconds. The horizontal axis’s properties are listed in the bottom right corner of the Scope window:



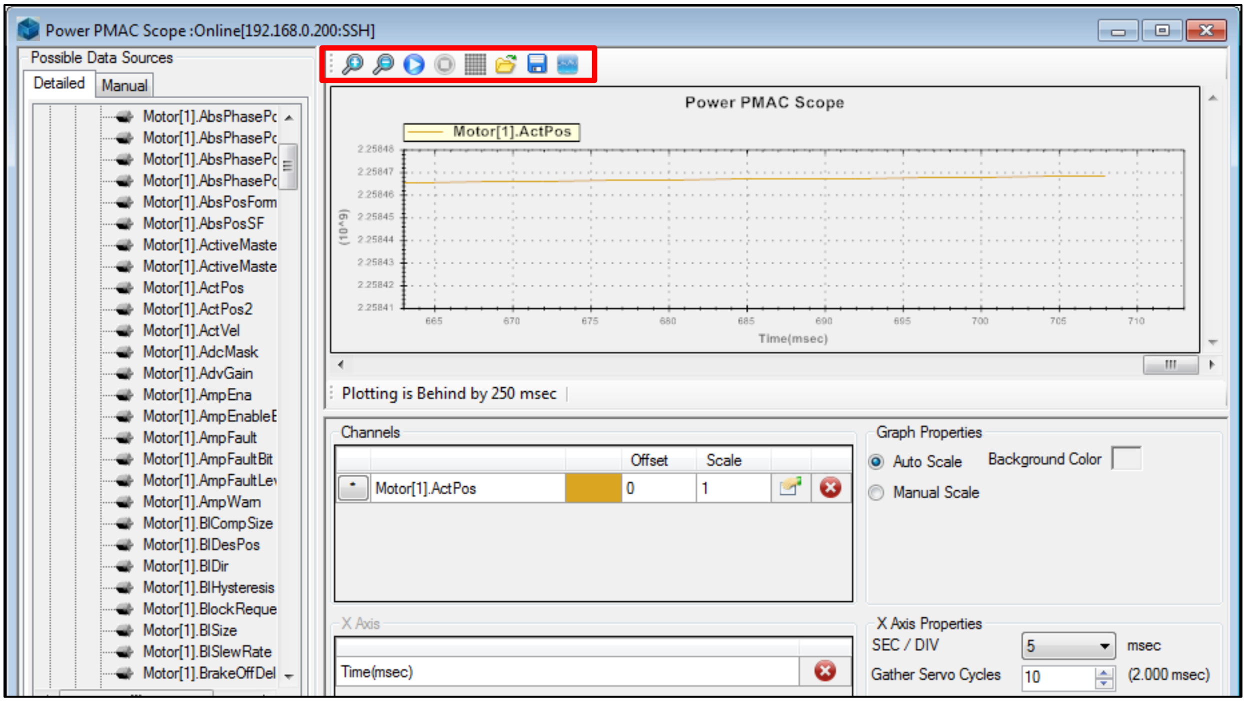
The seconds per division can be changed by clicking on the SEC / DIV button. The select divisions available are 2, 5, 10, 50, 100, 200, 500, or 1000 msec.

The plot period can be specified by typing a value, in units of servo cycles, to the right of "Gather Servo Cycles". The Scope window will calculate the number of msec which the number of servo cycles chosen occupies. The speed at which gathered points are plotted is calculated automatically. It appears next to "Plotting is behind by x" below the live scope as "Filter = x" which indicates that 1 of every x points is plotted. All data up to the allowed buffer size is retained for "Graph all Data points" even if it is not plotted live. Information on how far behind the plot is can be seen on the bottom of the plotting area as shown by the red box in the image below:

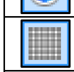
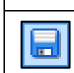


Scope Controls

A number of tools are available for manipulating the Scope plot area as highlighted in the red box in the image below:



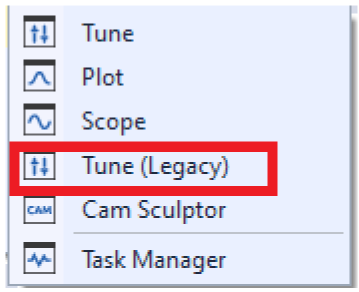
The table below describes the functionality of each of the buttons:

Scope Control Symbol	Tooltip Description	Functionality
	Zoom In	Makes the plot area occupy the entire Scope window
	Zoom Out	Returns the plot area to the upper right corner of the Scope window
	Start	Starts gathering and plotting data
	Stop	Stops gathering and plotting data
	Clear Graph	Clears the plot area of the Scope window
	Open Plot from a File	Opens a Power PMAC Realtime Plot (*.csv) file (see next row down)
	Save Plot to a File	Saves the plot’s contents to a Power PMAC Realtime Plot (*.csv) file
	Plot All Gather Points	Plots everything gathered so far (everything presently in the gather buffer) in the plot area of the Scope window.

Tune (Legacy)

The Tuning tool can be used to tune current loops and position (servo) loops the motors. “Tuning” refers to the process of adjusting the gains in the control loop until the desired performance level is achieved. The Tune tool can be used to configure filters for the position and velocity loops and also trajectory prefilters.

Access the Tuning by clicking Delta Tau→Tools→Tune:

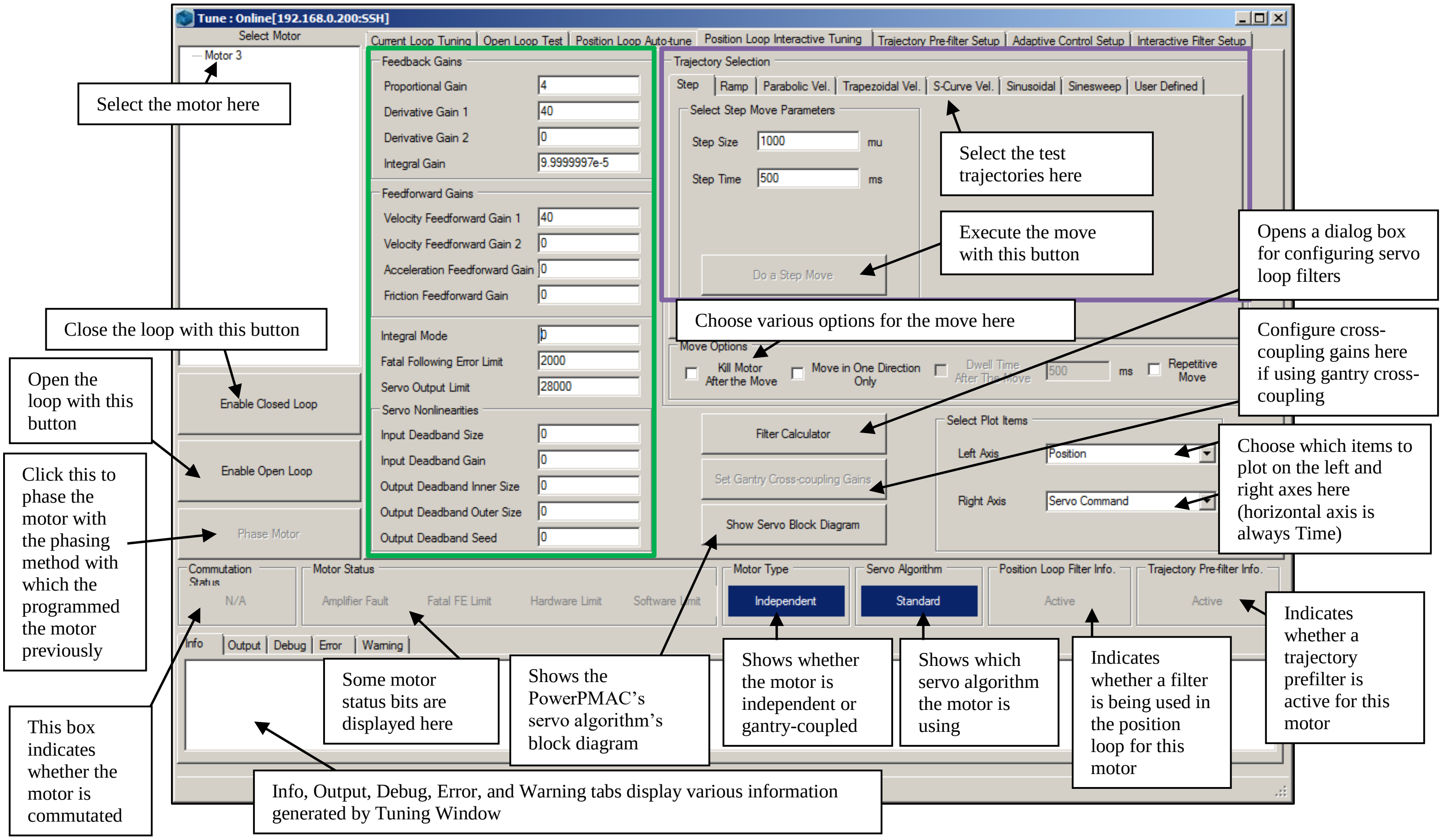


Although the Basic Tuning software provides “automatic tuning” of the servo loop this automatic tuning is only intended as a starting point. It might get the motor to jog but probably will not tune the motor to the exact performance specifications desired. Thus, it is recommended to always use the Tuning software to do any interactive tuning in order to obtain the performance goals desired.

Tuning Window Layout

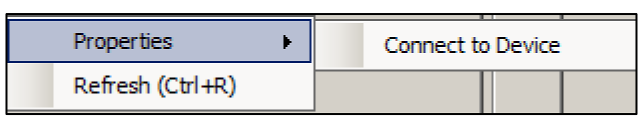
Clicking the “Tune” button in the menu shown above opens up this screen which is the default layout for the Tuning Window:

Select various tabs, different screens of the Tuning Window, here

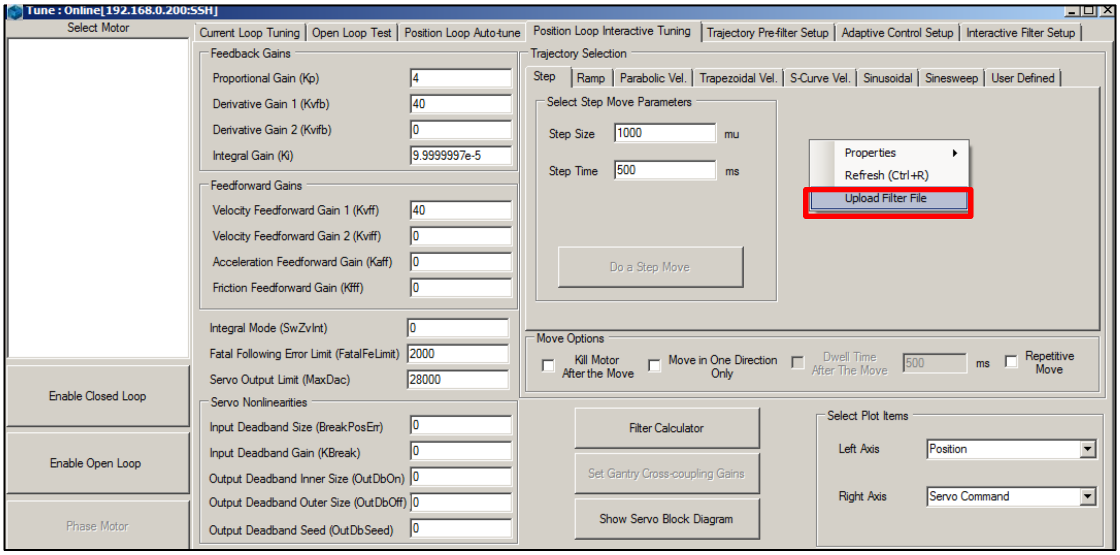


The first tab that opens by default is the "Position Loop Interactive Tuning" tab. This enables the commanding of various test trajectories to the motors, to observe the motor's response and then adjust the servo loop gains accordingly.

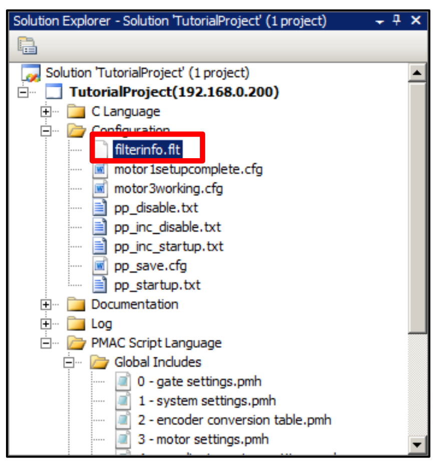
To refresh this window right-click and click Refresh or hit CTRL+R on the keyboard. To connect to another device right click and click Properties→Connect to Device:



To create a file containing values used to calculate servo loop filter or trajectory prefilter parameters that have been configured in Power PMAC right-click on a blank gray space on the tuning screen and then click “Upload Filter File” as shown in the red box below:



The file (**filterinfo.flt**) is added to the Configuration folder in the IDE project as shown in the red box below:

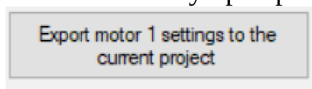


This file contains values that the Tuning software uses to calculate filter gains. Understanding the file’s contents is not important for the user. If this file is present in the Configuration folder, when the Tuning software window is opened, the software will load these settings into the Tuning window in order to show what filter settings are currently being used.

The Tuning software will also compare the values from this file against the filter parameters currently in the Power PMAC and will give a warning if they differ. If the parameters differ, and the filter gains specified in the file are to be retained rather than the filter gains currently in the Power PMAC, go to the Tuning window corresponding to the filter, for example for Servo filters, go to Interactive Tuning→Filter Calculators; or for Trajectory Prefilter, click on the Trajectory Prefilter tab, and then click Calculate→Implement. Note that this file does not contain the Power PMAC parameters but rather the values used to calculate filter-related the Power PMAC parameters.

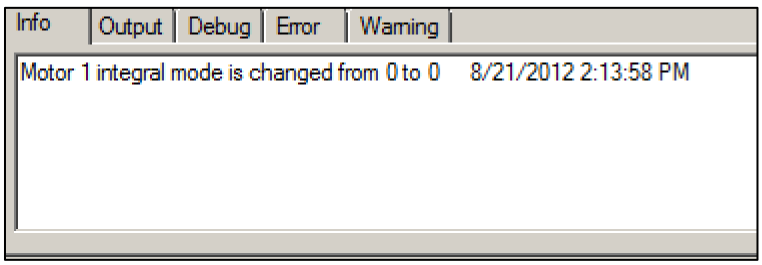
Note

IDE V4.2 onwards additional button “Export current motor<n> settings to current project where n is motor number currently selected for tune. Click this button when satisfied with tune result to copy settings to motor file in currently open project.

Button to click: 

Output Tab

The Tabs at the bottom of the screen contain useful information. Each tab contains a different type of information. There are five tabs as shown below:



Info
This tab displays any changes that the Tuning window made to Power PMAC parameters.

Output
This tab displays any I/O stream text that the window might print.

Debug
This tab announces what operations the window is trying to perform; for example, it will announce that it is preparing a step move or that a trapezoidal move just finished successfully.

Error
The tab will show any errors that the window reports. For example, the Error window will print an error message if the Tuning window tries to command the motor to move for its Automatic Tuning procedure and the motor’s Minimum Move is not made.

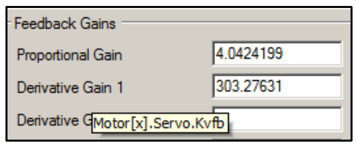
Warning
This tab gives any programming warnings that the window reports.

Position Loop Interactive Tuning
In the screenshot of the Position Loop Interactive Tuning window, under the heading “Tuning Window Layout”, the servo loop gains and other parameters related to servo control are shown within a green box. These can be adjusted, and the program will change the associated parameter within the Power PMAC.

Caution

Since the servo loop gains change as they are altered in the Tuning window only safe gains must be entered in order not to damage the motor or cause it to go unstable, potentially damaging equipment or people.

To understand the exact structure with which the gain parameters listed are associated just hover the mouse cursor over the parameter and a tooltip will appear with the structure name. In the example screenshot below “Derivative Gain 1” shows, via the tooltip, that the associated structure is Motor[x].Servo.Kvfb:



Test Trajectories

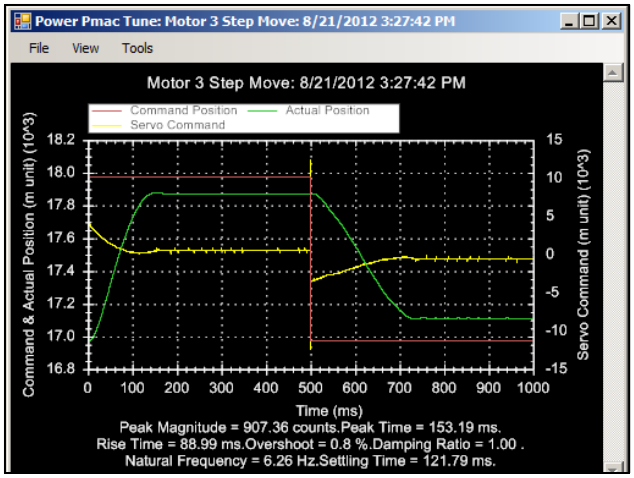
In the purple area in the screenshot of the Position Loop Interactive Tuning screenshot the various test trajectories which the motor can be commanded to are shown. These test trajectories can be useful for identifying characteristics of the motor and tuning it systematically. There are eight different test trajectories available:

- Step,
- Ramp,
- Parabolic Velocity,
- Trapezoidal Velocity,
- S-Curve Velocity
- Sinusoidal, Sinesweep, and
- User Defined.

Most users only need to use Step and Parabolic Velocity as these can be used to tune K_p, Kd, Ki, Kvff, Kaff, and Kfff. The other moves are available to simulate the kind of moves to which might subject the machine to optimize the servo loop’s gains to get the performance required during these kinds of moves.

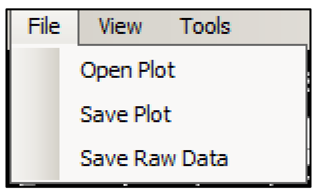
Features Common to All “Trajectories Plots”

Note that there are several properties of the plot window used to display the motor’s response and the test trajectory. Below is an example of a Step move:

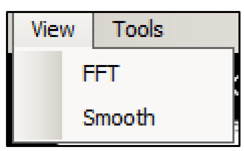


Along the top of the window are three menus: File, View, and Tools.

File opens a plot that was previously saved, saves this plot, or saves the plot's data in a raw data file:



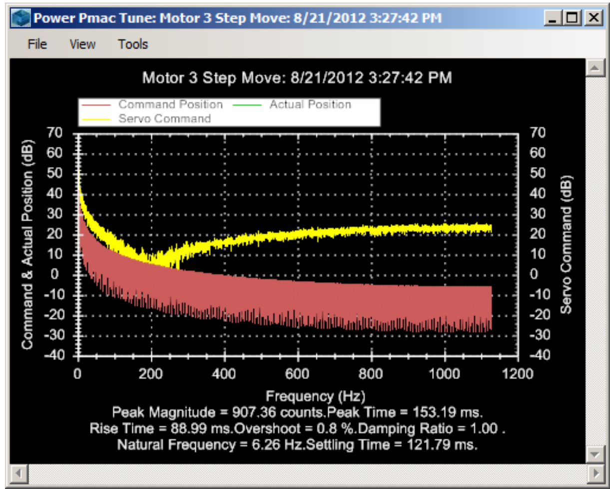
View allows the selection to subject the plot to a Fast Fourier Transform (FFT) or to smooth the data out with a filter:



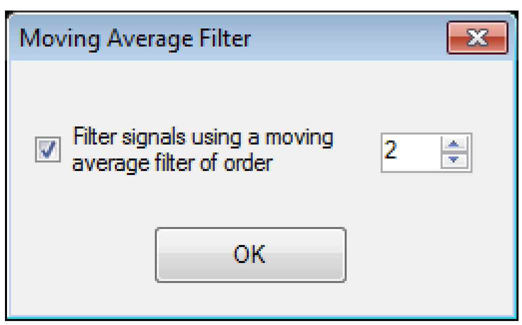
FFT

This tool will perform a Fast Fourier Transform (FFT) of the data. Choose whether to filter the signals or not or whether to plot the vertical axes in units of decibels (dB). Choosing to filter the data will result in the IDE performing a Hanning window filter on the data. If not it will use a Uniform/Rectangular window. The Horizontal Axis will not be logarithmic.

This is an example screenshot of the above Step move transformed with an FFT with filtering and with logarithmic axes:



Smooth
This tool will filter the signals chosen to plot with a moving average whose order can be set from 0 to 10:

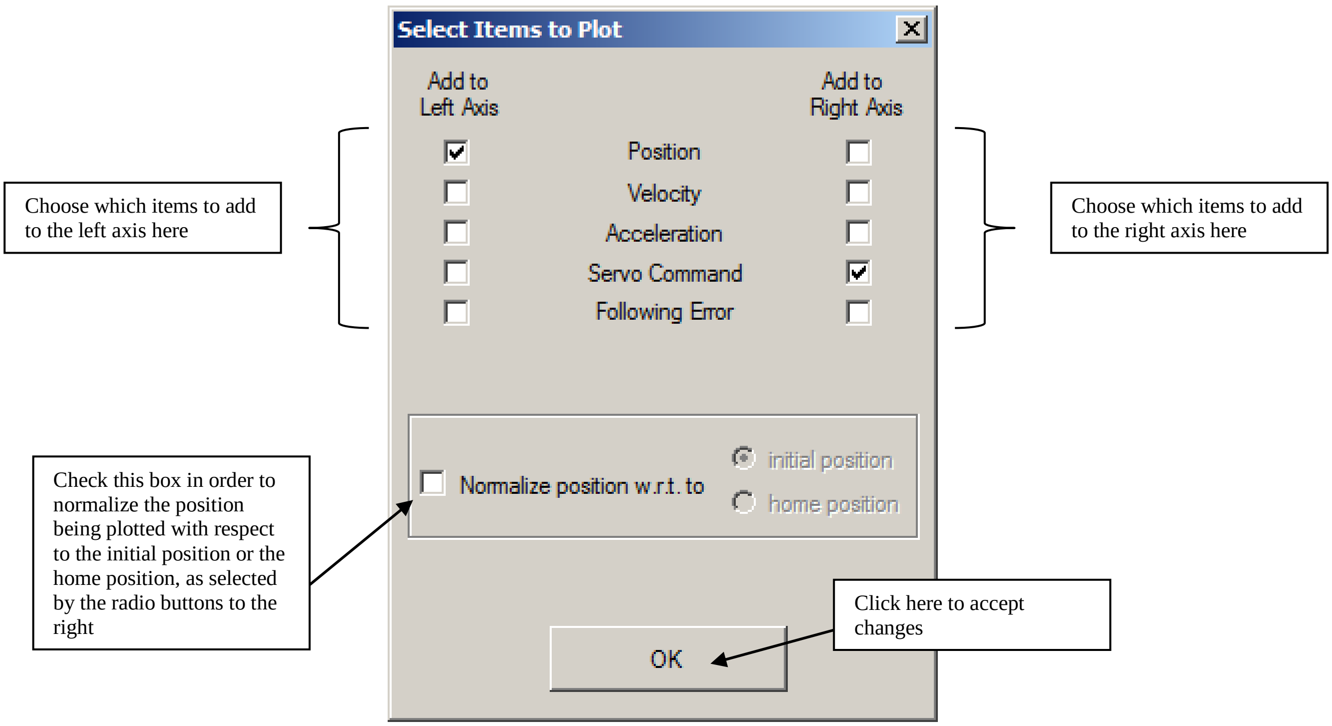


The default filter order is 2. The filter sums groups of points (the number of points in the sum is equal to the order of the filter) and then divides by the order of the filter. The equation of the filter is

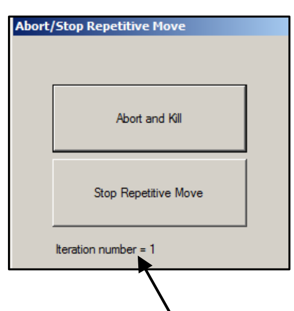
$$p_i = \frac{1}{N}\sum_{k=0}^{N} a_k,$$

where p_i is a point on the plot, where i runs from 0 to the total number of points on the plot, N is the order of the moving average filter, and a_k is a point of data in the group of points of size (N + 1) presently being processed by the filter.

Selecting Tools→Modify Plot Items opens this screen:



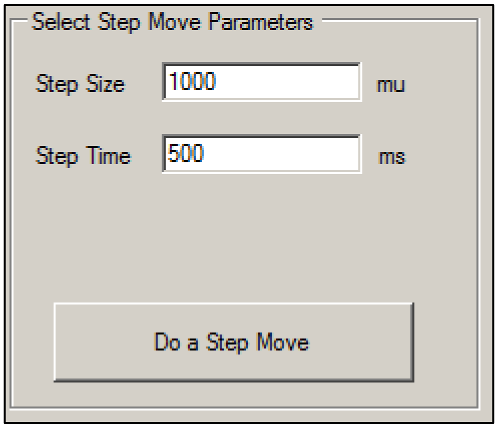
Move Options
There are a few options that apply to all moves directly beneath the test trajectories section of the window. Note by that selecting “Repetitive Move” the move will execute repeatedly until “Stop Repetitive Move” is clicked on the dialog box that pops.



The number of iterations are shown at the bottom of this dialog box.

Trajectories
The various trajectories available are described below:

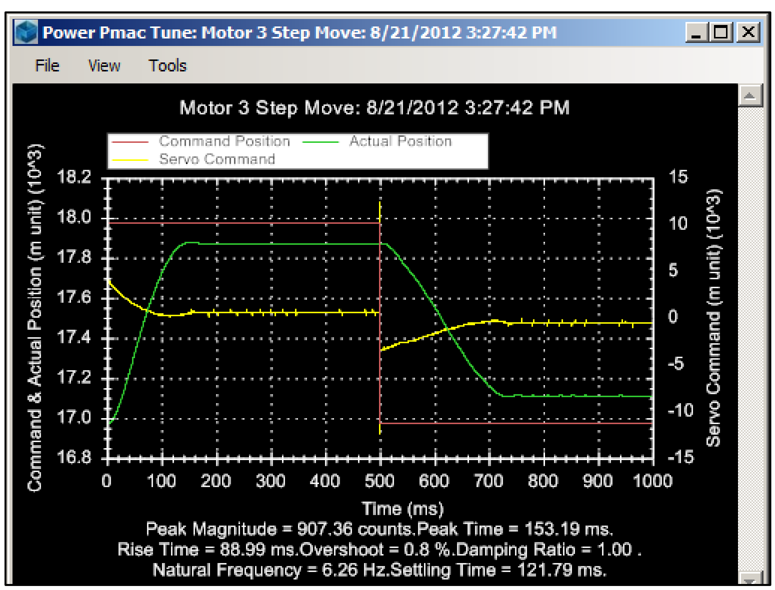
Step
The Step move commands a discontinuous change in desired position to the motor, dwells, and then returns to the starting position. The motor then tries to immediately react to move to the new position.



The only parameters needed to be set are the Step Size, in motor units, which is the magnitude of the instantaneous discontinuous change in position commanded to the motor and the Step Time, in milliseconds, which is how long the desired position dwells before returning to the starting position. These parameters can be in the Tuning window as shown to the left.

In order to keep the move within the linear region, wherein the tuning software operates best, it is recommended to command within ½ to ¼ of the motor's revolution if a rotary motor, or within ½ to ¼ of the motor's electrical cycle if a linear motor.

When ready click "Do a Step Move" to command the move. An example Step move appears below:



The commanded position is shown in red (**Motor[x].DesPos**), the actual position in green (**Motor[x].ActPos**), and the servo command (**Motor[x].ServoOut**) in yellow.

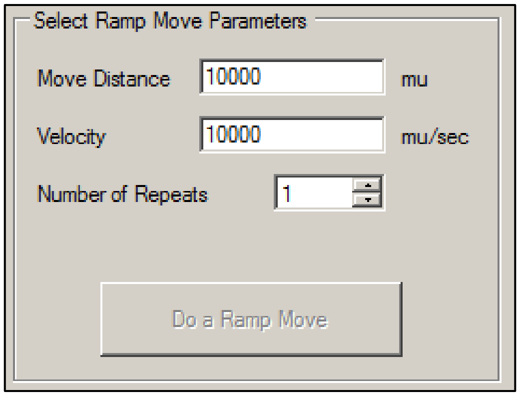
Note

This move is especially useful for determining the values of K_p, K_d, and K_i. See the “Tuning Guidelines” below for more details.

Plotting the servo command on the right axis is always recommended for the Step move. This is because it is possible to see when the servo command has saturated. The servo command has become saturated when its value truncates and becomes completely flat indicating that the servo command has reached its limit **Motor[x].MaxDac**. At this point increasing K_p will not help to improve the motor’s performance.

Ramp

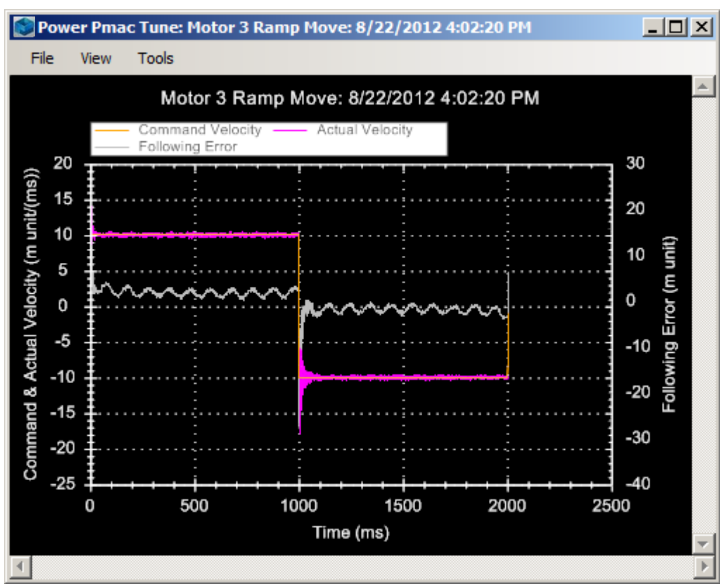
The Ramp trajectory commands a linear increase in motor position in the positive direction for a certain distance and then reverses that command for the same distance, returning the motor to its starting position. The selectable parameters for this move are as shown below:



- "Move Distance" is the distance [motor units] in one direction the motor will be commanded in a linear fashion.
- "Velocity" is the top speed [motor units per second] the motor will be commanded to achieve.
- "Number of Repeats" describes how many times to command the motor forward and back.

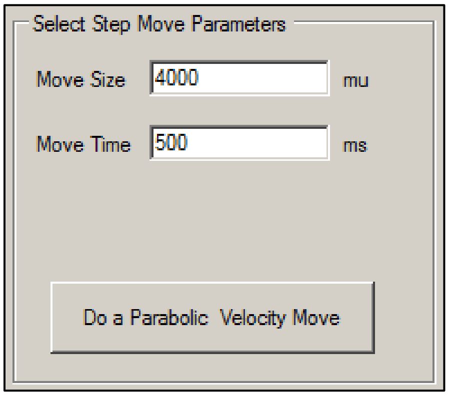
Click "Do a Ramp Move" when ready.

A plot similar to the one below will be shown:



Parabolic Velocity

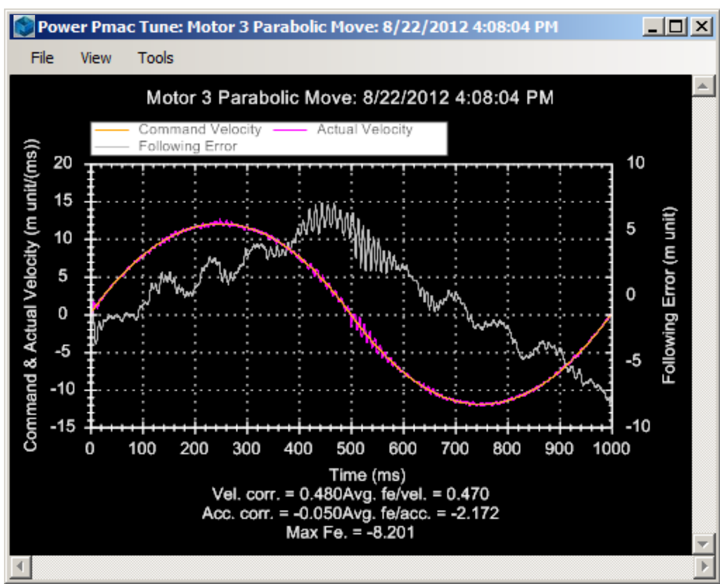
The Parabolic Velocity move commands a parabolic velocity trajectory first in the positive direction and then in the opposite direction to the motor. The parameters that can be specified for this motor are shown below:



- "Move Size" is the distance [motor units] the motor will first travel in the positive direction before reversing and traveling the same distance in the opposite direction.
- "Move Time" is the time span [msec] within the motor will traverse the distance specified in "Move Size" in the positive direction, and then that same distance in the opposite direction within the "Move Time" time span again

Click "Do a Parabolic Velocity Move" when ready.

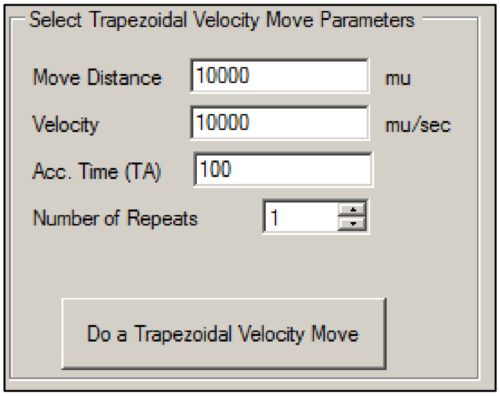
A plot similar to the one below will be shown:



Note

This move is especially useful for determining the values of K_{aff}, K_{vff}, and K_{fff}. See the "Tuning Guidelines" below for more details.

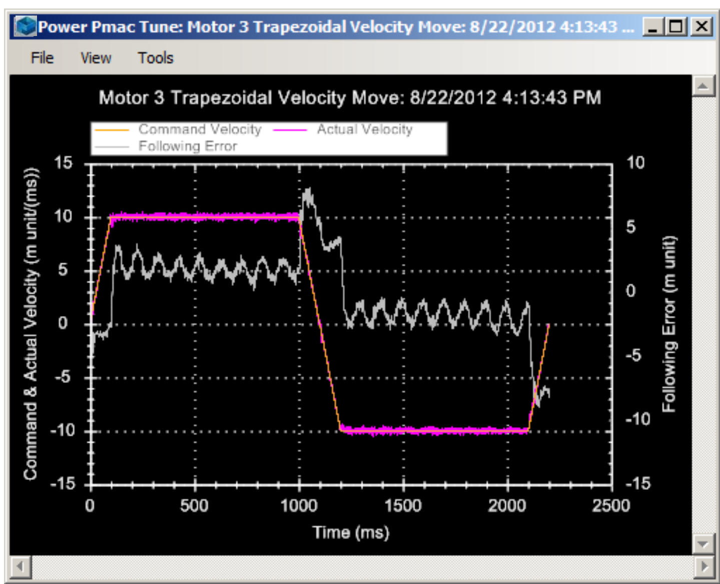
Trapezoidal Velocity
This trajectory commands a trapezoid-shaped velocity profile to the motor first in the positive direction and then in the negative direction. Below are the parameters that can be adjusted for this move:



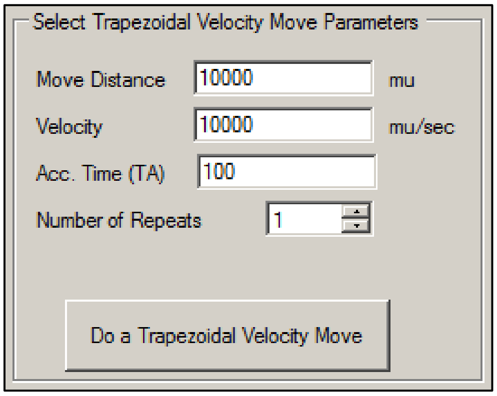
- "Move Distance" is the total distance [motor units] the motor will move in the positive direction before reversing and traveling that same distance in the opposite direction.
- "Velocity" is the maximum speed [motor units per second] commanded to the motor at the peak of the velocity profile.
- "Acceleration Time (TA)" is the time span [msec] over which the motor will accelerate to the top speed specified in "Velocity" right above this field.
- "Number of Repeats" is how many times to execute the forward-and-back motion path.

Click "Do a Trapezoidal Velocity Move" when ready.

A plot similar to the one below will be shown:



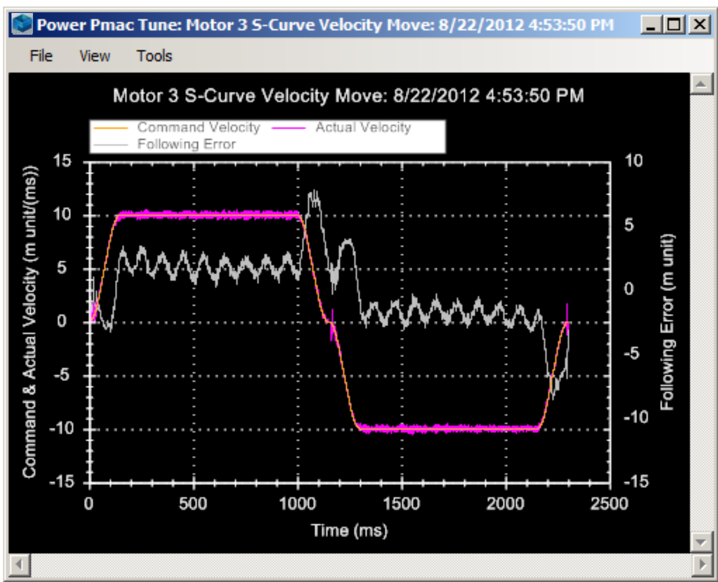
S-Curve Velocity
The S-Curve Velocity trajectory commands a cubic B-Spline shape to the motor's velocity first in the positive direction and then in the negative direction. The parameters that can be specified are shown below:



- "Move Distance" is the total distance [motor units] the motor will move in the positive direction before reversing and traversing the same distance in the opposite direction.
- "Velocity" is the peak speed [motor units per second] commanded to the motor in each direction.
- "Acceleration Time (TA)" is the time span over which the motor will accelerate to the speed specified in the "Velocity" field immediately above this field.
- "Number of Repeats" is the number of times the motor should perform the forward-and-back motion path.

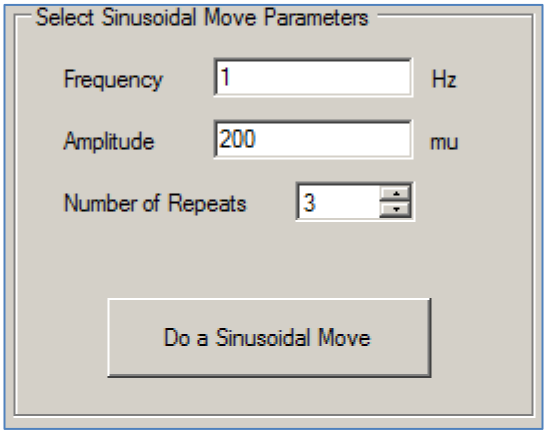
Click "Do a S-Curve Velocity Move" when ready.

A move similar to the one below will be shown:



Sinusoidal

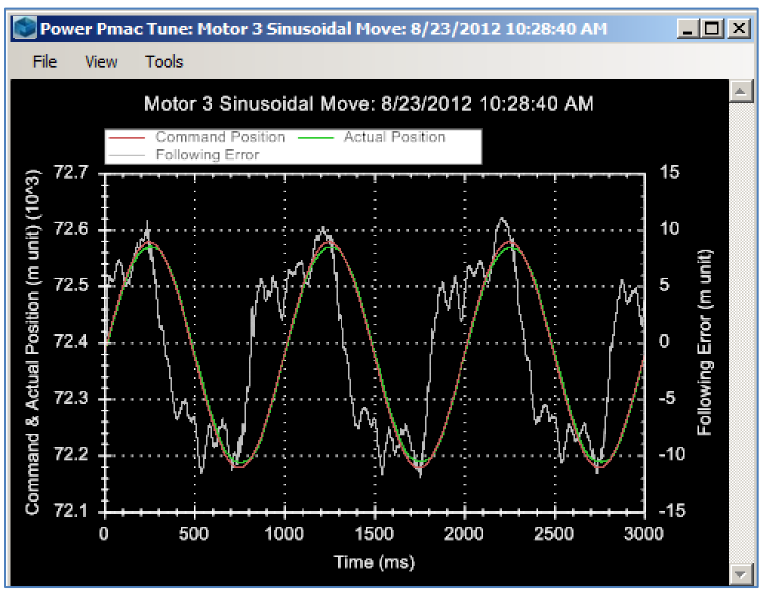
This trajectory commands a sine wave position signal to the motor. The parameters that can be specified are shown below:



- "Frequency" is the frequency of the sine wave [Hz].
- "Amplitude" is the amplitude of the sine wave [motor units].
- "Number of Repeats" is the number of periods of the sine wave to command the motor to traverse.

Click "Do a Sinusoidal Move" when ready.

A move similar to the one below will be shown:

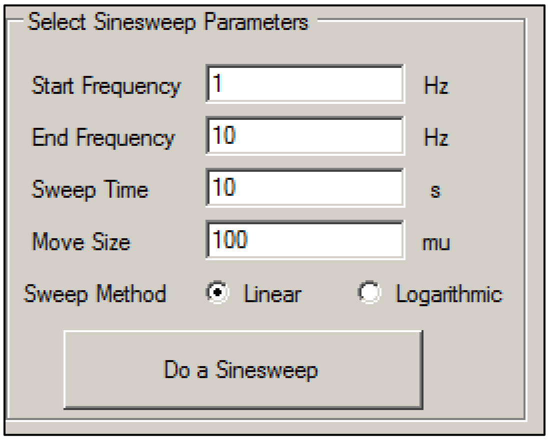


Note

This move is especially useful for system identification purposes. Try exciting the system at different frequencies, letting the motor enter a steady-state each time, and then exporting the data set. This can then be imported into, for example, MATLAB™ and batch-processed to produce a Bode magnitude/phase plot for the purpose of identifying DC gain, natural frequencies, and damping ratios.

Sinesweep

Sinesweep commands a sine wave position signal to the motor. This signal is different from the Sinusoidal test trajectory in that while the Sinusoidal trajectory remained at a constant frequency the Sinesweep trajectory's frequency increases either linearly or logarithmically, at the user's choice, over a time span that the user specifies. The parameters available can vary as follows:



- "Start Frequency" is the initial frequency [Hz] of the sine signal at the start of the move.
- "End Frequency" is the final frequency [Hz] of the sine signal at the end of the move. The signal should reach this frequency by the end of the "Sweep Time" [sec] specified in the field immediately below this one.
- "Sweep Time" is the time span [sec] over which the sine wave will be commanded to the motor; this is the time span over which the sine wave's frequency will increase either linearly or logarithmically as specified in the "Sweep Method" parameter two fields below this one.
- "Move Size" is the amplitude [motor units] of each period of the sine wave.

"Sweep Method" describes the manner in which to increase the frequency of the wave being commanded to the motor. Selecting Linear will increase the frequency (*f(t)* below) linearly such that the frequency change with time (*t* below) follows the following formula:

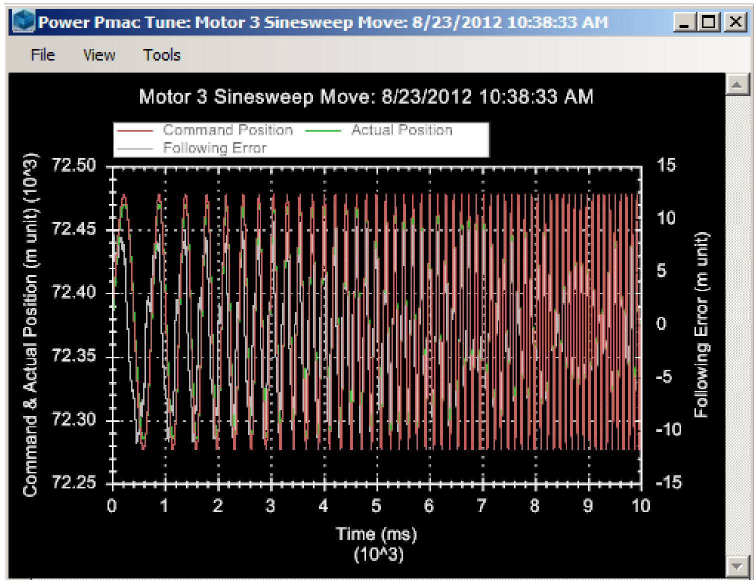
$$f(t)=((f_{end} - f_{start})/(T_{sweep}))t + f_{start},$$

where f_{end} is the "End Frequency" specified, f_{start} is the "Start Frequency" specified, and T_{sweep} is the "Sweep Time" specified. Selecting Logarithmic will increase the frequency logarithmically according to the following equation:

$$f(t) = f_{start} \cdot \left(\frac{f_{end}}{f_{start}}\right)^{\frac{t}{T_{sweep}}}$$

Click “Do a Sinesweep” when ready.

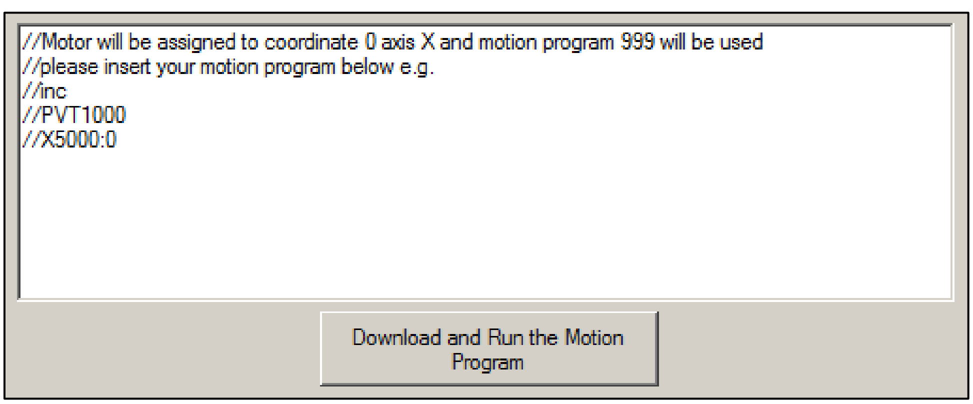
A plot similar to the one below will be shown:



User Defined

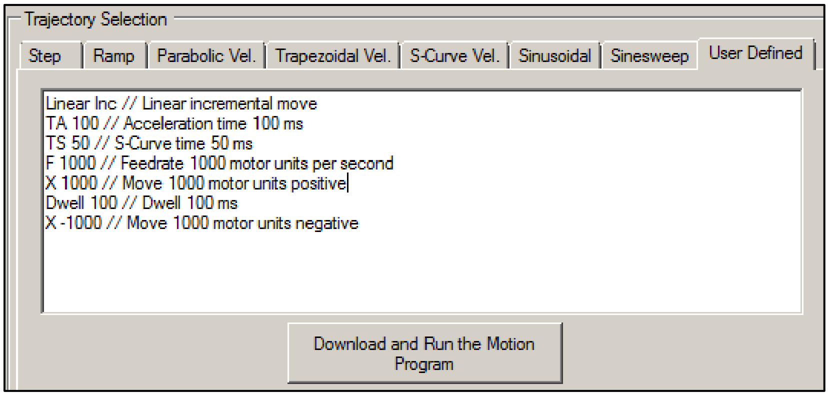
The final test trajectory available is user defined (i.e. the user designs it by writing a motion program). This is equivalent to adding a motion program to the project and running it but doing this in the Tuning software has a few advantages, namely that the motion program is only downloaded temporarily to be run once each time and the Tuning software automatically gathers and plots the motor’s response. This makes designing a move e.g. a move similar to that which the machine might actually experience once it is commissioned, and testing it to see if the servo loop gains produce the performance that desired.

The interface for designing the trajectory is as follows:



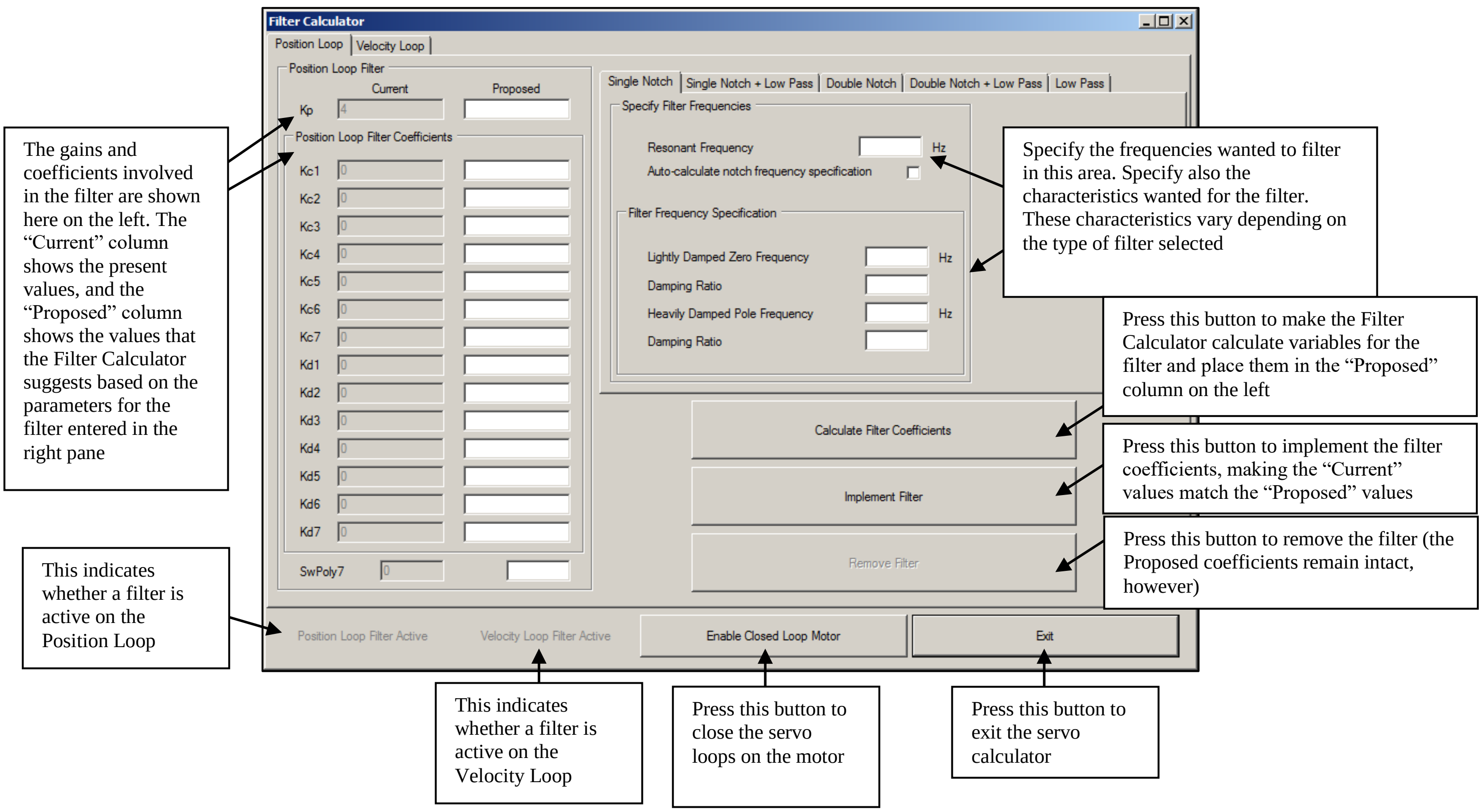
Type the motion program in the area provided and click “Download and Run the Motion Program”. The motion programs need to be written in order to achieve this. For more details on writing motion programs refer to the Power PMAC User’s Manual and the section labeled “Writing and Executing Script Programs in the Power PMAC”. The motor selected will be assigned to Coorrdinate System 0, Axis X and the program will be run in Motion Program 999.

This is an example of a simple motion program:



Filter Calculator

Clicking on the Filter Calculator on the Position Loop Interactive Tuning tab opens the following screen:



From this screen it is possible to design Single Notch, Single Notch/Low Pass, Double Notch, Double Notch/Low Pass, Low Pass filters for the Position Loop and a Low Pass filter for the Velocity Loop.

Note

The Filter Calculator should be used by Advanced Users only, that is, those who are familiar with filters and the parameters associated therewith. These filters consist of bilinear transformations, computations in the continuous time domain, and then a Tustin discretization process.

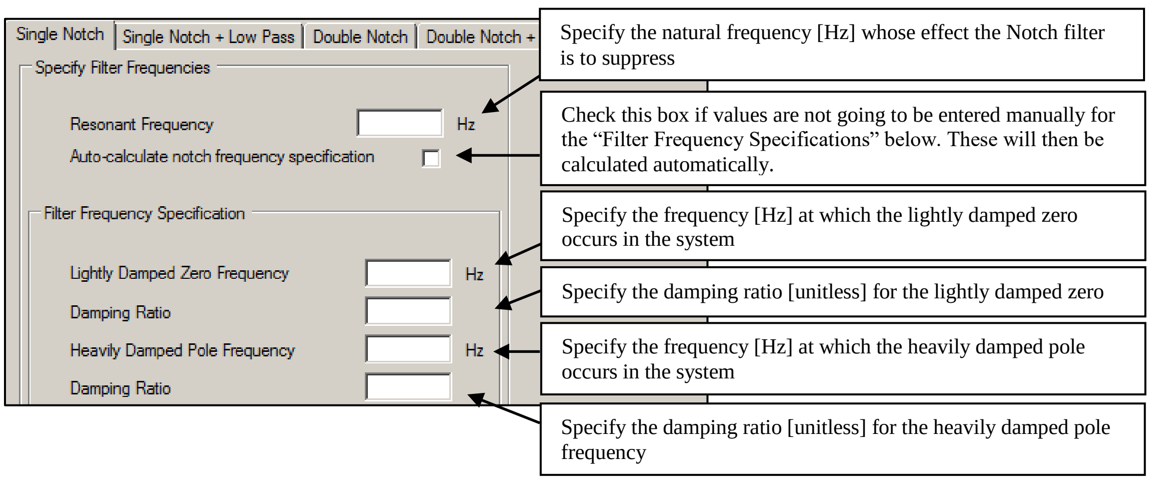
Note

Some of the servo loop gains must be modified when a filter is implemented when "Implement Filter" is clicked. If a retune is needed with these gains later it is recommended to first remove the filter, retune and then recalculate and reapply the filter.

Position Loop Filters

Single Notch

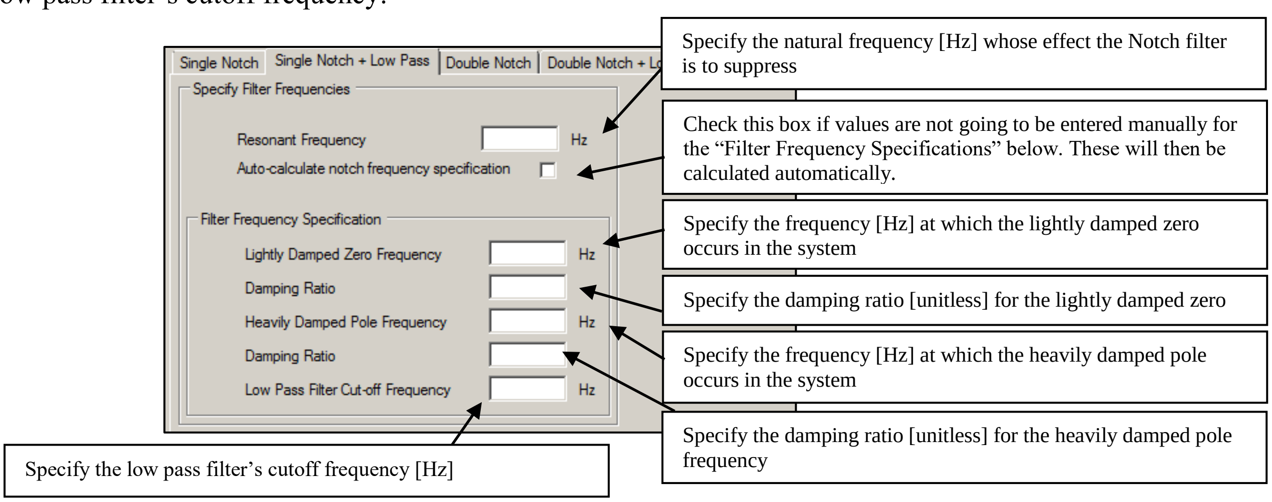
Under the single notch tab, the following parameters are available:



Single Notch/Low Pass

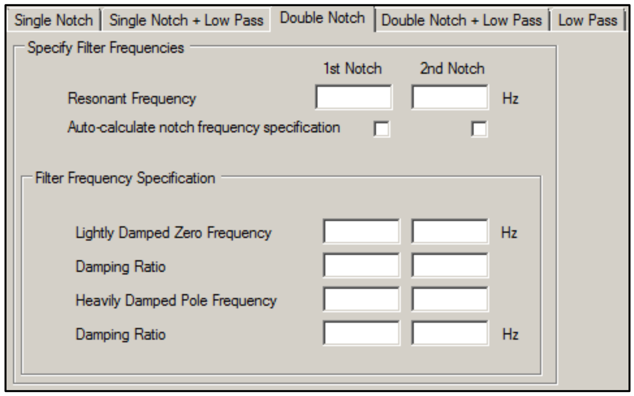
This filter type is the combination of a notch filter at a natural frequency that is specified and a low pass filter to attenuate frequencies above that which has been specified.

This filter's tab's layout looks much like the Single Notch tab but has one more field for the user to specify the low pass filter's cutoff frequency:



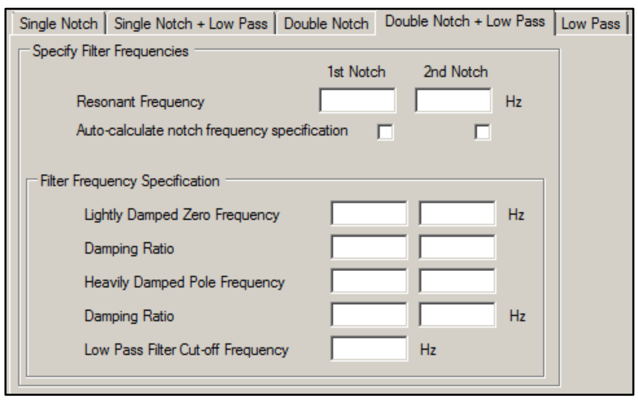
Double Notch

This filter type consists of two notch filters each of whose resonant frequencies can be specified separately. The configuration screen for this filter contains two sets of the same parameters listed under the Single Notch screen; see Single Notch above for the description of the fields:



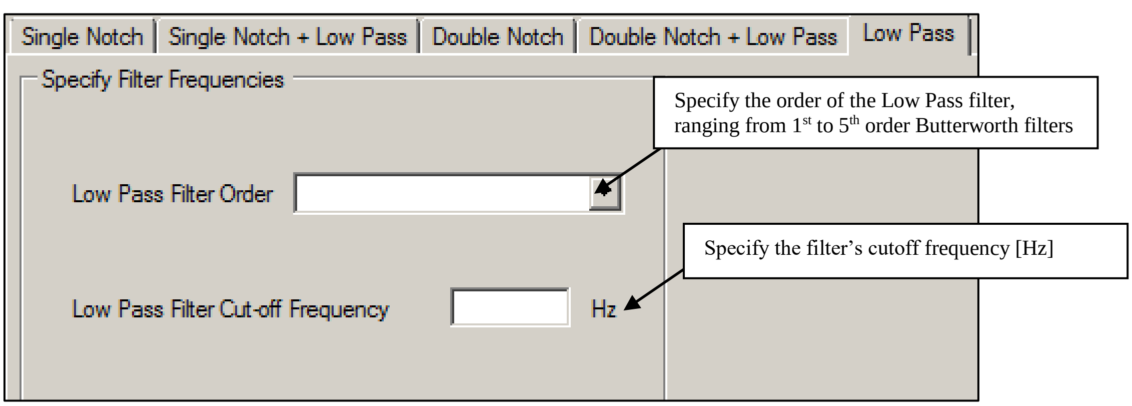
Double Notch/Low Pass

This filter type consists of two Notch Filters and one Low Pass filters. The parameters listed are the same as those listed in the Double Notch and the Low Pass filter screens:



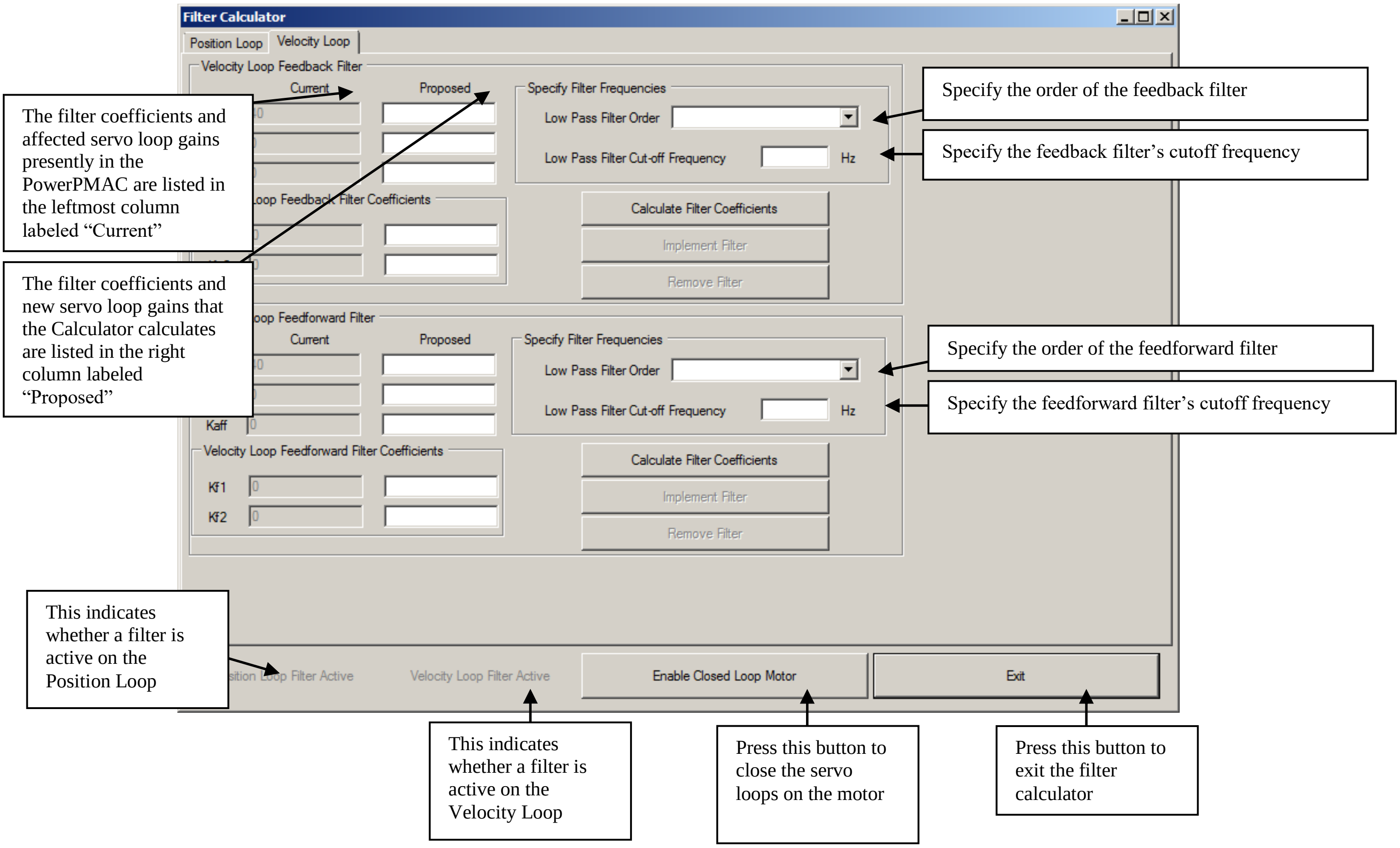
Low Pass

This filter attenuates frequencies above the cutoff frequency which can be specified:



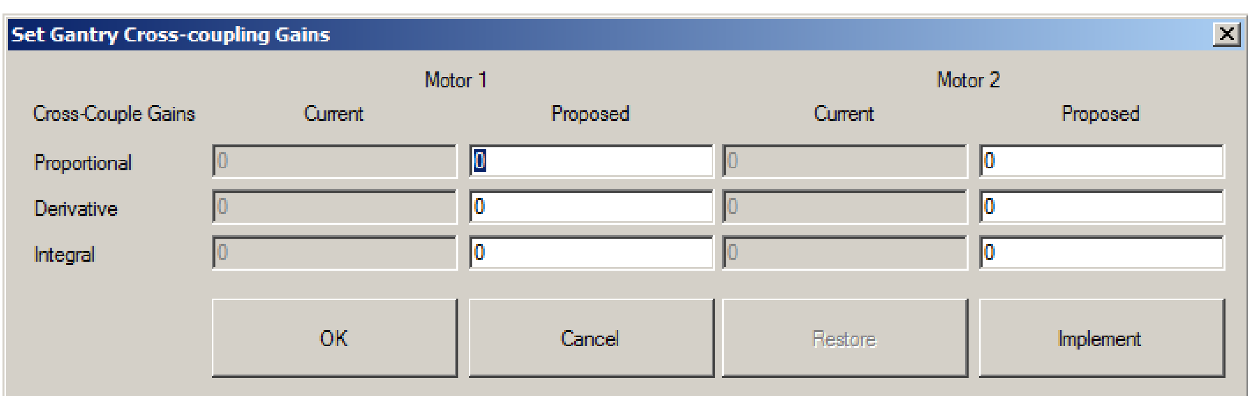
Velocity Loop Filters

The Velocity Loop filter page is used to configure 1st or 2nd order Butterworth Low Pass filters for the motor's feedback and feedforward velocity loops separately:



Set Gantry Cross-Coupling Gains

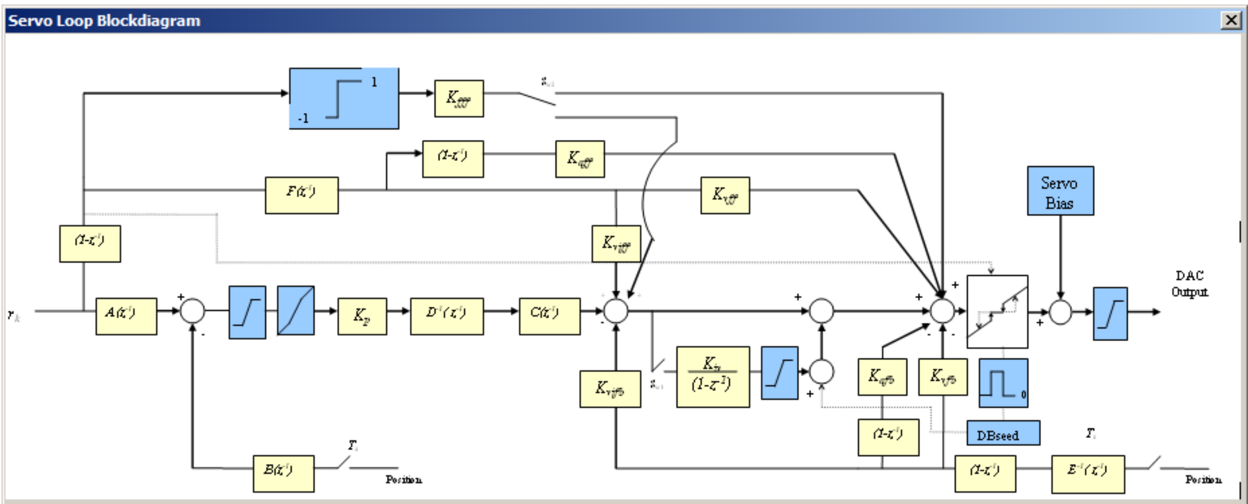
This feature is only available if the motor is using the Gantry Cross-Coupled servo algorithm i.e. when **Motor[x].Ctrl=Sys.GantryXCtrl**. This screen shows PID gains for each motor:



“Current” shows which gains are presently in the the Power PMAC. “Proposed” are the gains the window suggests for this pair of motors. “Implement” implements the proposed gains causing “Current” and “Proposed” to become the same. “Restore” will revert the effects that “Implement” caused.

Show Servo Block Diagram

Clicking the Show Servo Block Diagram button in the Position Loop Interactive Tuning tab will show the following screen:



This screen shows the block diagram for the Standard servo algorithm in Power PMAC.
This particular screenshot above is from the Standard servo algorithm. This diagram is disabled if a custom servo algorithm is chosen.

Note

The Servo Block Diagram shows only the Standard servo algorithm. If any other servo algorithm is chosen this diagram will not apply to this motor.

Interactive Tuning Guidelines

PMAC’s Servo Algorithm must be configured to properly control any given system with motors and amplifiers. Configuration is done by adjusting setup structures pertaining to the PID gains. Friction Feedforward is also needed. The most basic servo loop gains correspond to structures as follows:

- **Motor[x].Servo.Kp** Proportional Gain (K_p)
- **Motor[x].Servo.Kvfb** Derivative Gain (K_d)
- **Motor[x].Servo.Kvff** Velocity Feedforward (K_{vff})
- **Motor[x[.Servo.Ki** Integral Gain (K_i)
- **Motor[x].Servo.SwZvInt** Integration Mode
- **Motor[x].Kaff** Acceleration Feedforward (K_{aff})
- **Motor[x].Kfff** Friction Feedforward (K_{fff})

Note

The load should be connected to the motor before tuning the servo loop.

The process of determining proper values of PID gains is called “Tuning”. The procedure for tuning is as follows:

1. Set **Motor[x].Servo.SwZvInt** (Motor xx PID Integration Mode). This can be changed as needed
   =1, position error integration is performed only when Motor xx is not commanding a move
   =0, position error integration is performed always

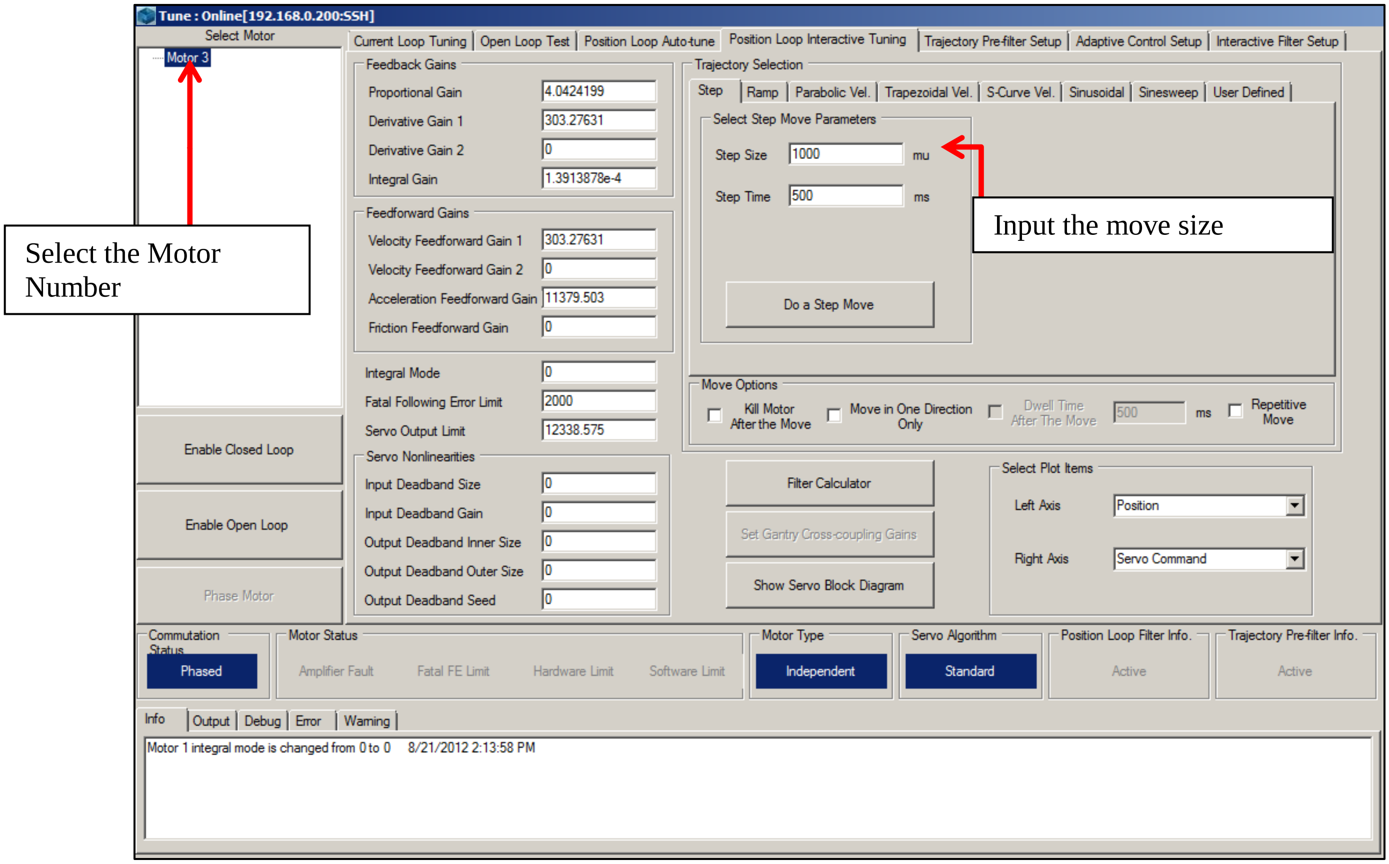
2. Using the Step Response tune the following parameters in this order:
   Proportional Gain, Kp (**Motor[x].Servo.Kp**)
   Derivative Gain, Kd (**Motor[x].Servo.Kvfb**)
   Integral Gain, Ki (**Motor[x[.Servo.Ki**)

3. Using the Parabolic Move tune the following parameters in this order:
   Velocity Feedforward, Kvff (**Motor[x].Servo.Kvff**)
   Acceleration Feedforward, Kaff (**Motor[x].Kaff**)
   Friction Feedforward, Kfff (**Motor[x].Kfff**)

Note

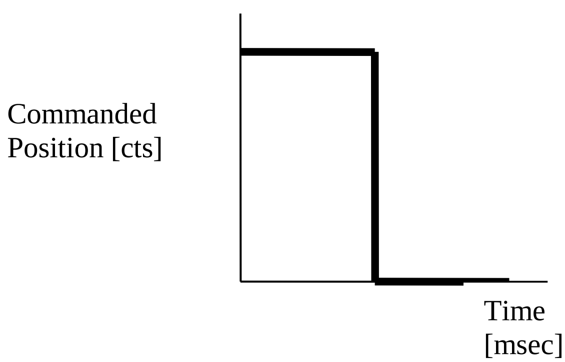
- When tuning the feedforward gains set **Motor[x].Servo.SwZvInt** =1 so that the dynamic behavior of the system may be observed without integrator action. After tuning these set **Motor[x].Servo.SwZvInt** back to the desired setting.
- Setting Kvff = Kd (**Motor[x].Servo.Kvff** = **Motor[x].Servo.Kvfb**) is a good place to start when tuning Kvff.

Steps 2 and 3 should be performed in the Interactive Tuning window in Tuning:



Step 2 (tuning K_p, K_d, and K_i)

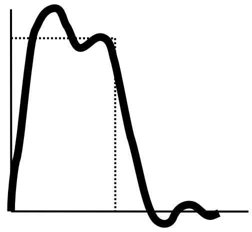
Select "Position Step" under "Trajectory Selection". Choose a "Step Size" that is within ½ to ¼ of a revolution of the motor, if it is a rotary motor, or within ½ to ¼ of one electrical cycle, if it is a linear motor. The step move's commanded position profile should look something like this:

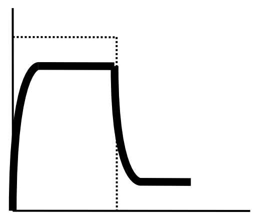


Compare the motor's actual position to the commanded position profile. Depending how the actual position looks adjust the servo loop gains until the desired response is achieved.

Observing the table below, match the actual position response to one of the response shapes below and then adjust the appropriate gain as listed next to each plot. In each of the figures below the vertical axis corresponds to Commanded Position [cts] and the horizontal axis to Time [msec]:

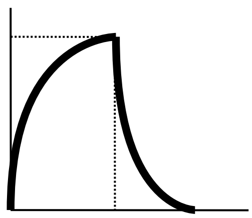
Overshoot and Oscillation
Cause:
Too much Proportional gain or
too little Damping
Fix:
Decrease K_p
Increase K_d

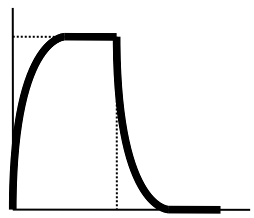




Position Offset
Cause:
Friction or Constant Force
Fix:
Increase K_i
Increase K_p

Sluggish Response
Cause:
Too much Damping or too little Proportional gain
Fix:
Increase K_p or
Decrease K_d

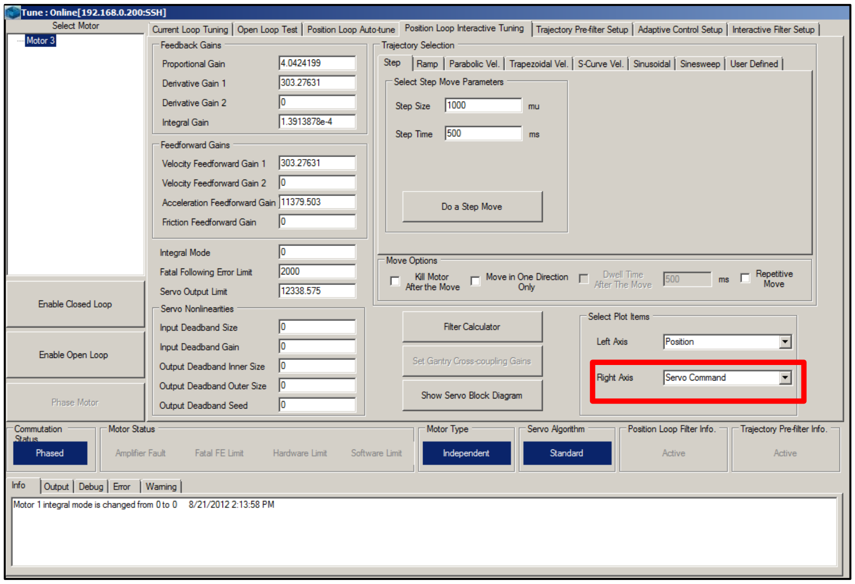




Physical System Limitation
Cause:
Limit of the Motor/Amplifier/Load and gain combination
Fix:
Evaluate Performance and
maybe add K_p

Typically, start by increasing K_p until an “Overshoot and Oscillation” condition is observed and then increase K_d and K_i until the performance goals for the step response are achieved. When executing the step response make sure that the Servo Command is selected on the Right Axis as shown in the red box in image below.

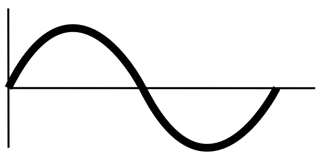
If there is a truncation of the servo command at the beginning of each move the maximum output command, as determined by **Motor[x].MaxDac**, has been reached. In this case adding more K_p will not improve the Step Response’s performance.

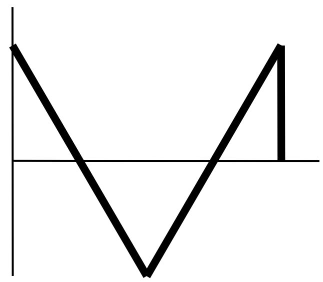


Step 3 (Tuning K_{vff}, K_{aff}, and K_{fff})
Select “Parabolic Velocity” under the “Trajectory Selection” in the Interactive Tuning Window. Select a move size and speed that will simulate the fastest, harshest moving conditions it is expected that the machine will experience. By tuning the motor at these settings the motor should be able to handle all the easier moves.

After commanding the Parabolic Velocity move the commanded Velocity Profile and Acceleration Profile should look like this:

Velocity
Commanded
Profile

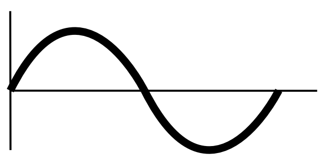


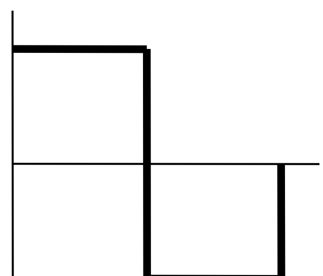


Acceleration
Commanded
Profile

Observing the table below, match the actual position response to one of the response shapes below and then adjust the appropriate gain as listed next to each plot. In each of the plots below the vertical axis corresponds to Actual Velocity [cts/msec] and the horizontal axis to Time [msec]:

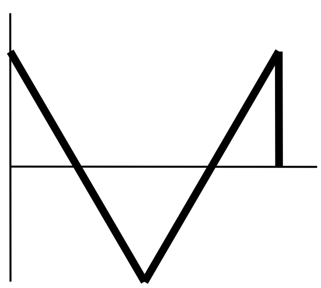
High Vel./F.E.
Correlation
Cause: Damping
Fix: Increase K_{vff}

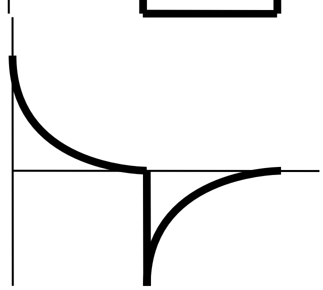




High Vel./F.E.
Correlation
Cause: Friction
Fix:
Add K_{fff}
and/or turn on Integral
Gain (K_i)

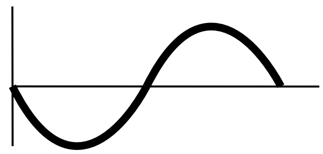
High Acc./F.E.
Correlation
Cause: Inertial Lag
Fix:
Increase K_{aff}

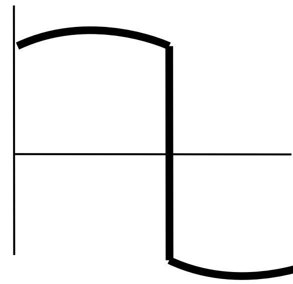




High Acc./F.E.
Correlation
Cause:
Physical System
Limitation
Fix:
Use softer acceleration
or add more K_{fff}

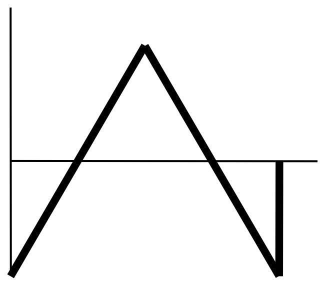
Negative Vel./F.E. Correlation
Cause:
Too much Velocity Feedforward
Fix:
Decrease K_{vff}

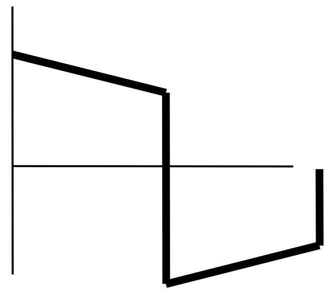




High Vel./F.E. Correlation
Cause: Damping & Friction
Fix:
Increase K_{vff} first
Possibly adjust K_{fff}

Negative Acc./F.E. Correlation
Cause:
Too much Acceleration Feedforward
Fix:
Decrease K_{aff}





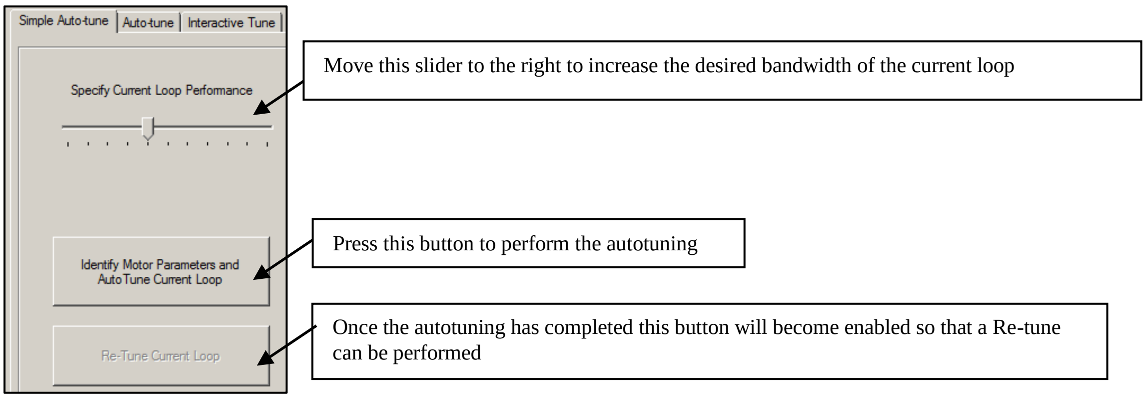
High Vel./F.E. & Acc./F.E.
Correlation
Cause:
Inertial Lag & Friction
Fix:
Increase K_{aff}
Possibly adjust K_{fff}

Note

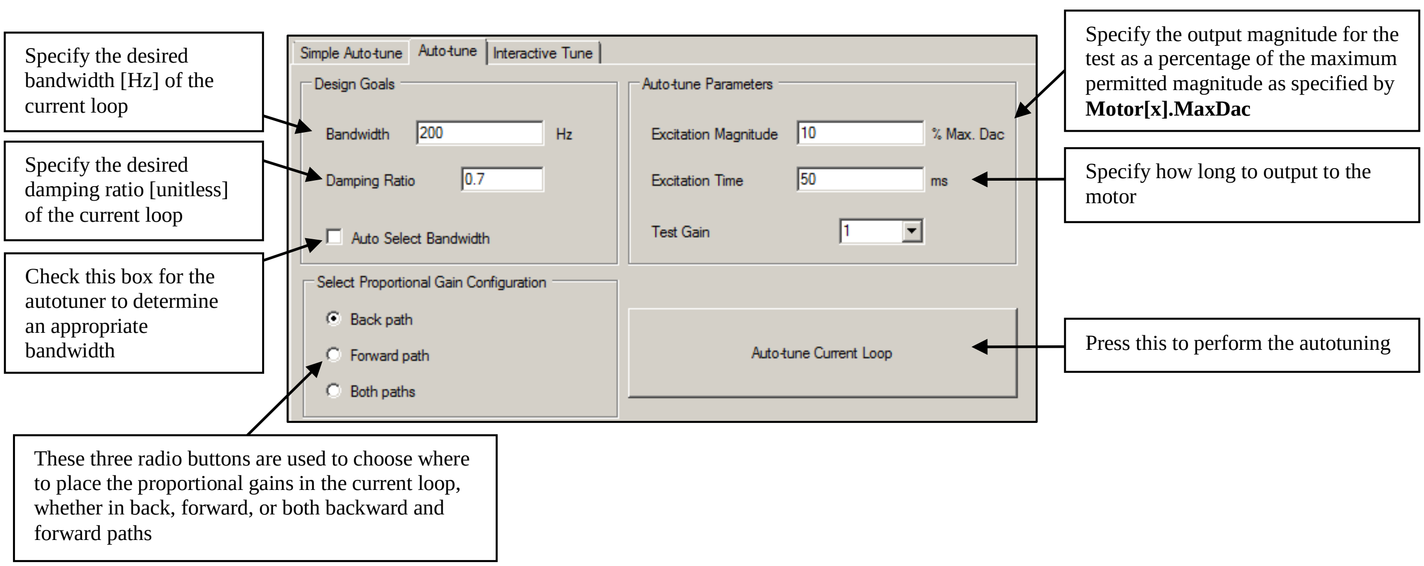
The aforementioned guidelines are just for tuning the PID parameters. For more details on configuring filters or custom servo algorithms, please consult the other areas of this manual or check the Power PMAC User's Manual's "Setting Up the Servo Loop" section.

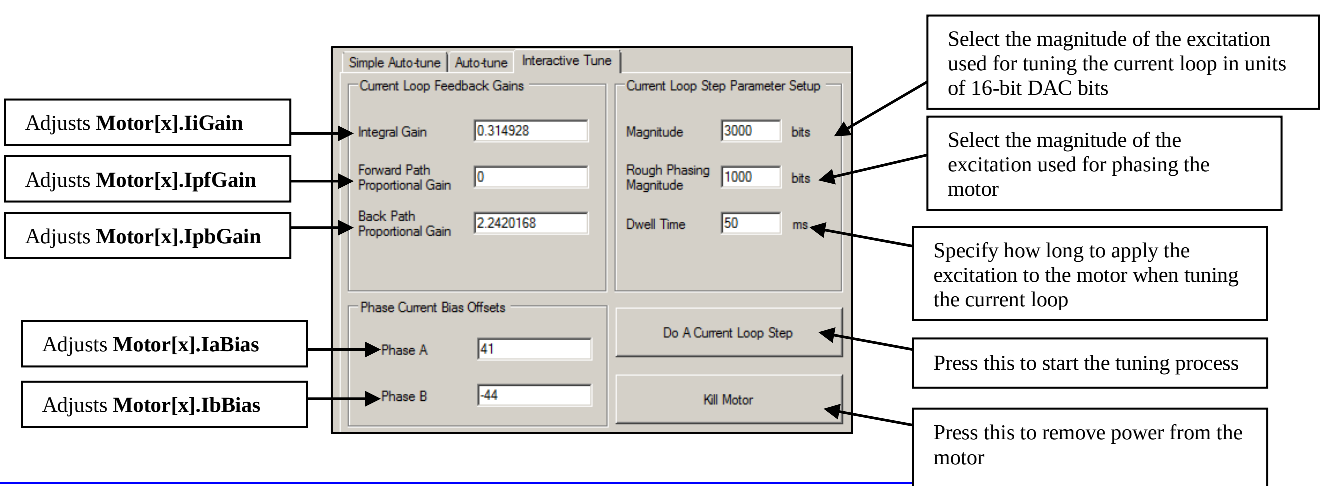
Current Loop Tuning

This tab has three sub tabs, Simple Auto-Tune, Auto-Tune and Interactive Tune. The first subtab is Simple Auto-Tune:



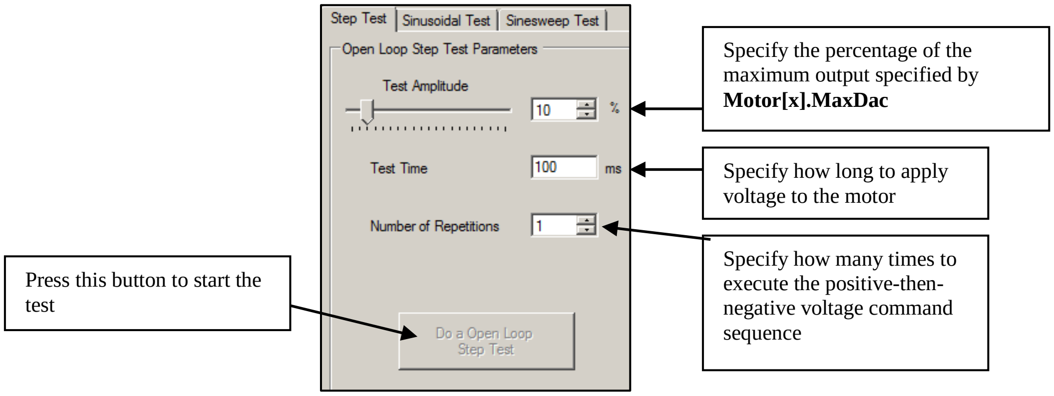
The next subtab is Auto-Tune:



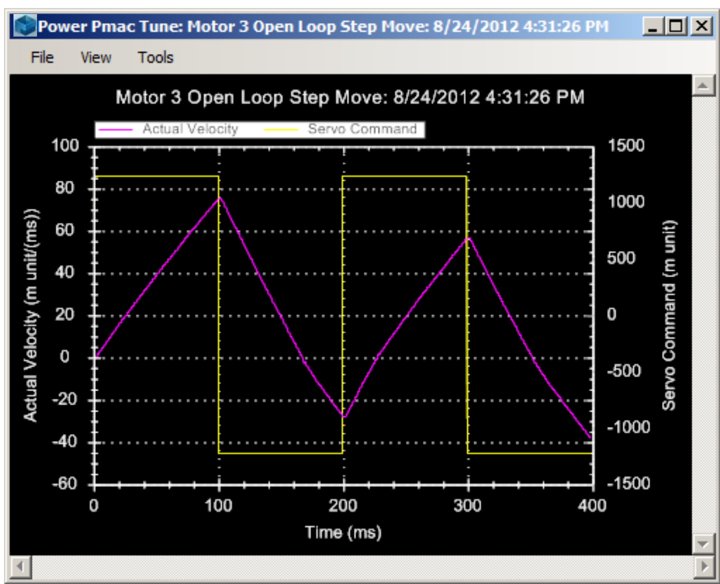


Open Loop Test
There are three sub tabs under the Open Loop Test tabs, Step Test, Sinusoidal Test and Sinesweep Test.

The Step Test instantaneously commands first positive voltage and then negative voltage to the motor:



This should produce a plot similar to the one shown below; this is with two repetitions:

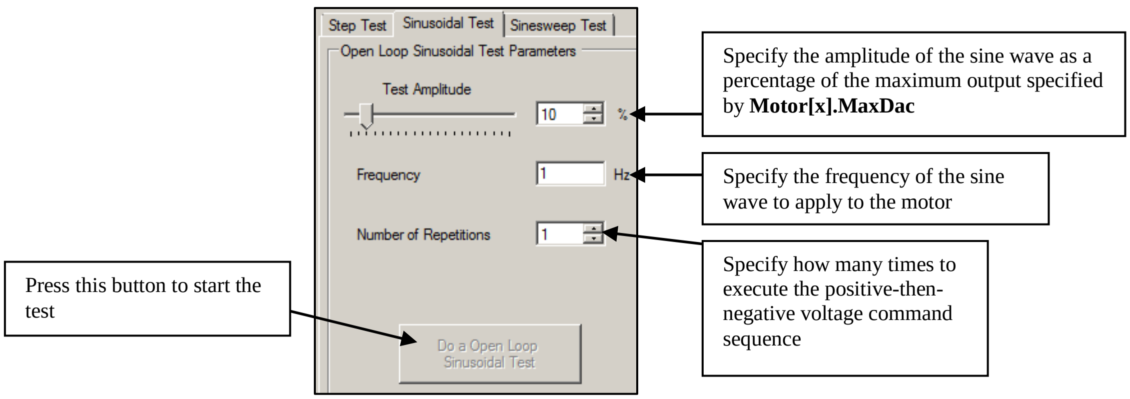


If the encoder feedback is working properly there should ba a positive actual velocity (pink) when the servo command (yellow) is positive and negative actual velocity when the servo command is negative. If the actual velocity is the opposite of what the previous sentence describes try changing the encoder decode direction of the Axis Interface and rephase the motor, if it is commutated.
The encoder decode for Gate1-Style Axis Interfaces is in **Gate1[i].Chan[j].EncCtrl**.
For Gate3-Style it is in **Gate3[i].Chan[j].EncCtrl**.
To reverse the direction, if this structure is a 3, change it to 7 and vice versa. This only applies to quadrature encoders.

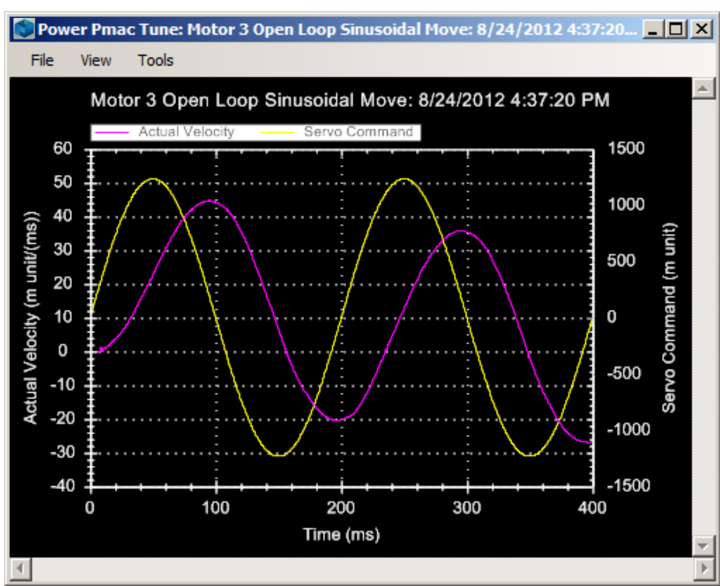
WARNING	Deactivate the bus power and allow the amplifier's capacitors to discharge fully before swapping motor phases, as described below, in order not to risk the cause of Electric Shock. Receiving the discharge of bus capacitors can be fatal!

Another way to change the motor's direction is by swapping two phases of the motor leads and then rephasing the motor.

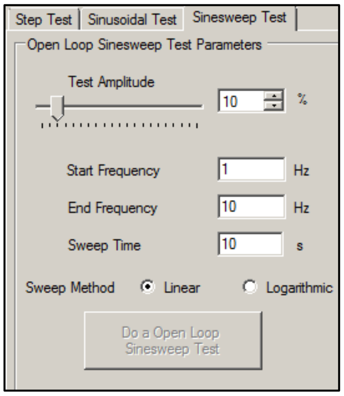
The Sinusoidal Test applies a sinusoidal voltage to the motor:



You should get a plot similar to this:



The Sinesweep Test applies a sine wave voltage signal to the motor. This signal is different from the Sinusoidal Open Loop Test in that while the Sinusoidal test remains at a constant frequency the Sinesweep test's frequency increases either linearly or logarithmically, at the user's choice, over a time span the user specifies:



- "Start Frequency" is the initial frequency [Hz] of the sine signal at the start of the move.
- "End Frequency" is the final frequency [Hz] of the sine signal at the end of the move. The signal should reach this frequency by the end of the "Sweep Time" [sec] specified in the field immediately below this one.
- "Sweep Time" is the time span [sec] over which the sine wave will be commanded to the motor; this is the time span over which the sine wave's frequency will increase either linearly or logarithmically as specified in the "Sweep Method" parameter two fields below this one.

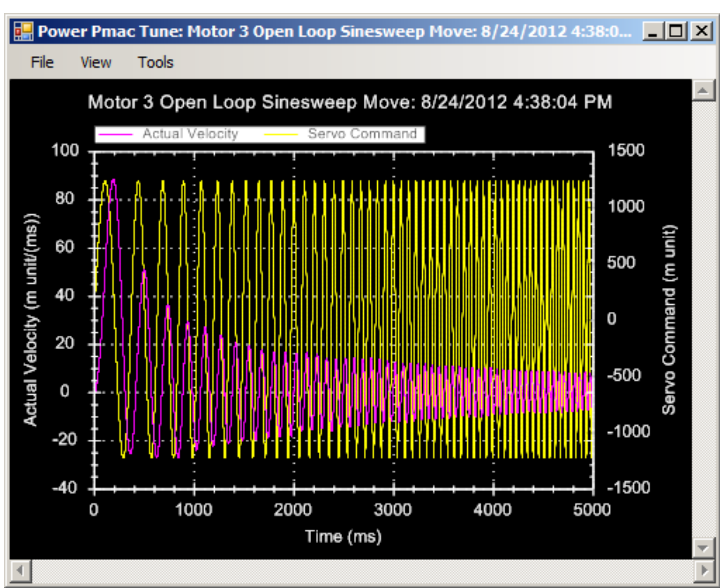
"Sweep Method" describes the manner in which to increase the frequency of the wave being commanded to the motor. Selecting Linear will increase the frequency (*f(t)* below) linearly such that the frequency change with time (*t* below) follows the following formula:

$$f(t)=((f_{end} - f_{start})/(T_{sweep}))t + f_{start},$$

where f_{end} is the "End Frequency" specified, f_{start} is the "Start Frequency" specified, and T_{sweep} is the "Sweep Time" specified. Selecting Logarithmic will increase the frequency logarithmically according to the following equation:

$$(t) = f_{start} \cdot \left(\frac{f_{end}}{f_{start}}\right)^{\frac{t}{T_{sweep}}}$$

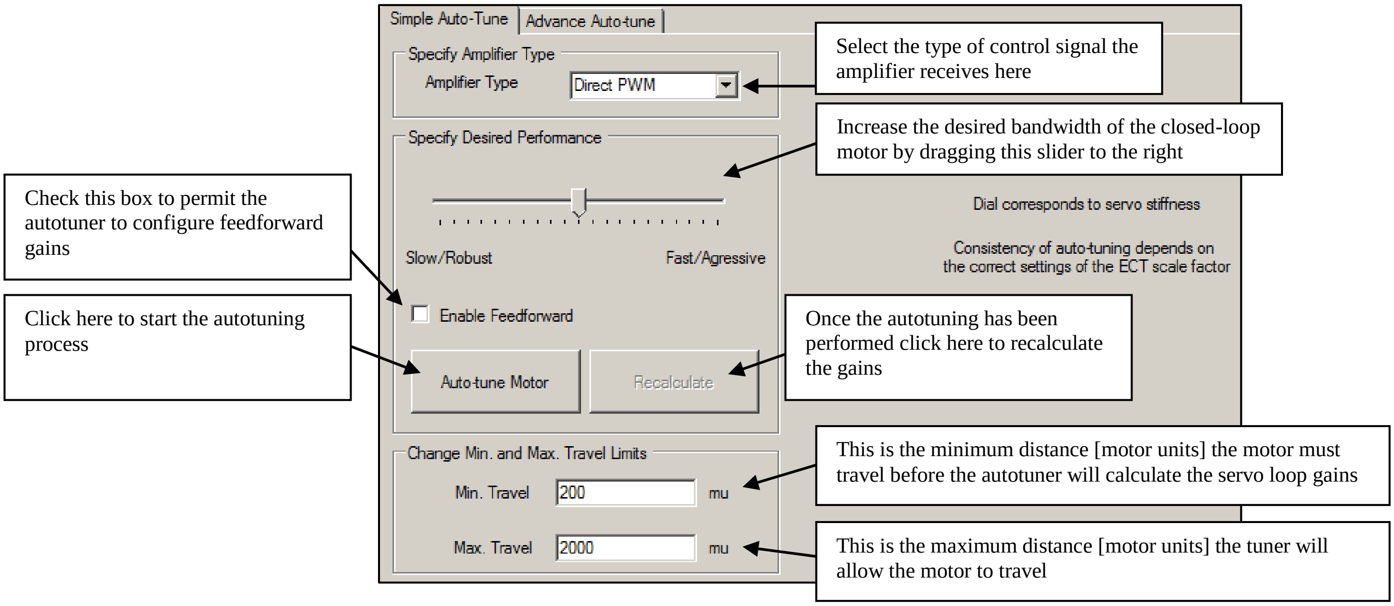
This is an example plot of a linear sweep:



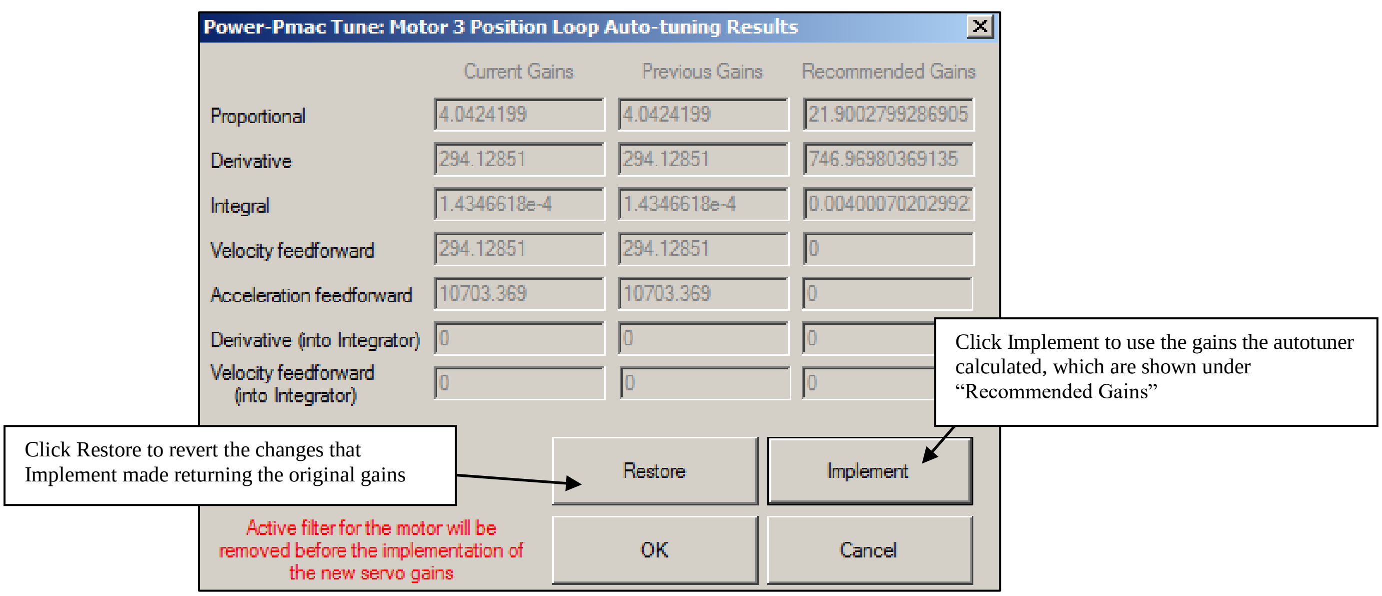
Position Loop Auto Tuning

This tab can automatically tune the motor. This is a good starting point for finding gains that can get the motor moving. It is recommended, however, to do Interactive Tuning after this in order to achieve the performance goals desired.

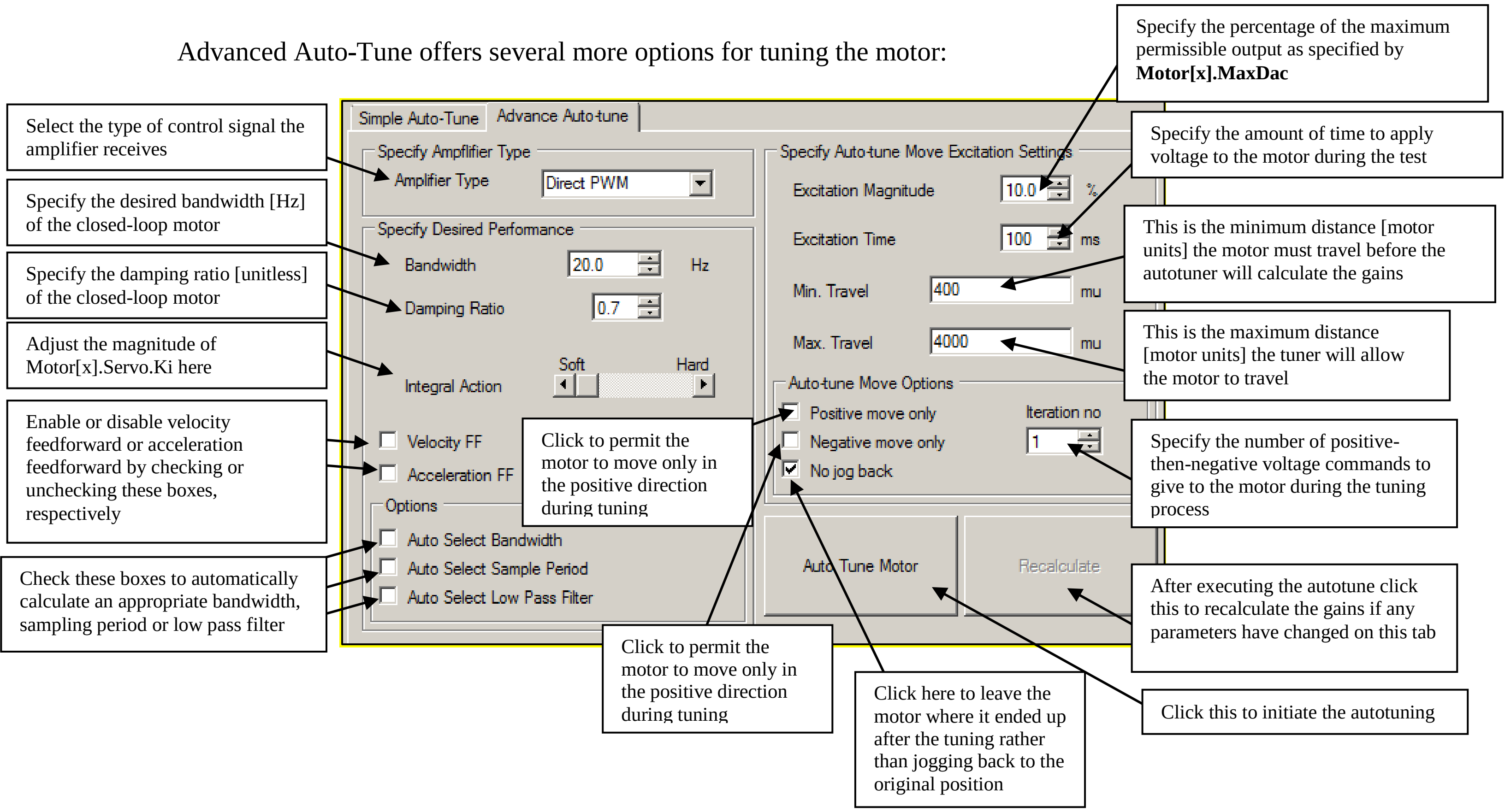
There are two sub tabs on this tab, Simple Auto-Tune and Advanced Auto-Tune. The first is Simple Auto-Tune:



After clicking Auto-Tune Motor the following screen will be displayed:

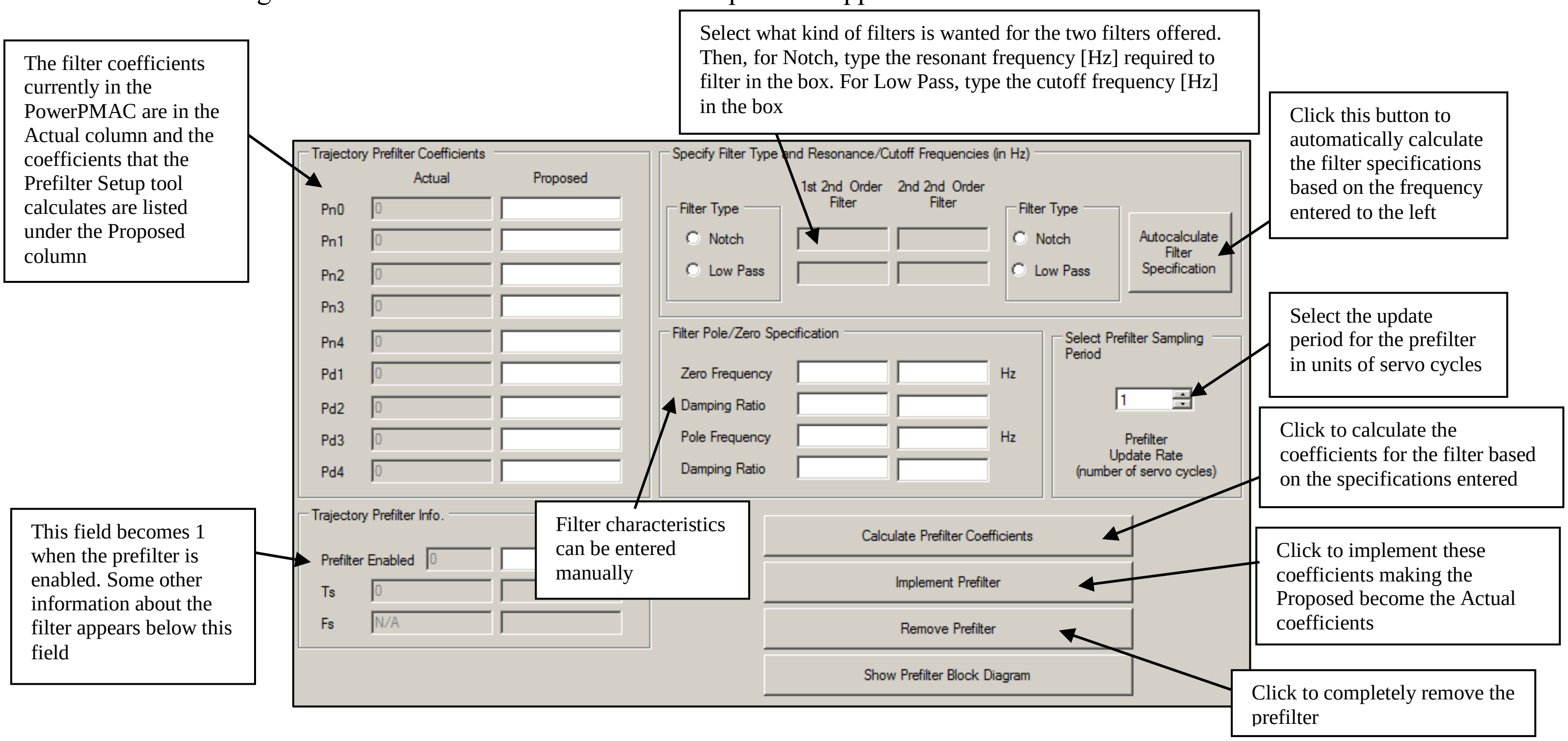


Advanced Auto-Tune offers several more options for tuning the motor:

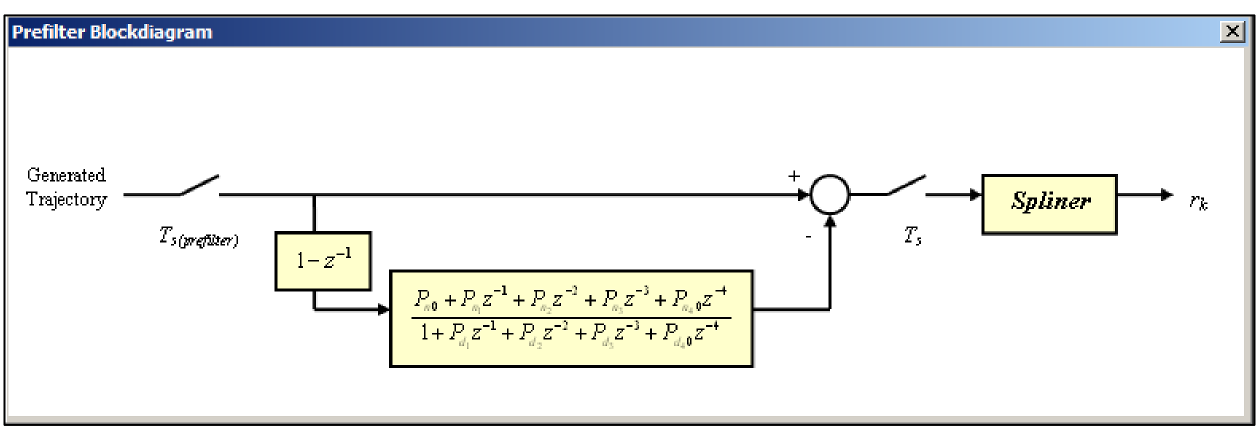


Trajectory Prefilter Setup

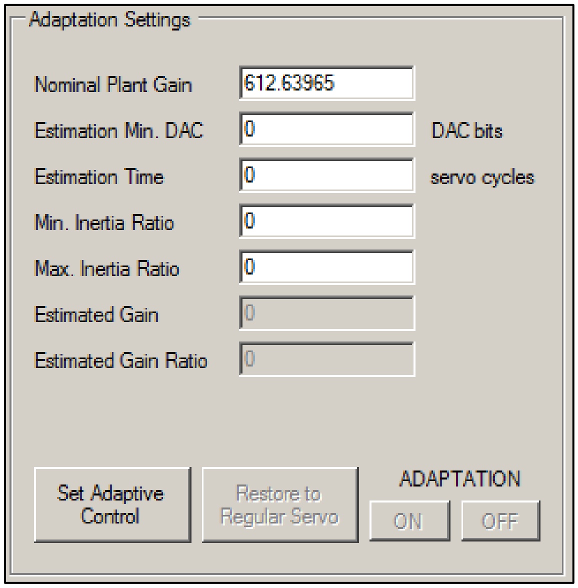
The Trajectory Prefilter Setup is used to enable the Trajectory Prefilter feature of Power PMAC and configure whether to use it as a Notch Filter, a Low Pass Filter or both as there are two filters available which can be applied to the trajectories. The Trajectory Prefilter filters any trajectory that the Power PMAC generates before commanding it to the motor in order to prevent low frequency oscillations from occurring at the machine's end effector. The Setup screen appears as follows:



Clicking "Show Prefilter Block Diagram" shows this screen demonstrating the algorithm used for the prefilter:



Adaptive Control Setup
The Adaptive Control Setup tab contains parameters related to setting up Adaptive Control:

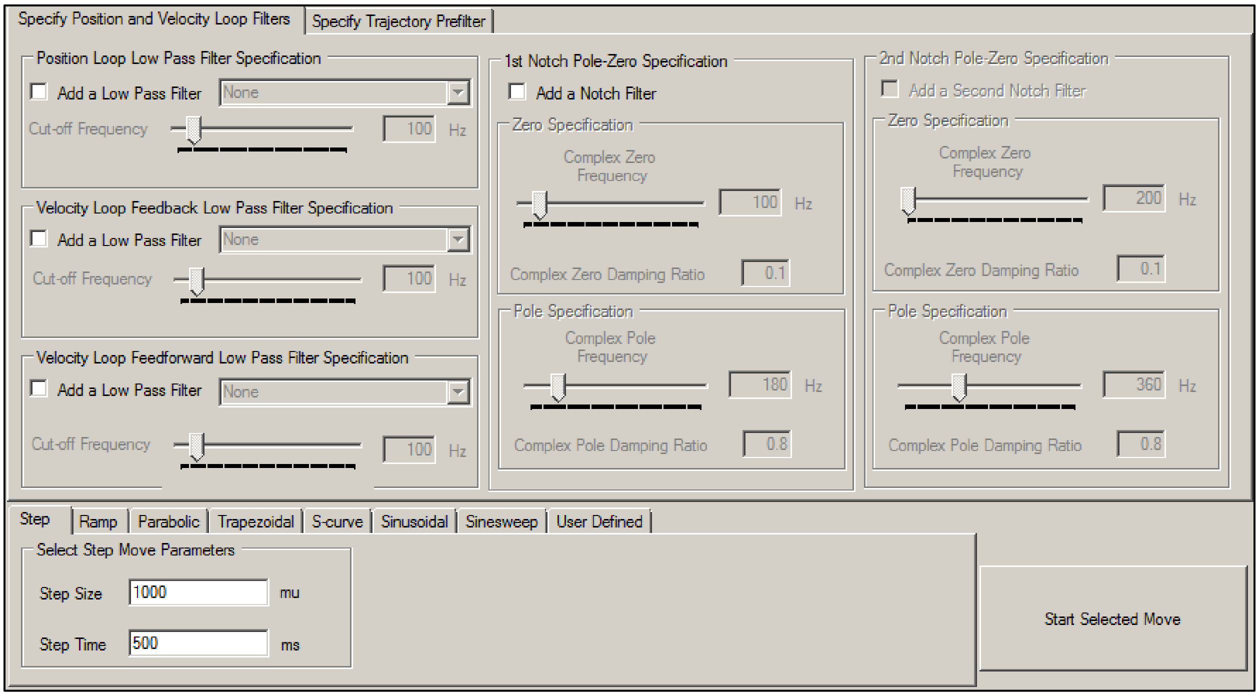


Type in the parameters required in order to configure Adaptive Control and then click "Set Adaptive Control" to enable the feature. Click "Restore to Regular Servo" to remove the feature. Click "ON" to turn the feature on, or "OFF" to turn it off.

Note

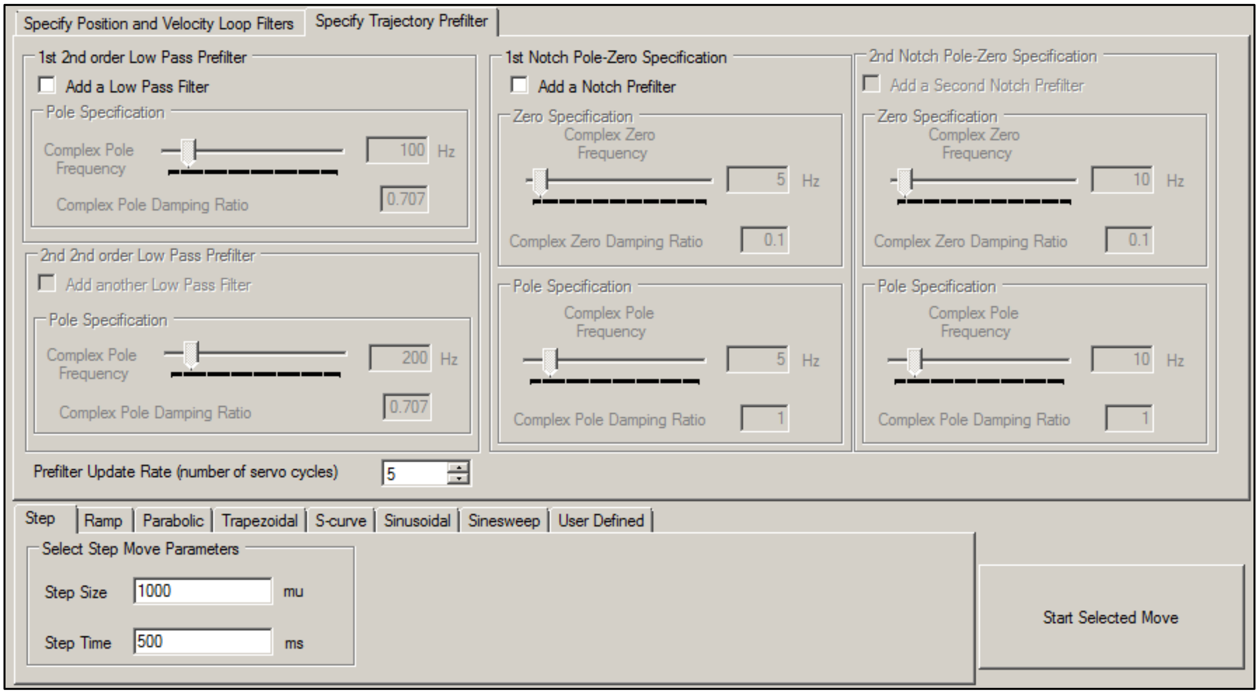
To learn about how to set these parameters properly please refer to "Adaptive Servo Control" in the Power PMAC User's Manual.

Interactive Filter Setup
Selecting the "Interactive Filter Setup" tab on the Position Loop Interactive Tuning window will open up the following screen:



On the "Specify Position and Velocity Loop Filters" tab choose which filters are to be added to the system and then select the associated parameters for those filters by either typing in the parameter or by adjusting the parameter using the slider.
Choose the various move trajectories to execute on this motor in order to test the filter. Using the Tool interactively adjust the filters and observe their effects easily and flexibly.

The “Specify Trajectory Prefilter” tab shows similar settings allowing the selection of various types of prefilters, the adjusting of their associated parameters and then the execution of moves in order to observe the effects of the filters:



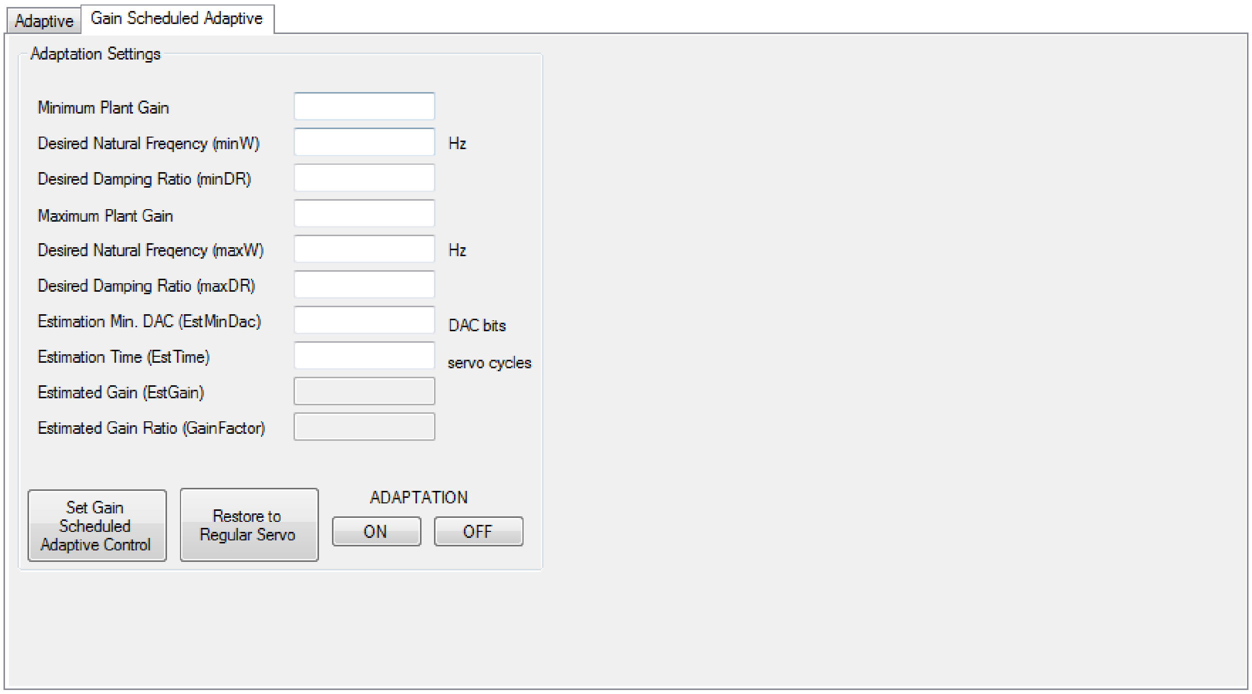
Gain-Scheduled Adaptive Control Setup

Gain Scheduled adaptive control is a variation of the adaptive control algorithm. In the standard adaptive control algorithm, the control gains are updated such that the closed loop bandwidth of the system remains the same (i.e. the same closed-loop performance) when the overall estimated gain changes (i.e. when the load changes).

In the gain-scheduled adaptive control algorithm the control gains are updated such that the closed loop bandwidth and the damping ratio change in a linear fashion depending upon the estimated gain or load changes.

The setup parameters Estimation Minimum DAC and Estimation Time are the same as in standard adaptive control. The user has to specify the minimum plant gain (i.e. at maximum inertia), the maximum plant gain (i.e. at minimum inertia), the desired bandwidth, and desired damping ratio corresponding to the two cases above.

The default tab looks like the following:



Cam Learning Control Setup

Cam learning control algorithm is a spatial, position-based, iterative control algorithm where the torque compensation table for a target motor following its source cam table is automatically filled. The control law is a proportional learning control law and is given as:

$$U_{LC}(k+1) = U_{LC}(k) + K_{LC} \cdot e(k)$$

where k is the cycle number for the cam profile, $U_{LC}(k)$ is the control effort at cycle k, K_{LC} is a proportional learning gain, and $e(k)$ is the following error at cycle k. Note that the above control law is an integrator in the cyclic base; that is, if the disturbances acting on the target motor are not time varying, it will eliminate the following errors at steady state.

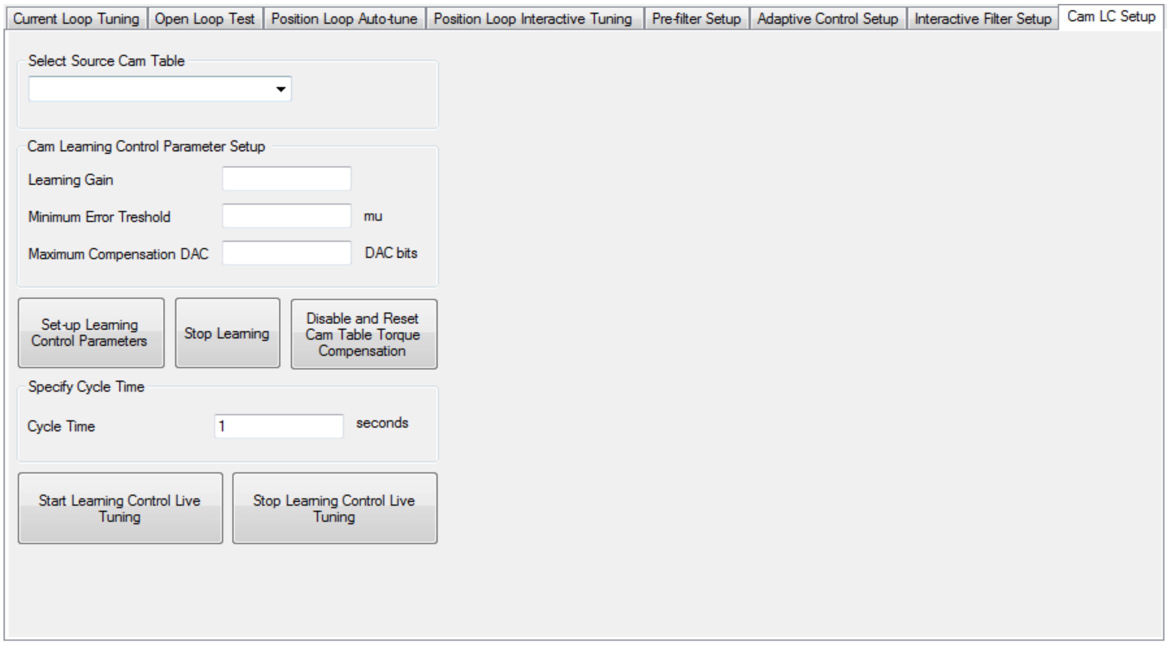
The user has to specify the source cam table, the learning gain, the minimum error in terms of motor units, and the maximum compensation torque. The software checks if there are active cam tables and populates the combo box accordingly.

The minimum error acts like a dead zone in that the torque compensation table value for a cam zone will stay the same if the following error at the specific zone is less than this value at the last iteration.

The maximum compensation DAC specifies the maximum and minimum values for the torque compensation table values.

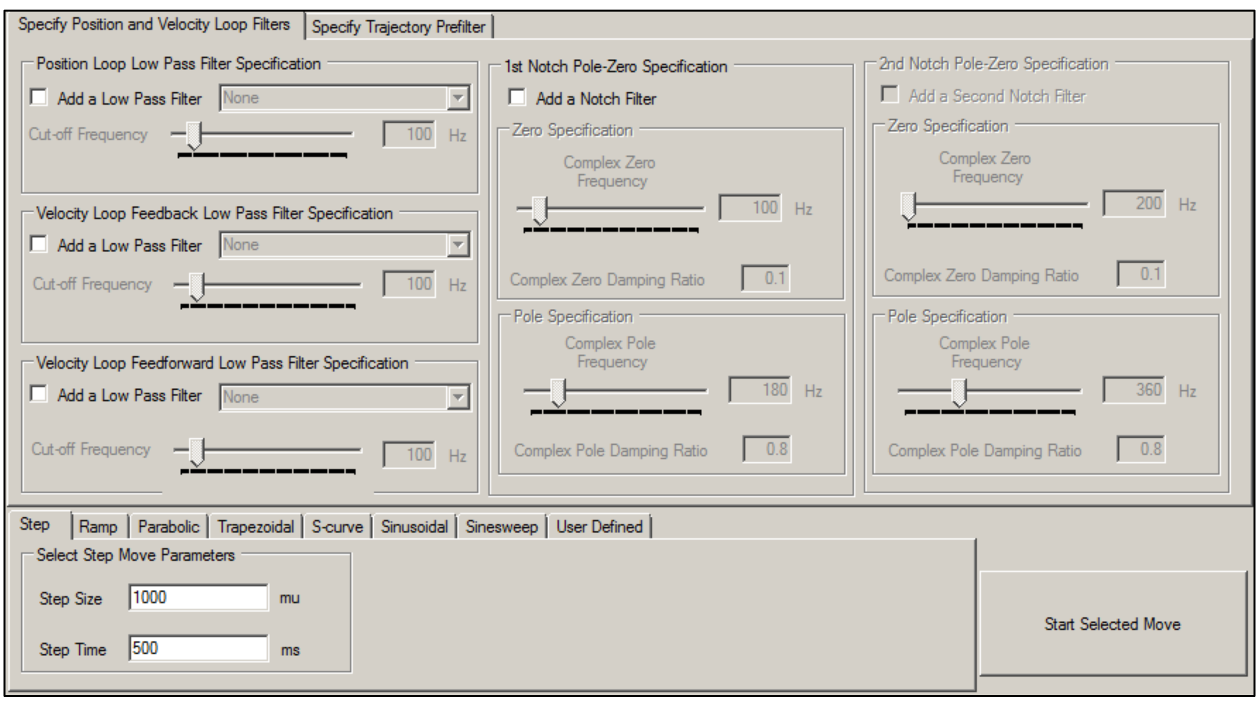
The cycle time specifies the time for the total cam profile in terms of seconds.

The live tuning feature allows the user to tune the learning gain via providing the maximum and RMS following error values for each cycle.



Interactive Filter Setup

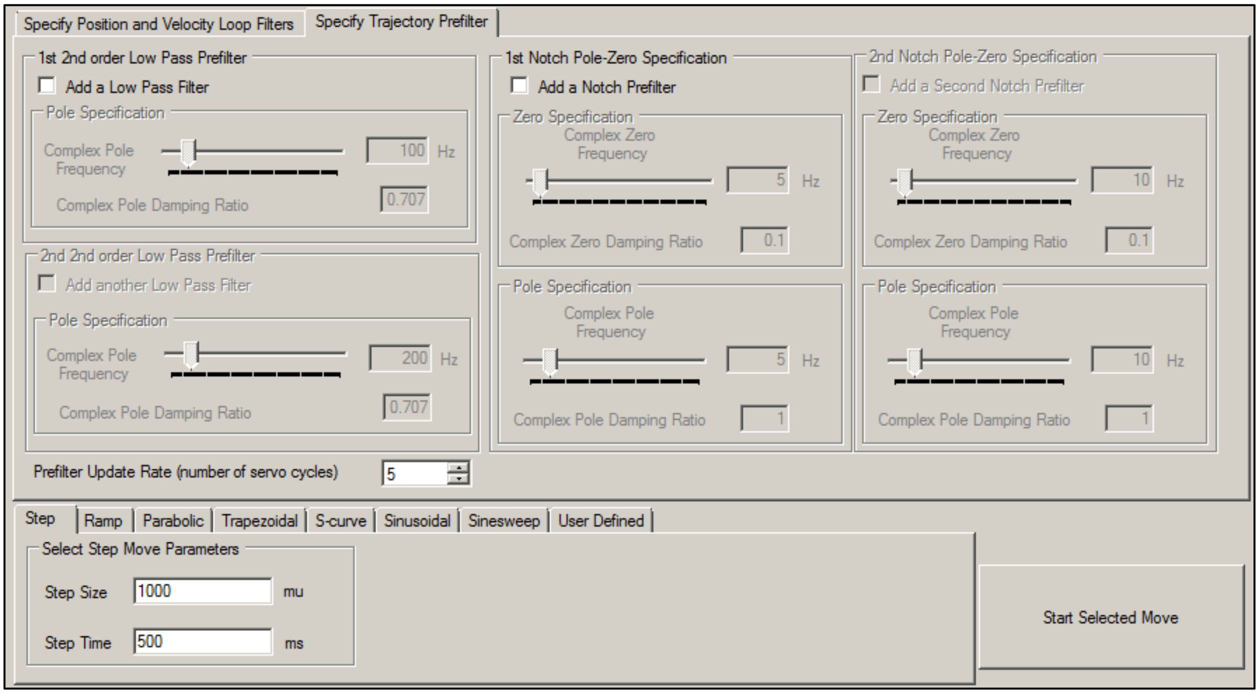
Selecting the "Interactive Filter Setup" tab on the Position Loop Interactive Tuning window will open up the following screen:



On the "Specify Position and Velocity Loop Filters" tab choose which filters are to be added to the system and then select the associated parameters for those filters by either typing in the parameter or by adjusting the parameter using the slider.

Choose the various move trajectories to execute on this motor in order to test the filter. Using the Tool interactively adjust the filters and observe their effects easily and flexibly.

The “Specify Trajectory Prefilter” tab shows similar settings allowing the selection of various types of prefilters, the adjusting of their associated parameters and then execution of moves in order to observe the effects of the filters:



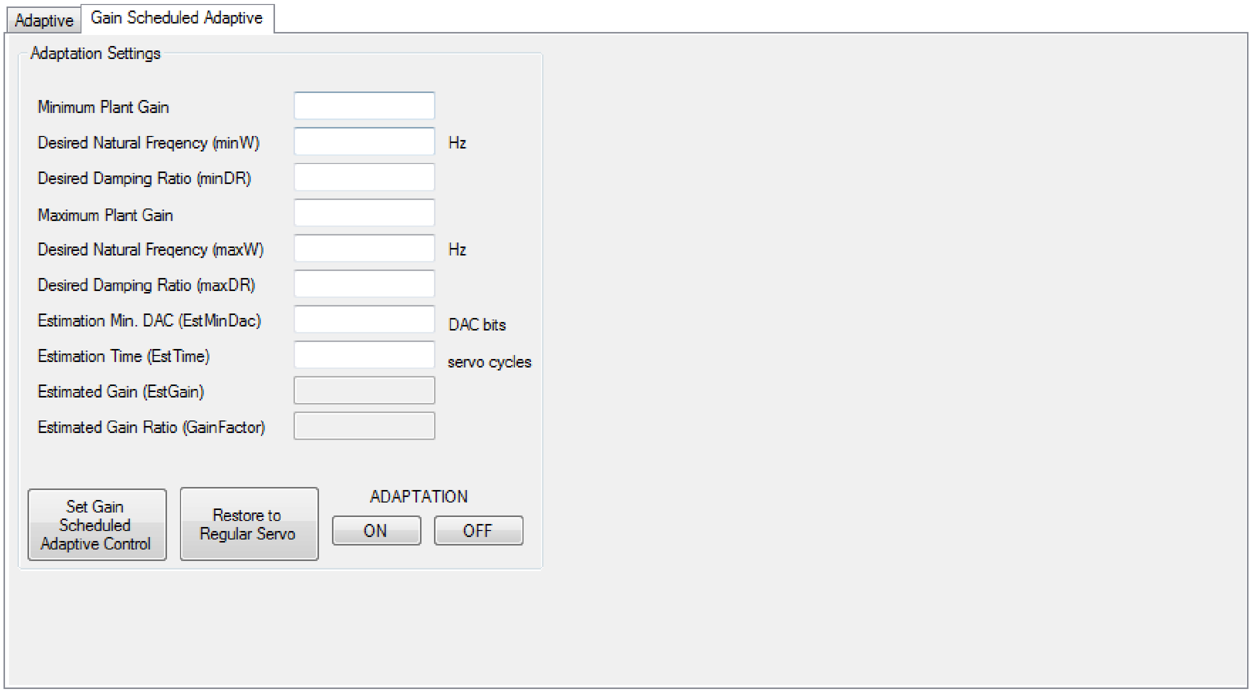
Gain-Scheduled Adaptive Control Setup

Gain Scheduled adaptive control is a variation of the adaptive control algorithm. In the standard adaptive control algorithm, the control gains are updated such that the closed loop bandwidth of the system remains the same (i.e. the same closed-loop performance) when the overall estimated gain changes (i.e. when the load changes).

In the gain-scheduled adaptive control algorithm on the other hand, the control gains are updated such that the closed loop bandwidth and the damping ratio change in a linear fashion depending upon the estimated gain or load changes.

The setup parameters Estimation Minimum DAC and Estimation Time are the same as in standard adaptive control. The user has to specify the minimum plant gain (i.e. at maximum inertia), the maximum plant gain (i.e. at minimum inertia), the desired bandwidth, and desired damping ratio corresponding to the two cases above.

The default tab looks like the following:



Cam Learning Control Setup

Cam learning control algorithm is a spatial (position-based) iterative control algorithm where the torque compensation table for a target motor following its source cam table is automatically filled. The control law is a proportional learning control law and is given as:

$$U_{LC}(k+1) = U_{LC}(k) + K_{LC} \cdot e(k)$$

where k is the cycle number for the cam profile, $U_{LC}(k)$ is the control effort at cycle k, K_{LC} is a proportional learning gain, and $e(k)$ is the following error at cycle k. Note that the above control law is an integrator in the cyclic base; that is, if the disturbances acting on the target motor are not time varying, it will eliminate the following errors at steady state.

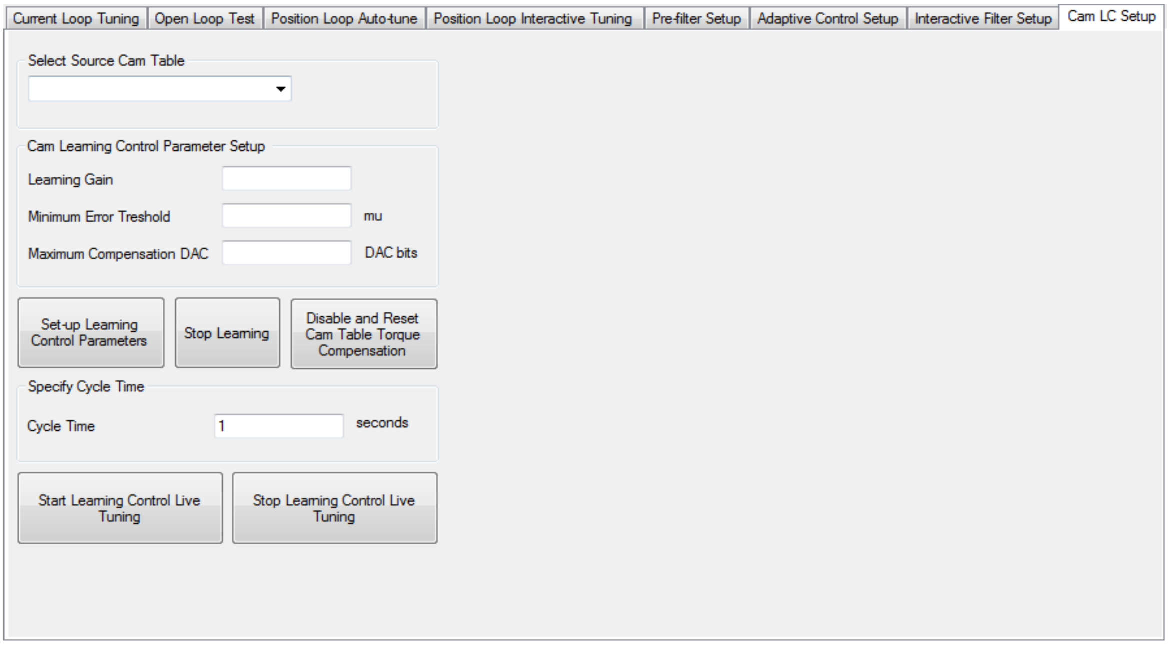
The user has to specify the source cam table, the learning gain, the minimum error in terms of motor units, and the maximum compensation torque. The software checks if there are active cam tables and populates the combo box accordingly.

The minimum error acts like a dead zone in that the torque compensation table value for a cam zone will stay the same if the following error at the specific zone is less than this value at the last iteration.

The maximum compensation DAC specifies the maximum and minimum values for the torque compensation table values.

The cycle time specifies the time for the total cam profile in terms of seconds.

The live tuning feature allows the user to tune the learning gain via providing the maximum and RMS following error values for each cycle.

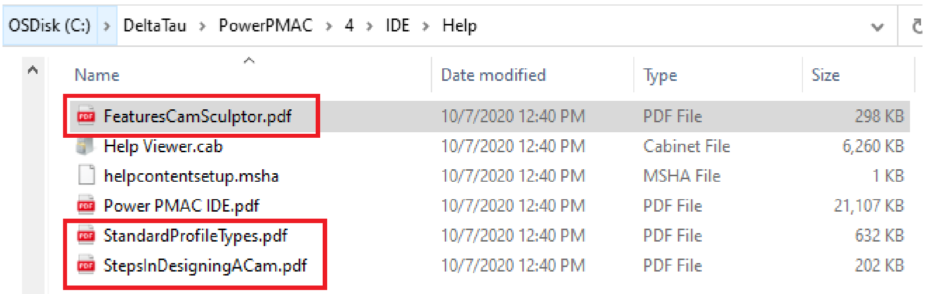


Kill Motors

This menu option kills all motors. This is equivalent to issuing a CTRL+ALT+K command in the Terminal window.

CAM Sculptor

This software feature is licensed. This option will allow user to define CAM but if the software is not licensed it will not allow to download the CAM profiles to Power PMAC.
The help for this menu item is separate and available in the Power PMAC IDE installation under Help folder. On a standard installation it is available…



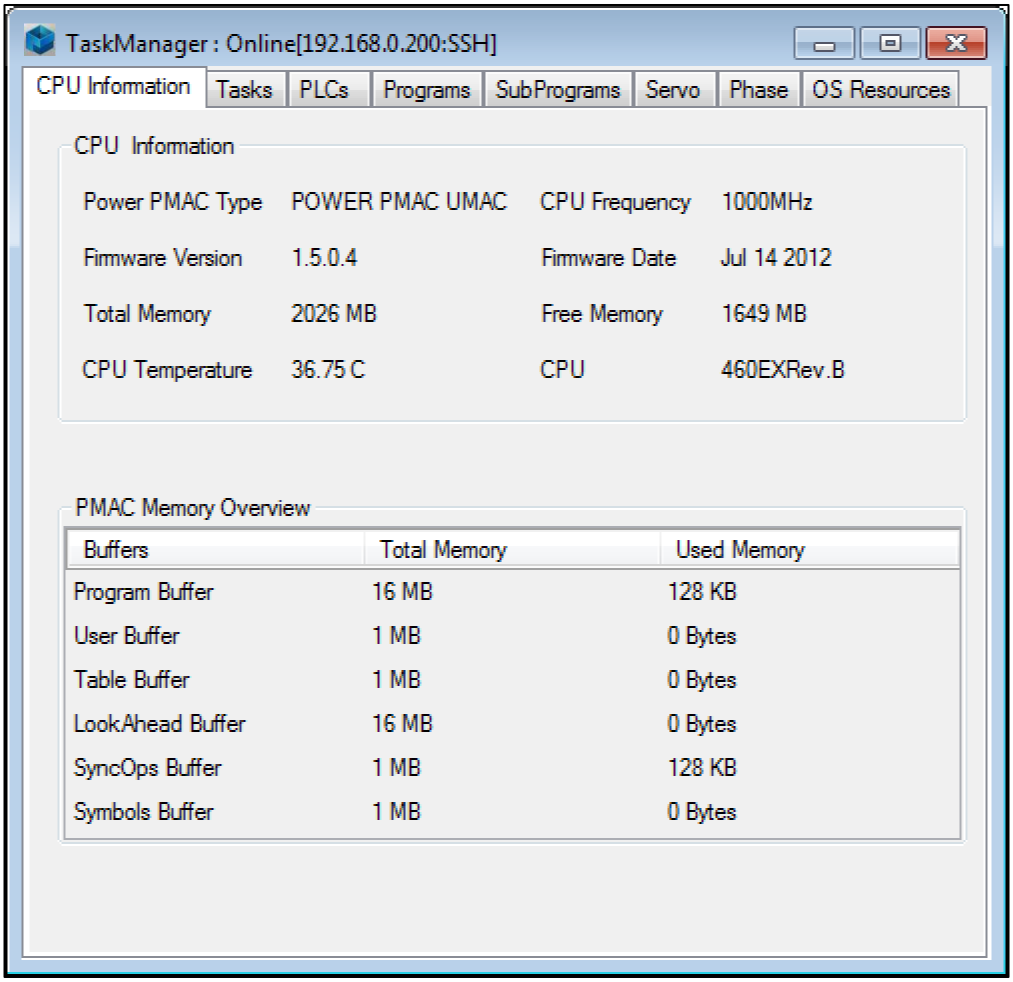
Task Manager

The Task Manager:

- Provides information about the Power PMAC CPU and about programs running thereon
- Permits the start and stop of programs
- Displays which servo and phase algorithms the motors used

CPU Information

The first tab of the Task Manager is the CPU Information tab:


TaskManager : Online[192.168.0.200:SSH]
CPU Information
Tasks
PLCs
Programs
SubPrograms
Servo
Phase
OS Resources
CPU Information
Power PMAC Type
POWER PMAC UMAC
CPU Frequency
1000MHz
Firmware Version
1.5.0.4
Firmware Date
Jul 14 2012
Total Memory
2026 MB
Free Memory
1649 MB
CPU Temperature
36.75 C
CPU
460EXRev.B
PMAC Memory Overview
Buffers
Total Memory
Used Memory
Program Buffer
16 MB
128 KB
User Buffer
1 MB
0 Bytes
Table Buffer
1 MB
0 Bytes
LookAhead Buffer
16 MB
0 Bytes
SyncOps Buffer
1 MB
128 KB
Symbols Buffer
1 MB
0 Bytes

The table below describes the fields beneath “CPU Information:”

Field	Description
Power PMAC Type	This field states the type of Power PMAC form factor in which this CPU resides (e.g. UMAC, Brick, etc.)
Firmware Version	The version number of the firmware installed on this Power PMAC CPU
Total Memory	The total RAM with which this CPU is equipped
CPU Temperature	The present operating temperature of this CPU in degrees Celsius
CPU Frequency	The frequency at which this CPU is clocked in MHz
Firmware Date	The date of the build of the firmware installed on this CPU
Free Memory	The amount of RAM presently unused
CPU	The PowerPC CPU’s revision in this Power PMAC

The next section of this tab is the “PMAC Memory Overview”. In this section there are three columns: Buffer, Total Memory and Used Memory whose purpose is as follows:

- The buffer column describes each buffer in the Power PMAC memory
- The Total Memory column describes how much total memory space is allocated for that buffer
- The Used Memory column describes how much that Total Memory is actually being used or occupied presently

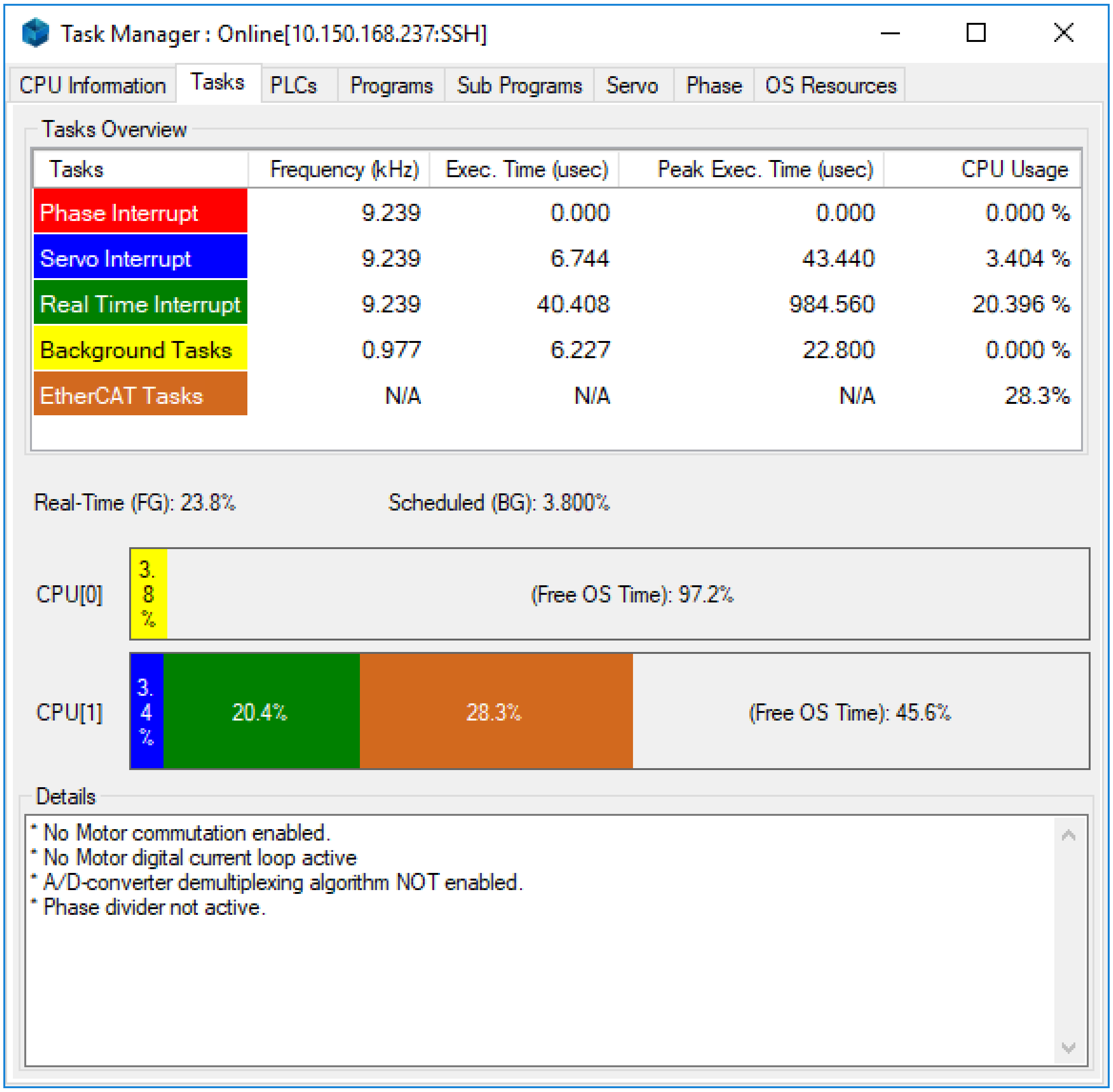
The table below describes each buffer beneath “PMAC Memory Overview” in the Buffer column:

Buffer	Description
Program Buffer	Allocates space for motion programs and PLC programs written in Script
User Buffer	Allocates space for general purpose use
Table Buffer	Allocates space for compensation tables (position and torque)
LookAhead Buffer	Allocates space for the Special Lookahead feature
SyncOps Buffer	Allocates space for Synchronous Operations (i.e. Synchronous M-Variables)
Symbols Buffer	Allocates space for variable names

The exact amount of memory allocated for each buffer can been seen by typing the **size** command into the Terminal Window and the exact amount of free memory within those buffers with the **free** command.

Tasks

The Tasks tab shows five categories of tasks being executed on the Power PMAC CPU:



The purpose of each column shown in the Tasks tab is described below:

Column Name	**Description**
Tasks	Lists the task whose properties are being described in the columns to this right of this one
Frequency	The frequency with which this task occurs
Calculation Time	The average time this task requires to finish
Peak Time	The largest measured amount of time this task has taken to finish since startup
% Task Time	The percentage of total CPU time this task consumes on average

The tasks in the Tasks column are described below:

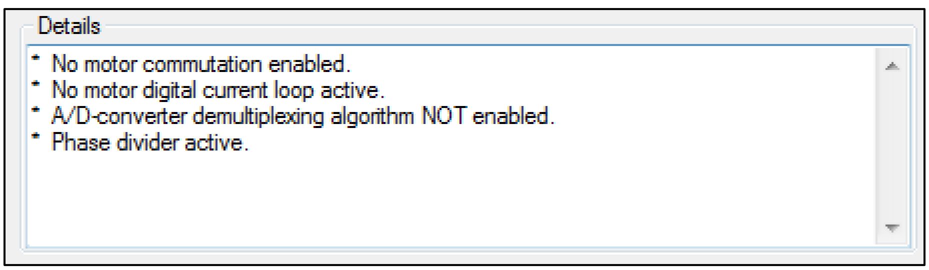
Task Name	Type of Calculations Performed Within This Task
Phase Interrupt	Phase algorithms, typically used for commutating motors
Servo Interrupt	Servo algorithms, typically used for servo control of motors
Real Time Interrupt	Move planning, real time Script and C PLCs
Background Tasks	Background Script and C PLCs, Background C Applications, Watchdog Timer Resetting, Checking Limits and Safety Features, Communicating with Host Computer
EtherCAT Tasks	Amount of time taken for EtherCAT task.

Clicking each task in the Task column will show details about that task in the Details box at the bottom of the Task Manager window.

Clicking on Phase Interrupt will show:

- How many motors are being commutated
- How many digital current loops are active
- Whether the A/D converter demultiplexing algorithm is enabled
- Whether the phase divider is active

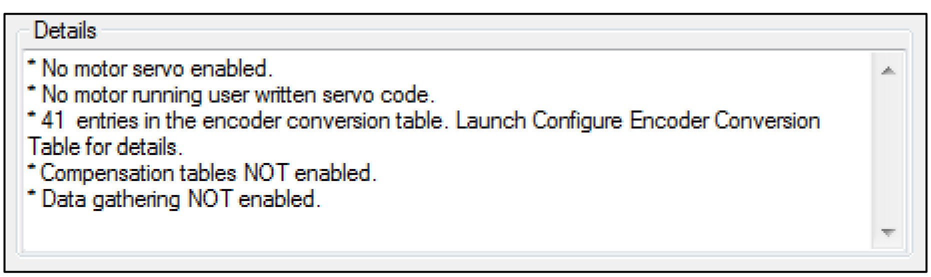
Example “Details” Contents for the Phase Interrupt task:



Clicking on Servo Interrupt will show:

- How many motors’ servo control is enabled
- How many motors are using user-written servo code
- How many entries are in the Encoder Conversion Table
- How many compensation tables are enabled
- Whether data gathering is enabled

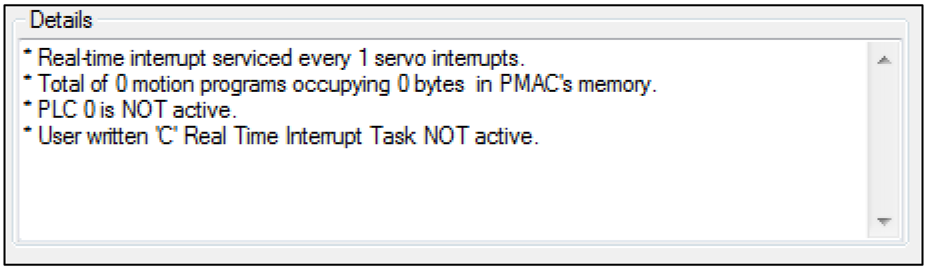
Example “Details” Contents for the Servo Interrupt task:



Clicking on Real Time Interrupt will show:

- How often the Real-Time Interrupt (RTI) is serviced
- How many motion programs occupy how much space of the Power PMAC's memory
- Whether the Real-Time PLC (PLC 0) is active
- Whether the user-written Real-Time Interrupt C Program (RTICPLC) is active:

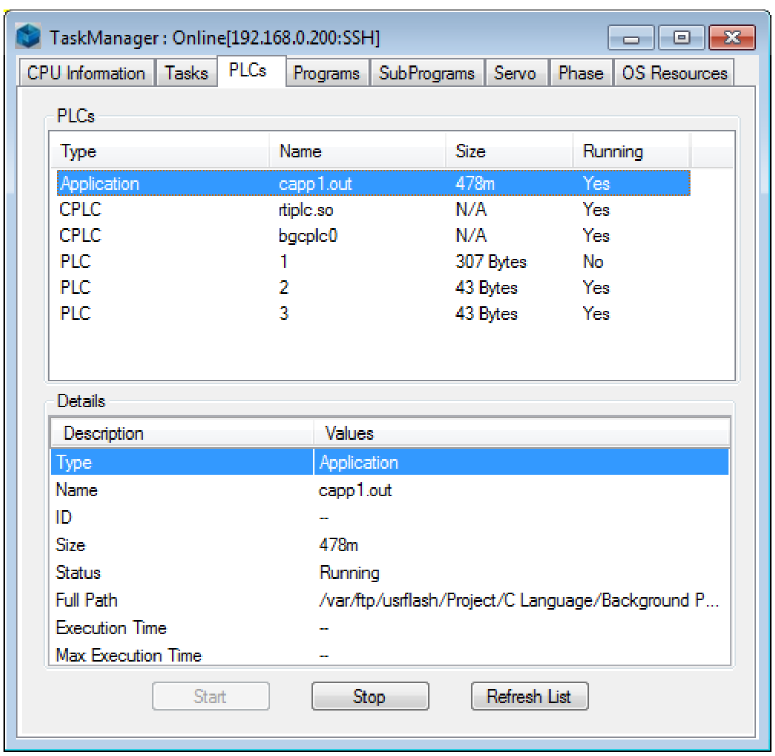
Example "Details" Contents for the Real-Time Interrupt task:



Clicking on Background Tasks shows nothing.

PLCs

Clicking the PLCs tab lists all Background C Applications, Script PLCs, Real-Time C PLCs (RTICPLCs), and Background C PLCs (BGCPLCs) running on the Power PMAC presently:



In the "PLCs" box there are four columns:

- The Type column shows the type of the program
- The Name column shows the name (if it has been named) or number of the program
- The Size column shows the amount of RAM the program occupies

- The Running column shows whether the program is running presently

The “Details” box at the bottom of the window shows various properties about the program. The table below describes these properties:

Detail Name	Description
Type	The type of program this is
Name	The name of the program
ID	The ID number of the program, if it has one
Size	The amount of RAM this program occupies
Status	Shows whether the program is running or not
Full Path	For C programs only; describes the directory path for the executable file
Execution Time	The time the program takes to finish executing each time it executes
Max Execution Time	The longest amount of time this program has taken to run since startup

The user can start a program by clicking on the program in the list and then clicking on the 

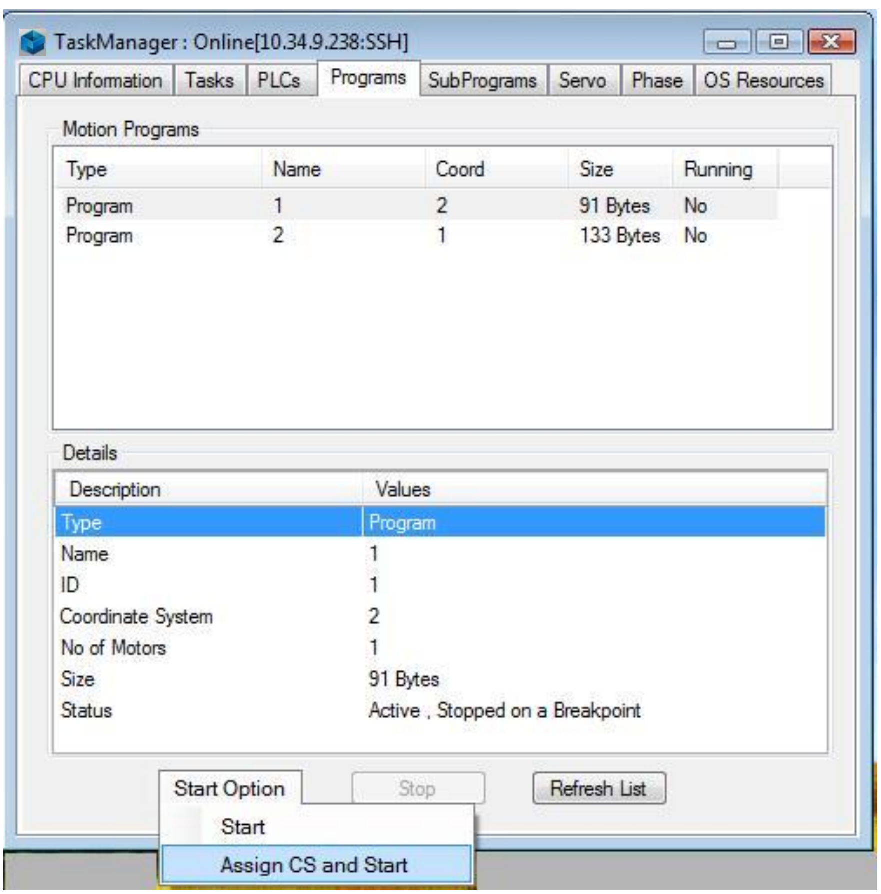
button, or stop the program by clicking 

. To refresh the list of programs, press 

.

Programs

The programs tab lists all motion programs in the Power PMAC:



In the “Motion Programs” box there are five columns:

- The Type column shows the type of the program
- The Name column shows the name (if it has been named) or number of the program
- The Coord column shows the number(s) of the coordinate system(s) which is/are presently running this program (more than one coordinate system can be running the same program simultaneously)
- The Size column shows the amount of RAM the program occupies
- The Running column shows whether the program is running presently

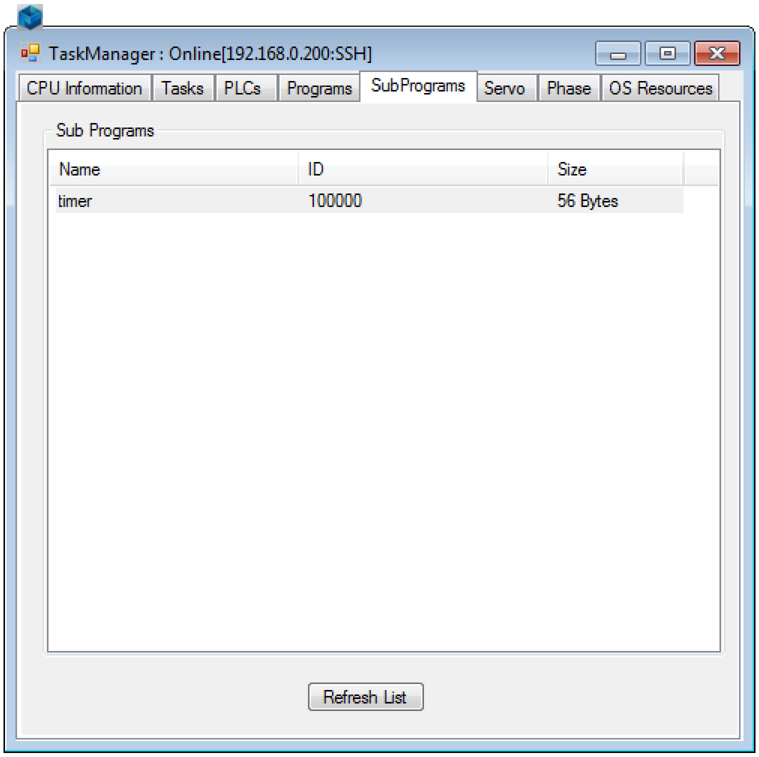
In the “Details” box there are 7 properties of the motion programs:

Detail Name	**Description**
Type	The type of the program
Name	The name of the program if it has been named, or the number if not
ID	The program ID number
Coordinate System	The coordinate system in which this program is presently running
No of Motor	Number of motors in this coordinate system
Size	The amount of RAM this program occupies
Status	Shows whether this program is running or not

Start a program by clicking on the program in the list, clicking on the Start Option menu and then selecting Start. This option will be grayed out if the coordinate system (CS) column for the selected row displays “Not Assigned.”
In this case use the second menu option “Assign CS and Start”. Selecting this menu will show a dialog box where a coordinate system can be specified to start the program. A coordinate system number can be entered or for multiple entries the numbers should be separated by comma’s. Stop the program by clicking Stop. To refresh the list of programs, press Refresh List.

SubPrograms

The SubPrograms tab shows all subprograms in the Power PMAC:



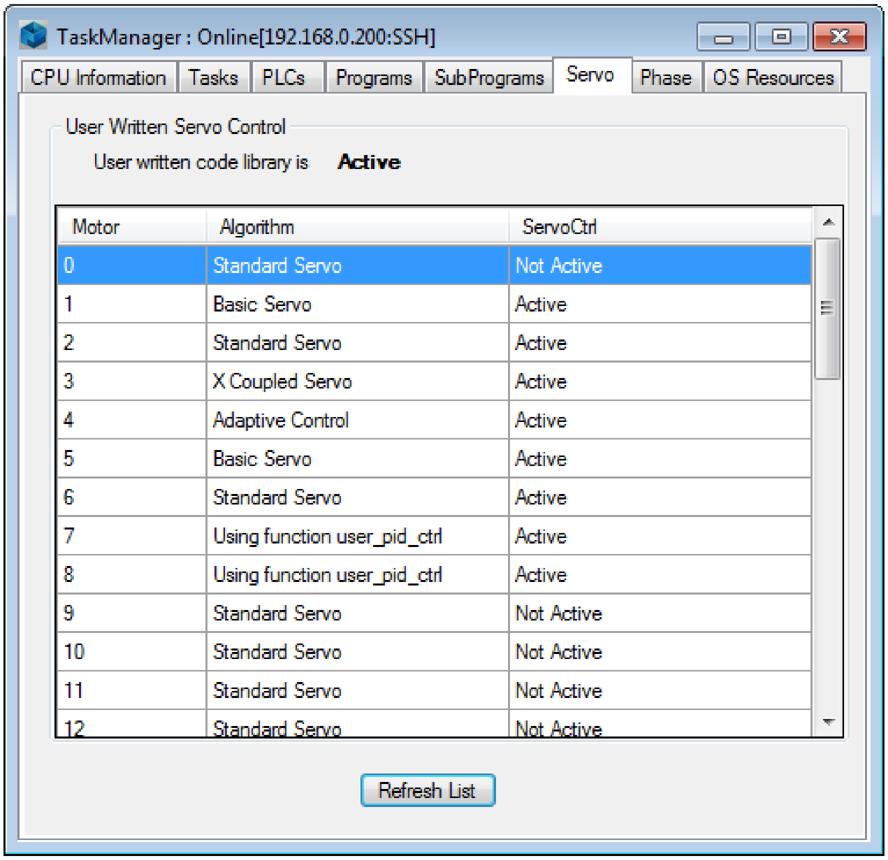
There are three columns in the SubPrograms tab:

- The Name column shows the name of the subprogram or it's number if it has no name
- The ID column shows the ID of the subprogram which the IDE has assigned it
- The Size column shows how much RAM this subprogram occupies

To refresh the list of subprograms, press Refresh List.

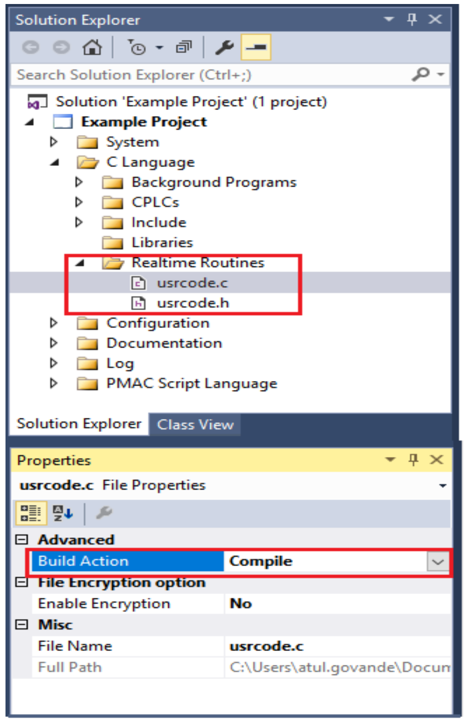
Servo

The Servo tab shows which servo algorithms are being used for which motors:



This tab shows whether the user-written code library is active; that is, whether the user is using any real-time C routines in the IDE project.

The IDE recognizes that the user is using real-time C routines if the build action on usrcode.c (in the IDE's Solution Explorer under C Language→Realtime Routines) is set to Compile, as shown in the screenshot below:

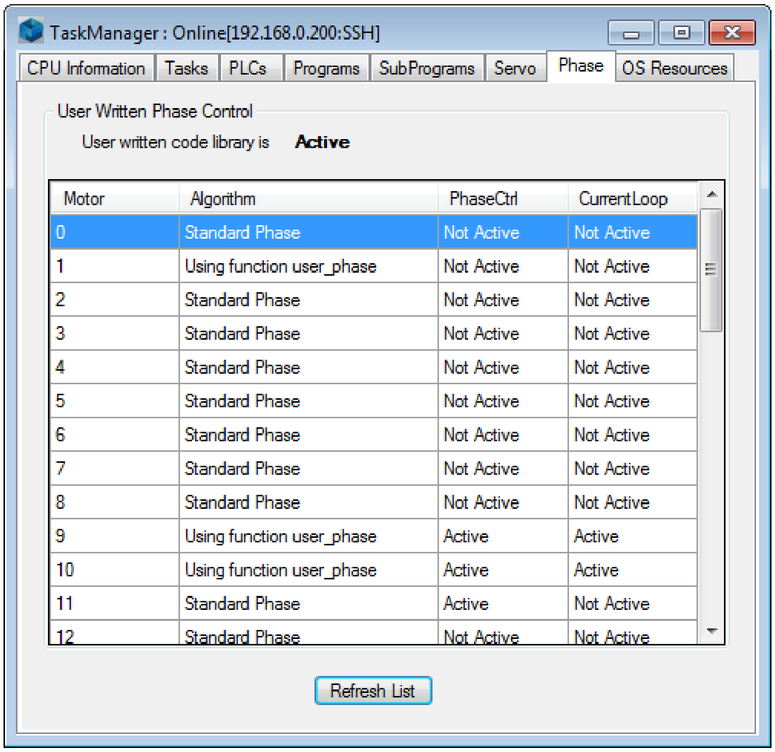


There are three columns on the Servo tab:

- The "Motor" column shows each motor number, ranging from 0 to the value of (**Sys.MaxMotors** - 1)
- The "Algorithm" column shows which servo algorithm is being used for this motor. The servo algorithms available are as follows:
    - "Standard Servo": This is the Power PMAC's standard, default servo algorithm; basically PID with some filters, saturations, and deadbands
    - "Basic Servo": This is just a standard PID servo algorithm with no additional filters and nonlinearities
    - "X Coupled Servo": This is the cross-coupled gantry servo algorithm
    - "Adaptive Control": This is the adaptive control algorithm
    - "Using function *{user function here}*": This is the user-written servo algorithm, which the user needs to have written in C code and then set this motor to use that algorithm. In the example screenshot above the servo algorithm is named "user_pid_ctrl." See the "Configuring User-Written Servo Algorithms" section of this manual under the "Project System" header for more details on configuring user-written servo algorithms.
- The "ServoCtrl" column shows whether this motor is active
-

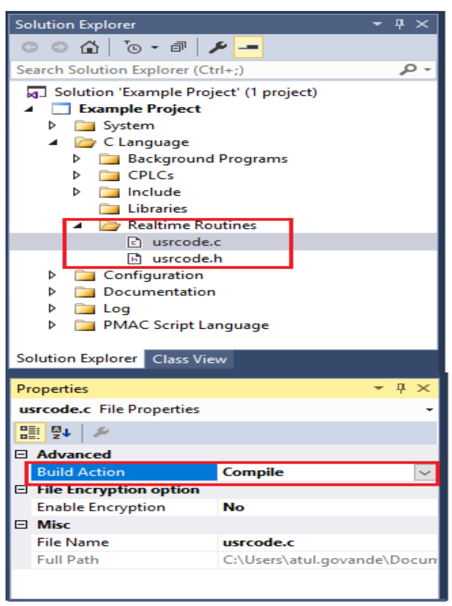
Phase

The Servo tab shows which servo algorithms are being used for which motors:



This tab shows whether the user-written code library is active; that is, whether the user is using any real-time C routines in the IDE project.

The IDE recognizes that real-time C routines are being used if the build action on usrcode.c (in the IDE's Solution Explorer under C Language→Realtime Routines) is set to Compile as shown in the screenshot below:

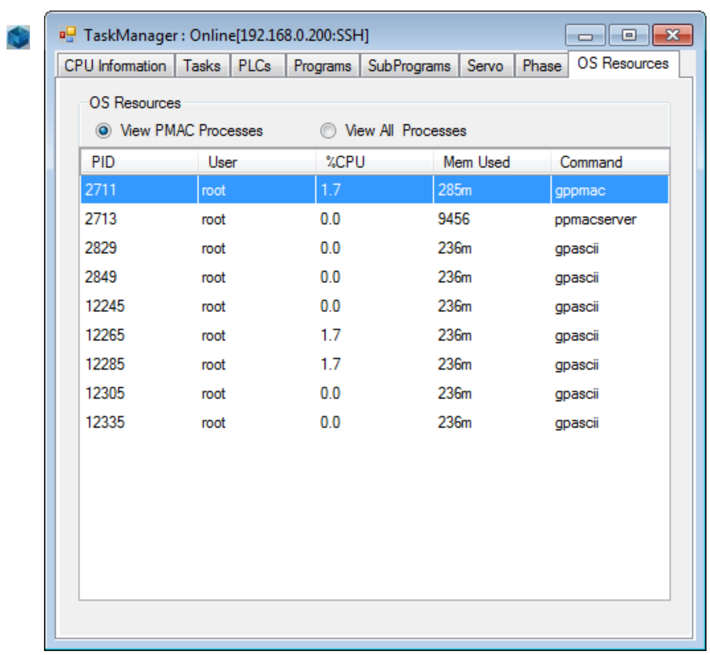


There are four columns on the Phase tab:

- The "Motor" column shows each motor number ranging from 0 to the value of (**Sys.MaxMotors** - 1)
- The "Algorithm" column shows which phase algorithm is being used for this motor. The phase algorithms available are as follows:
    - "Standard Phase": the Power PMAC's standard, default motor commutation algorithm
    - "Using function *{user function here}*": This is the user-written phasealgorithm, which the user needs to have written in C code and then set this motor to use that algorithm. In the example screenshot above the servo algorithm is named "user_phase." See the "Configuring User-Written Phase Algorithms" section of this manual under the "Project System" header for more details on configuring user-written phase algorithms.
- The "PhaseCtrl" column shows whether this motor is commutated
- The "CurrentLoop" column shows whether the Power PMAC is closing a digital current loop for this motor

OS Resources

The OS Resources tab shows all of the processes (also known as threads) running on the Power PMAC. Choose either to show only the Power PMAC processes (i.e. processes related to the Power PMAC's tasks listed in the Tasks tab) or all processes, including processes that may be running in the background and are not part of the Power PMAC's tasks:



There are five columns on the "OS Resources" (Operating System Resources) tab:

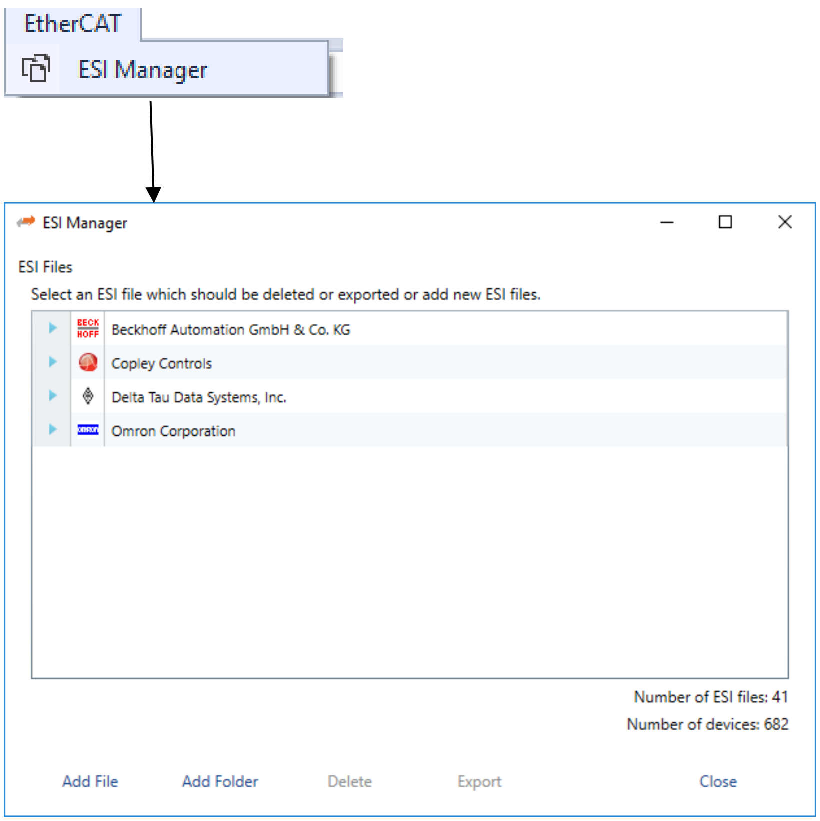
- The "PID" column shows the Process ID number for this thread
- The "User" column shows with which user this process is associated
- The "%CPU" column shows what percentage of the CPU's time this process consumes
- The "Mem Used" column shows how much RAM this thread consumes.
- The "Command" column shows the name of the Process; that is, the name of the function which this thread is executing

Note

The amount of RAM used is shown in the “Mem Used” column consists of the sum of the actual RAM the program occupies, and the shared memory shared between each program using Delta Tau’s C Libraries. Thus, for example in the screenshot above, seeing “236m” for several programs does not mean that each one occupies 236 MB but rather that they all are roughly the same size and share the same library space, bringing their total up to 236 MB.

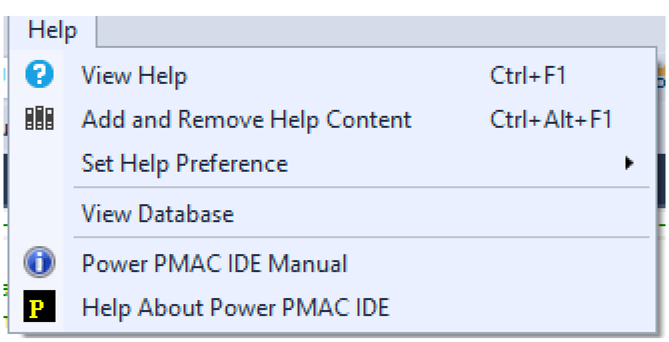
EtherCAT

This menu allows the loading and management of the device ESI files or ESI folder.



Help

The Help menu provides a submenu for help on the IDE.



View Database: The database is primarily used for setup program, intellisense, etc.

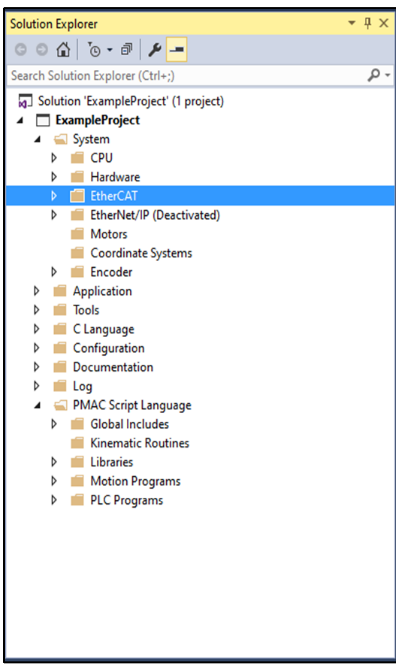
PROJECT SYSTEM

IDE 4.x primary focus is on handling everything from the project system. The Project System is far more powerful as compared to V2.x and V3.x.
The Project System incorporates the entire system setup, ECAT setup and Setup Variable. The Project System maintains all the saved structure elements as changes are made within project domain. The Build and download generates a systemsetup.cfg file that contains all the user settings removing the necessity of the backup of the Power PMAC manually.
The Project System integrates the EtherCAT setup (EC-Engineer) removing the need to use an external program to setup the network.

Project Organization

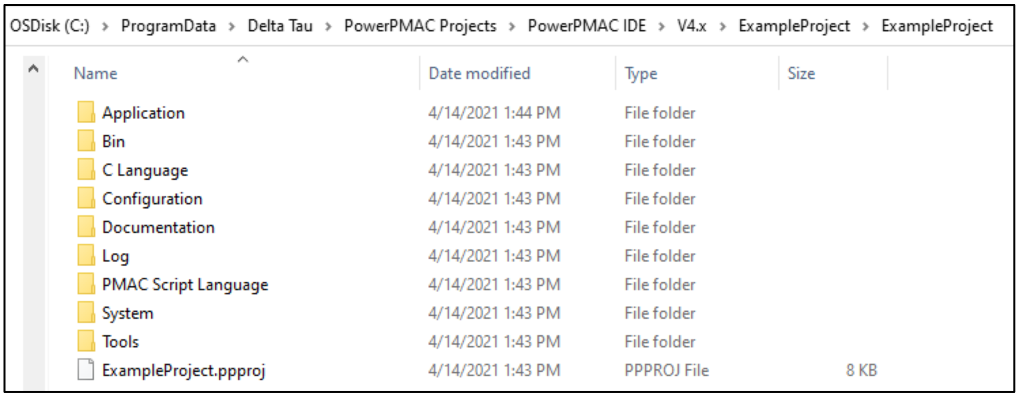
Layout

Projects in the Power PMAC IDE are organized into a folder structure which can be navigated within the Solution Explorer which, by default, is the farthest right window in the IDE. This can be opened by clicking View→Solution Explorer from the main IDE screen or pressing CTRL+ALT+L.
The Explorer appears as follows:



- System: Folder allows to configure, set system settings
- CPU: Global Clock and common CPU settings including core management for ARM cpu.
- Hardware: Displays detected hardware .
- EtherCAT: Allows EtherCAT network setup.
- EtherNetIP: Allows Ethernet IP network setup
- Motors: Allows to Add motor and configure motor
- Coordinate Systems: Allows configure coordinate system and it's structure elements.
- Encoder: Maintains current encoder table settings to be used in build and download.
- Application: User interface for configuring application setup.
- Tools: Integrated New Tuning interface
- C Language: C language programming files.
- Background Programs: This folder contains Background C programs and associated header files.
- CPLCs: This folder contains all Background C PLCs (BGCPLCs)
- Include: Folder includes all the header files required for C programs
- Libraries: Common libraries that you add or develop and use it in the C app.
- RealTime Routines: this folder contains code for user written servo and phase routines.
- Configuration: This folder contains files you can use to manage what runs first at startup and what runs upon download
- Documentation: stores General-purpose documentation files for the project in this folder.
- Log: This folder contains detailed log files that can be view in case of the build and download failure
- PMAC script Language: This folder contains all script programs, PLC and associated header files.
- Global Includes: Stores header files used in script programming.
- Kinematic Routines: This folder contains script kinematics subroutine files.
- Libraries: Folder stores script subprograms files that can be called from Motion program.
- Motion Programs: This folder contains script Motion programs files.
- PLC Programs: This folder contains Script PLC files.

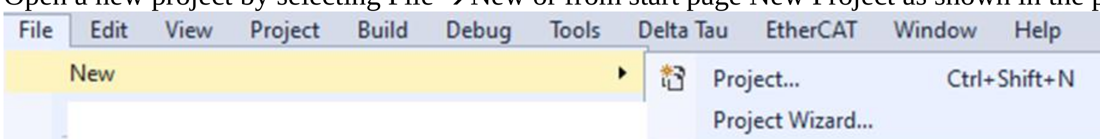
On the PC the project file organization looks like this…

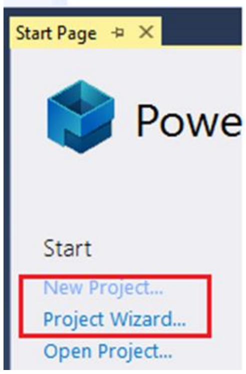


Opening a Project

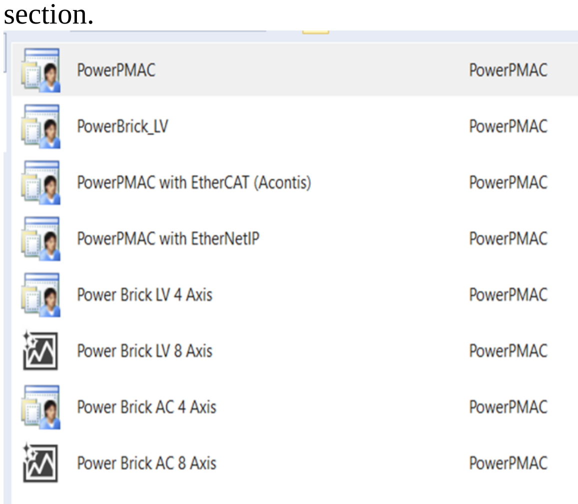
File-New-Project

Open a new project by selecting File →New or from start page New Project as shown in the picture.





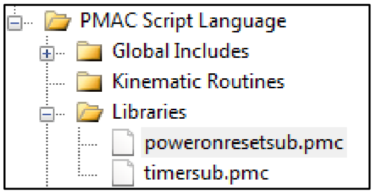
A new project menu is added as shown in the picture above. Open New Project from Wizard.
There are many types of project template available. The User can make their own project templates, and those will be shown in the New Project dialog. Exporting custom project templates is covered in a later section.



PowerPMAC	Standard basic PowerPMAC project template
PowerBrick_LV	Standard basic PowerBrick LV project
PowerPMAC with EtherCAT (Acontis)	Standard Project with EtherCAT Master Node
PowerPMAC with EtherNetIp	Standard Project with EtherNetIP node
PowerBrick_LV 4 Axis	PowerBrick LV 4 axis project
PowerBrick_LV 8 Axis	PowerBrick LV 8 axis project
PowerBrick_AC 4 Axis	PowerBrick AC 4 axis project
PowerBrick_AC 8 Axis	PowerBrick AC 8 axis project

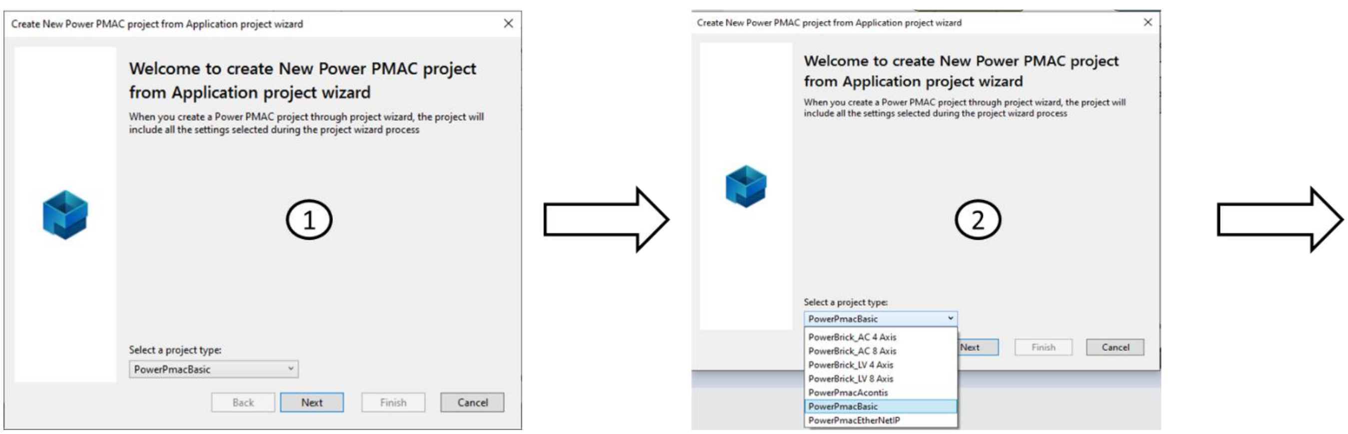
Project templates provide a quick way to start Power PMAC programming. Required programs are already included in the Power PMAC project templates. For example:

The PowerBrick_LV project template is specific to PowerBrick LV. This template provides the required subprograms for PowerBrick LV stored under the libraries folder.
PowerBrick_LV project template:



File-New-Project Wizard

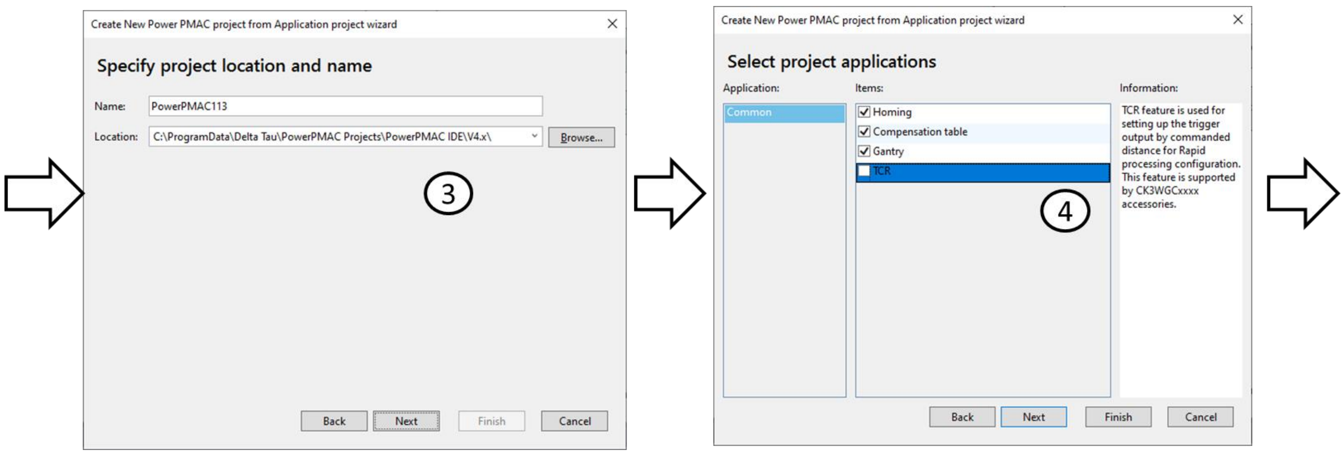
This new menu for creating project will walk you through series of questions and on Finish will create project.
Here is the workflow of creating project from wizard. This functionality will keep growing in the future releases of the IDE.
You can navigate Back and Next by clicking buttons and use Finish to end wizard and create Project.
Open a new project from wizard by selecting File →New → Project Wizard or from start page as shown above. Project wizard will open see picture mark with 1. It allows you to select necessary Project template, see picture 2. Follow the wizard Next button



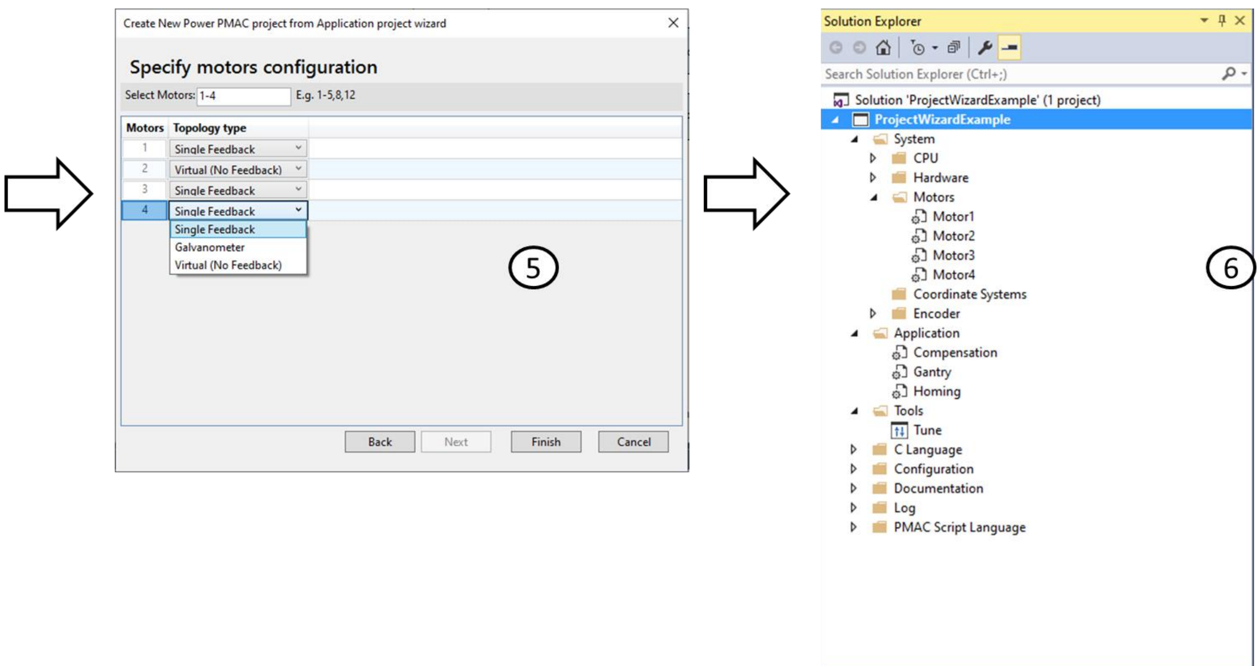
After Step 2 select Next to go to Step 3. Here you can provide Project name and location. Press next to go to Step 4. In this steps you will see Common Application Homing, Compensation Table, Gantry, TCR. Choose the application you want to add to the project and press Next to go to step 5.

Note

TCR application will require CK3WGCxxxx Hardware, part of CK3M series.



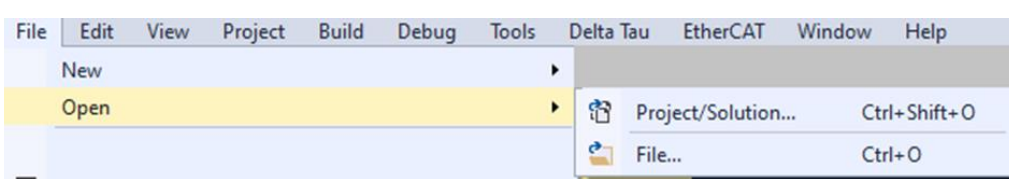
In step 5 user can add number of motors for the application. Currently only three topology types are supported from wizard. Single feedback, Virtual and Galvanometer. This is shown below.

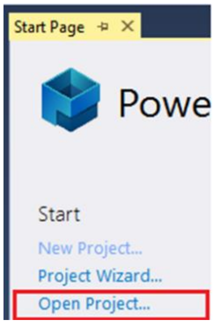


Click Finish to create Power PMAC project from wizard shown in step 6.

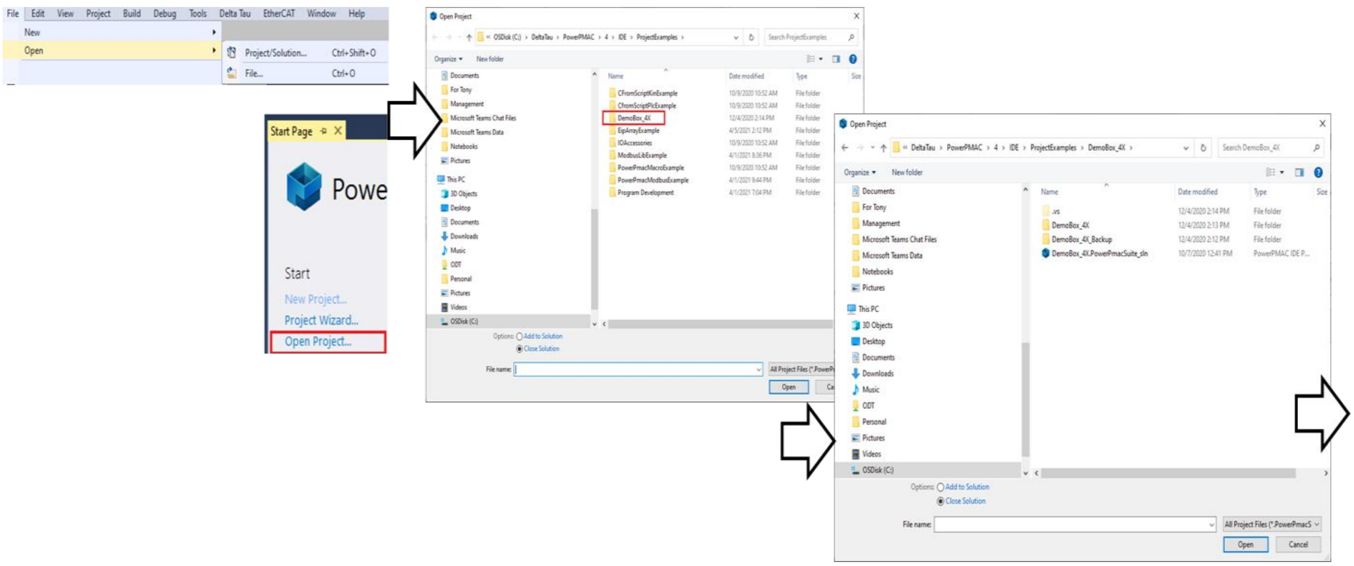
File-Open-Project

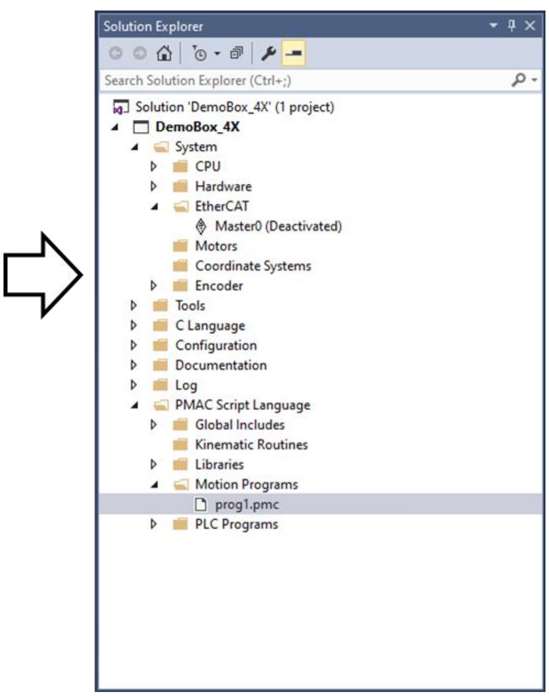
Opening an existing project by selecting File →Open menu or from start page Open project as shown in the picture.





The workflow for Opening an existing project is shown below…



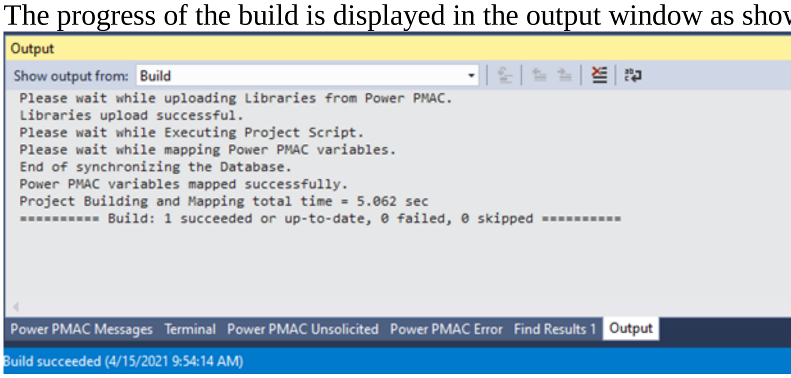


Project – Context menu

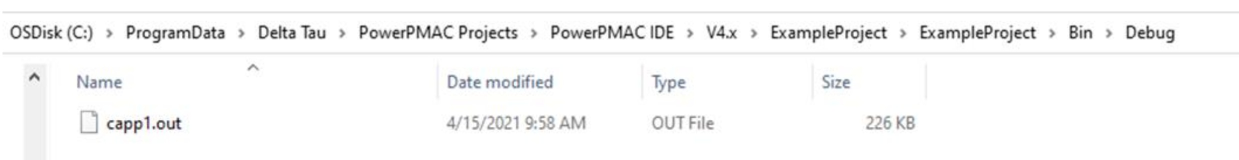
Right click on the Project to get the context menu. Here is the menu looks like…

Build

Build will build the project. This option is mainly for building C application.
The progress of the build is displayed in the output window as shown below.

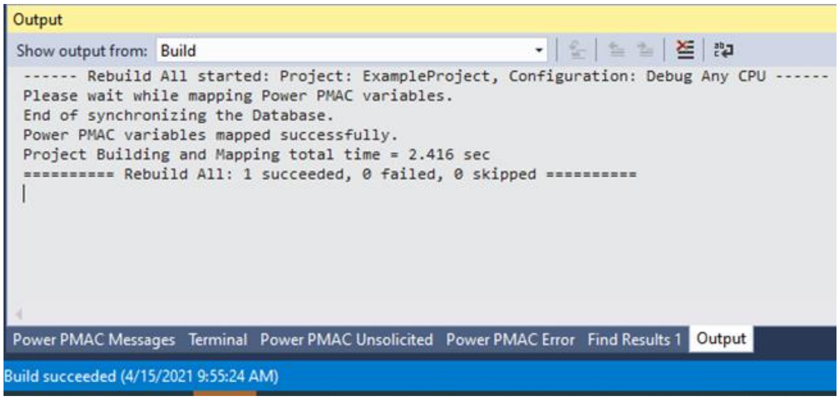


The output of the build is mainly c programs. For example after successful build operation the c output file available under Debug or Release folder depending on the mode selected.

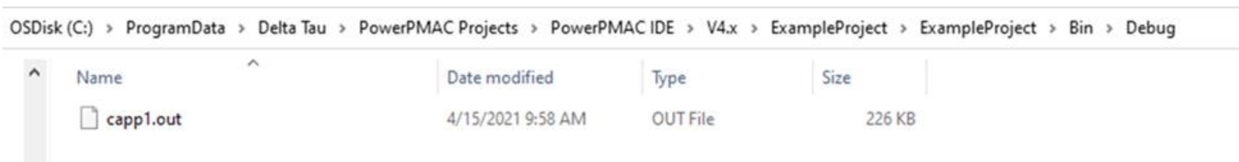


Rebuild

ReBuild will build the project. This option is mainly for building C application.
The progress of the rebuild is displayed in the output window as shown below.

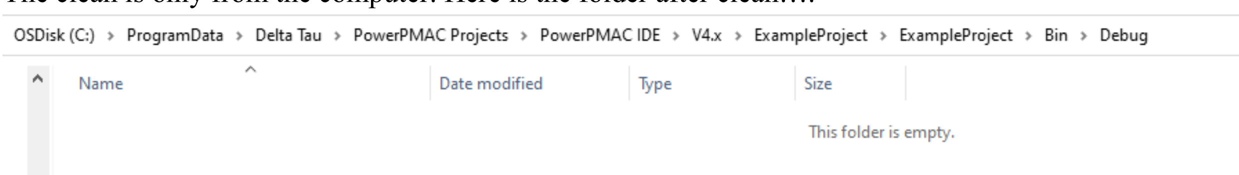


The output of the rebuild is mainly c programs. For example after successful build operation the c output file available under Debug or Release folder depending on the mode selected.



Clean

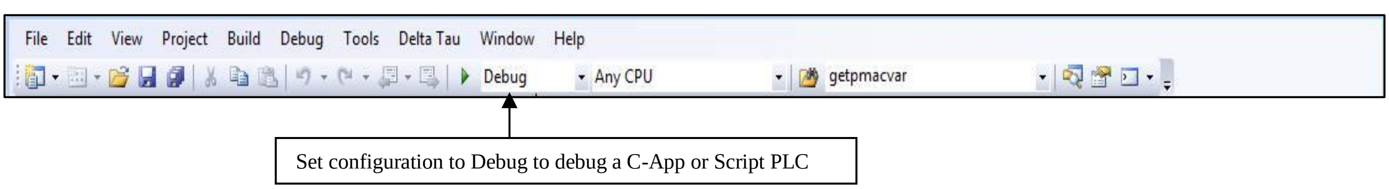
Clean will clean the build files from Debug or Release. As specified earlier build files are C files. The clean is only from the computer. Here is the folder after clean….



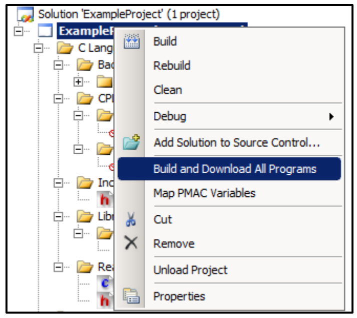
Building and Downloading the Project

This process requires two steps; the first step is to set the Solution's configuration mode.

By default, the Solution configuration is in Release Mode. If the C-App or Script PLC is required to be debugged, then set the solution configuration to Debug Mode. Debug mode generates a bigger binary file size and may fill up the Power PMAC disk. It is a good practice to compile the final version of the project in Release Mode to save space on the Power PMAC.

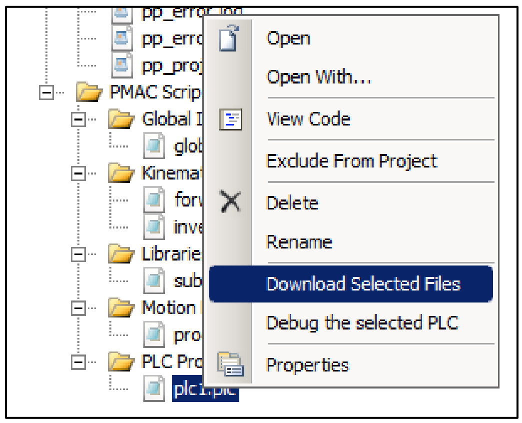


The Second step is to build and download the project to the Power PMAC. Right click the project's name and click "Build and Download All Programs" as shown below:



This will download the entire project to Power PMAC. Selected Script files can download individually or in multiples by selecting Shift+Click and selecting each file and then clicking “Download Selected Files”.

The screenshot below shows downloading just PLC 1:



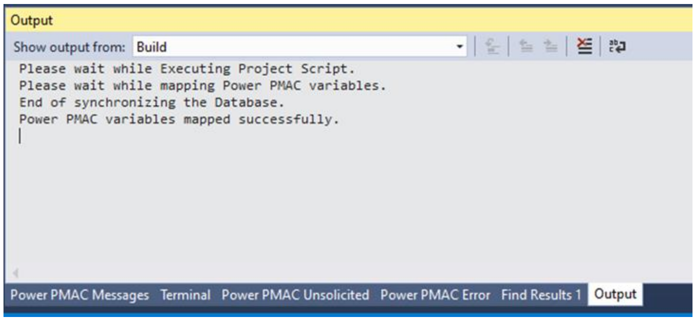
The entire project must have been built and downloaded in order to be able to download selected files. This is because the IDE must compile the C programs and map all variables as a whole; this cannot be done individually. The “Download Selected Files” feature is intended for development purposes, e.g. making several iterations of changes to just one file and then testing these changes without having to download the whole project again.

Note

Build and Download will always generate systemsetup.cfg and will download to Power PMAC. The file should not be altered as this file is maintained by Project System.
In the case of an EtherCAT® configuration downloading will also download the eni.xml and ECATConfig.cfg to Power PMAC.

Map Power PMAC Variables

Mapping, preprocesses all the script files and generates the symbol tables and pp_Proj.h file to be used by c apps. The progress of mapping is displayed in the output window.



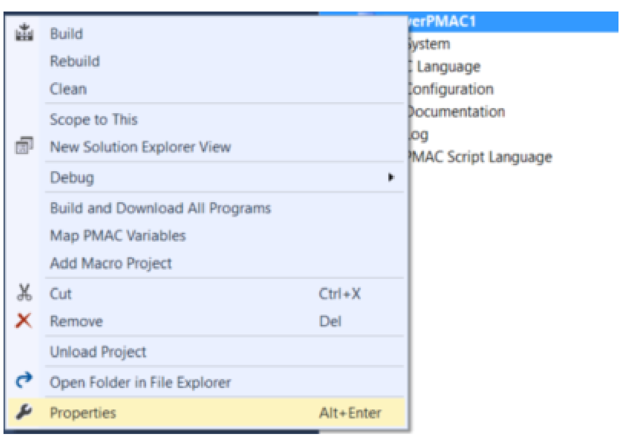
Export Project with IP Protection

IP (Intellectual property) protection allows OEM builders, independent integrators and users to protect their intellectual property by encrypting script programs. The encryption is password protected.
The current implementation of IP protection is three level.

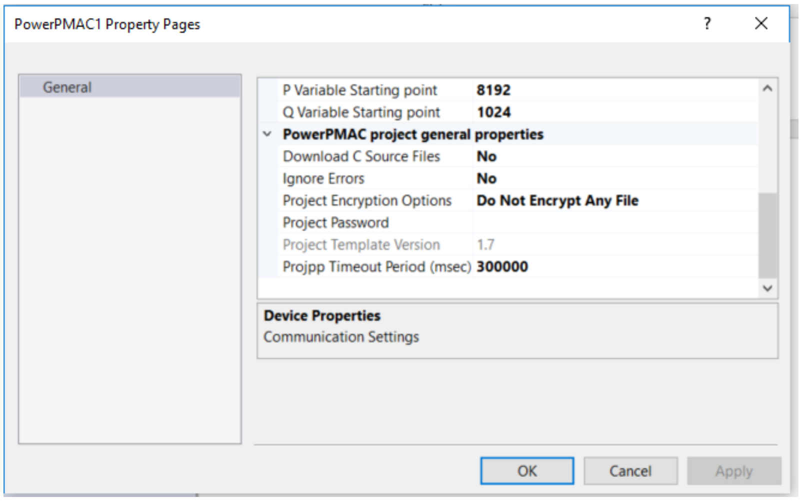
1. Customer-A can encrypt the script programs and pass the project on to Customer-B. This is level one.
2. Customer-B can take the project from Customer-A and add their own logic and protect it by encrypting and give it to Customer-C. This is level two. Customer-B cannot list or view Customer-A's code.
3. Customer-C can take the Project from Customer-B and add their own logic and protect their part by encrypting it and give to Customer-D. This is level three. Customer-C cannot list or view Customer-A's or Customer-B code.
4. Customer-D cannot list or view Customer-A's or Customer-B code.

Steps

1. Open project (New or previously created)
2. Open project properties to choose how to encrypt the project.



In the properties windows go to the project encryption options



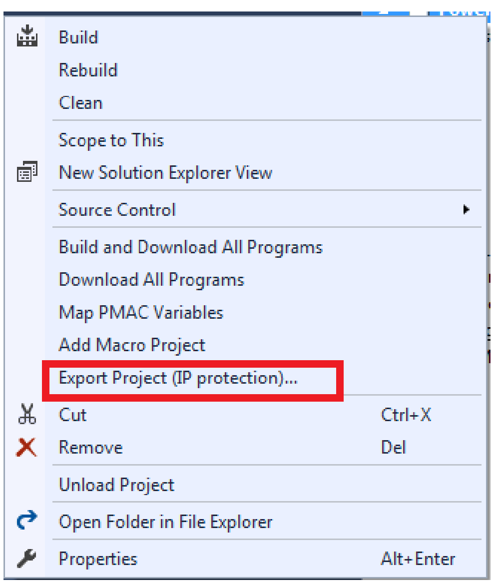
Select encrypt all project files or some project files and set a password for the project.
If some items are selected to be encrypted click on a project item and choose yes to Enable Encryption property, build and download to verify the project is building and downloading.
Right click on the project and select Export menu option.

3. Build and Download the project.

Note

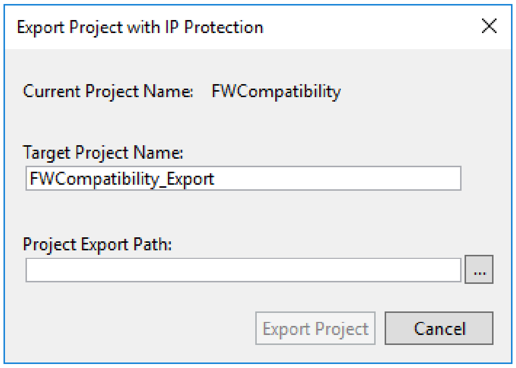
Build and Download is a necessary step for the Export Project (IP Protection) option to become enabled.

Right click on the project and select Export Project (IP Protection)… menu option.



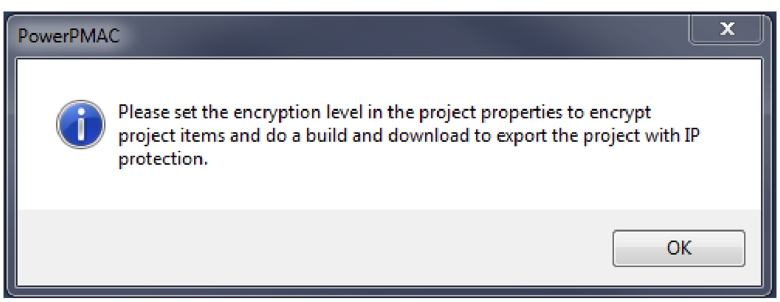
The opened dialog will ask for an exported project name and the path to export the project to. Click export once the project name is entered.

4. Click Export to export the project and follow the instruction to name the project and etc.

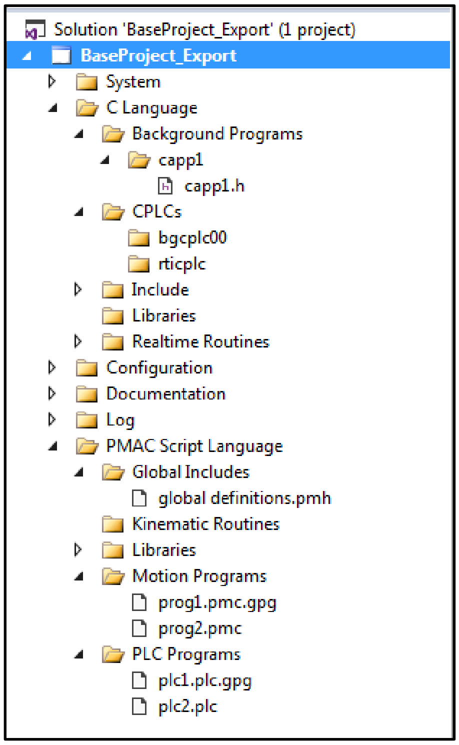


On opening the exported project, the PowerPMAC script items chosen to be encrypted will have been replaced by the encrypted versions of the files. The global include *.pmh files will not be exported as an encrypted item even if they were selected to be encrypted. The password field must be empty so that a new password can be entered for the exported project. IP protection supports two level password and three level of IP protection. IP protection will support multilevel c apps as well.

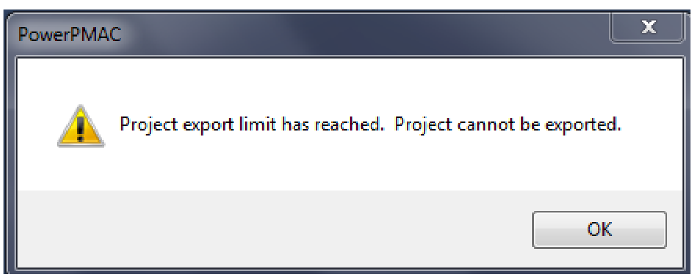
If the Export Project (IP Protection)… is clicked and the build and download of the project have not been performed the following message will be shown:



5. Opening the exported project will look like the following:

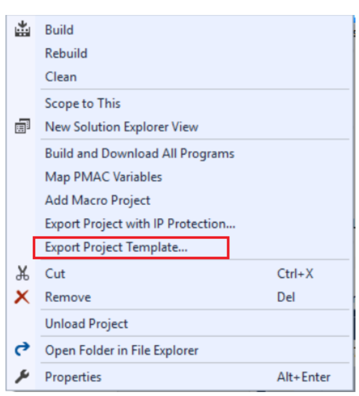


6. As stated earlier the IDE supports three level IP Protection meaning the project can be exported twice. If an attempt is made to export for a third time the following warning will be shown:



Export Project Template

This option can be access from Project level context menu. Right Click on the project and select Export Project Template as shown below.



Note

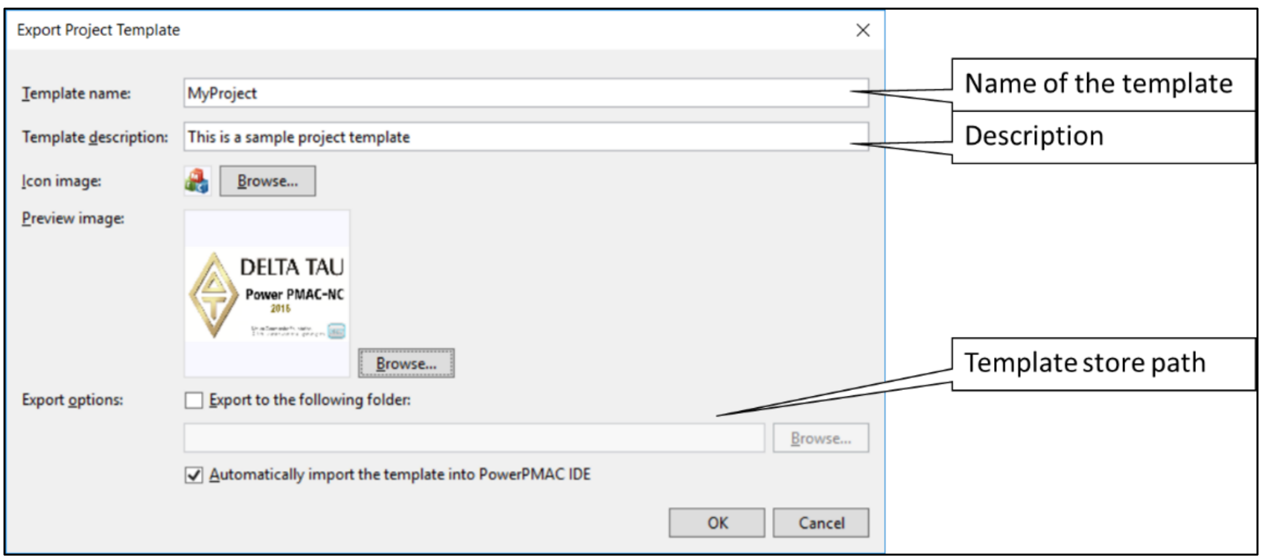
IDE V 4.3.0.x and below support accessing Export/Import Project Template from the File-Export-Project Template menu. This option is replaced with Template Manager in IDE V4.3.2.x and above.

This option allows the user to:

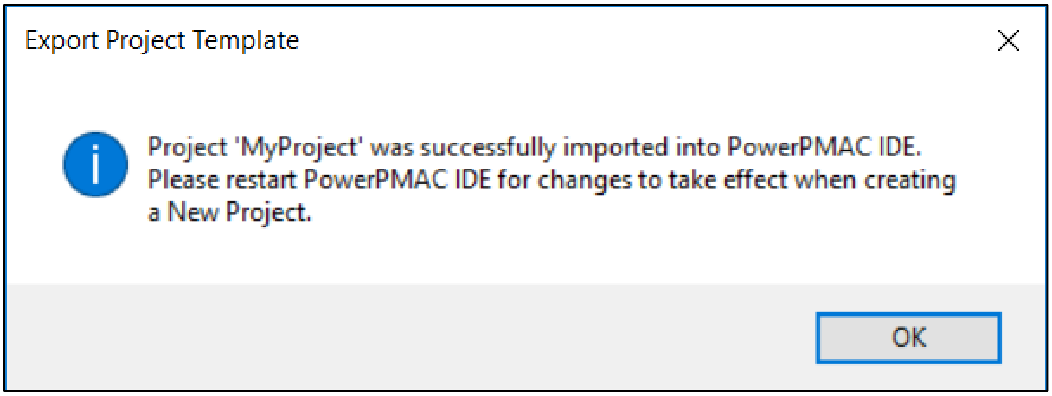
- Export a project so that other users can use it as a base for their projects.
- To add an icon for the custom template.
- Export a project and automatically add it to their New Project dialog.
- Preview the information about the project that is being imported.
- Delete custom project templates.
- Import a project template in order to use a pre-configured base project.

Note: - The user is prevented from importing a template that is not supported by the current IDE version.

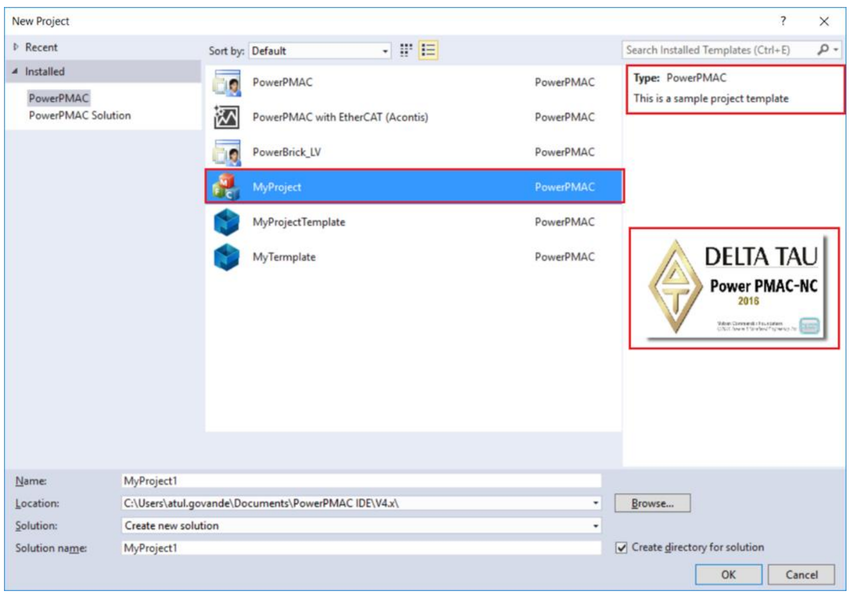
On clicking the option to Export the following dialog will open:



In this dialog the default is set to “Automatically import the template into Power PMAC IDE”. The User can uncheck this option. On selecting Ok, the template will be created, and a success message will be displayed.

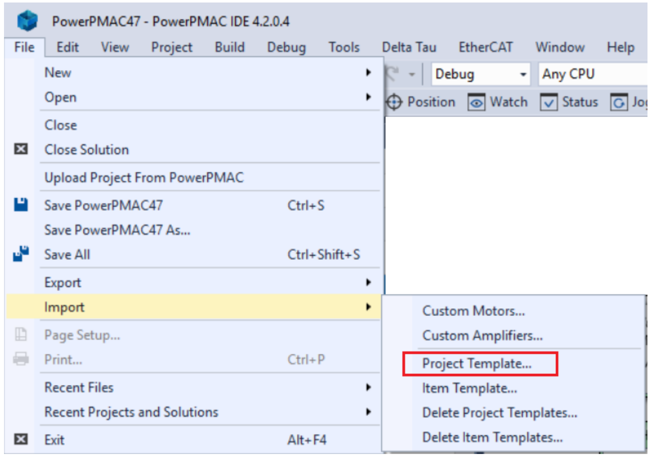


The exported template is available to load from File-New-Project, as shown below.

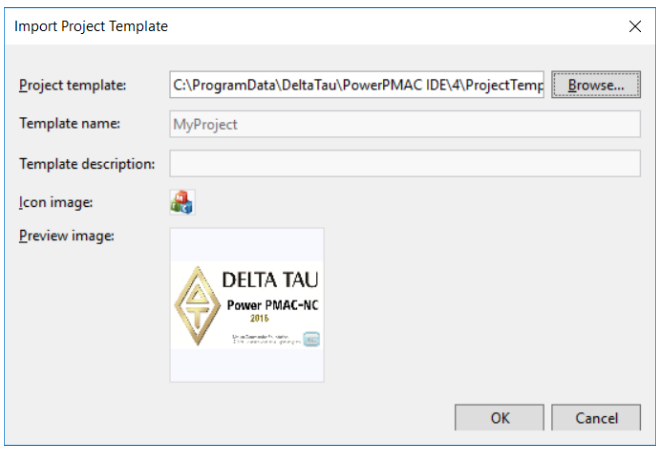


The red square shows the options selected when the template was created.

To share a exported template use the option File-Import-Project Template option as shown below.



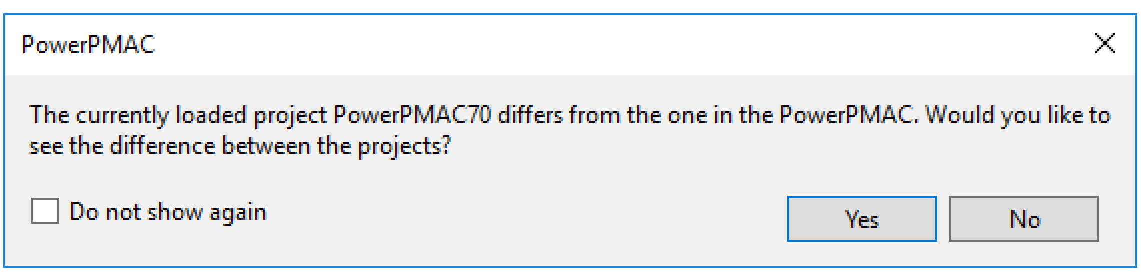
This option allows user to create the base project and export as a template and share.
On selecting, this will open the Import Project Template dialog as shown below:



On clicking OK, the project template will be imported and will be available to use from File-New-Project dialog.

Comparing a Project

IDE version 4.3 and above allows the User to compare the active project on Power PMAC with the local one on the PC. On opening the project, the User will see this message…

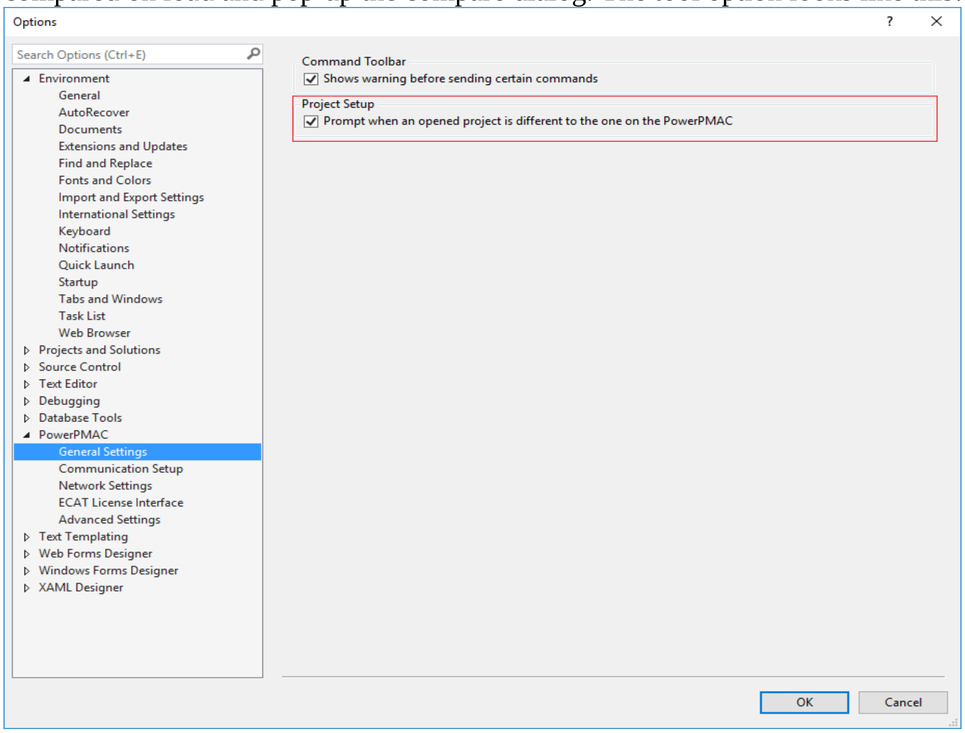


The User will be presented with three choices.

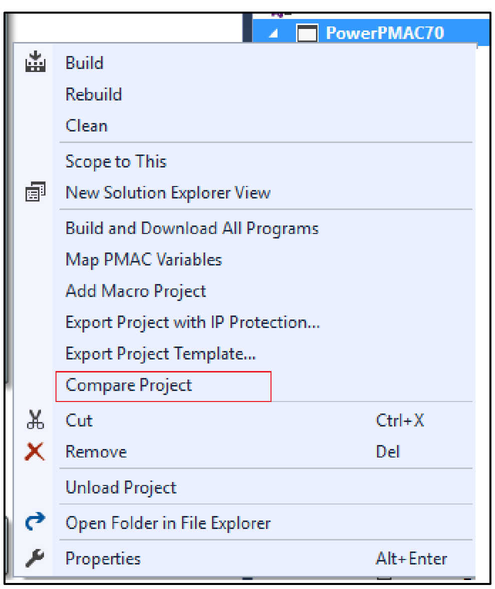
1. If the User would not like to see the differences, then they will click 'No'.
2. If the User would like to see the differences between the two projects, then they will click 'Yes'.

3. If the User does not want to compare project every time they open their IDE they can select 'Do Not Show again' check box which will stop this dialog from being shown.

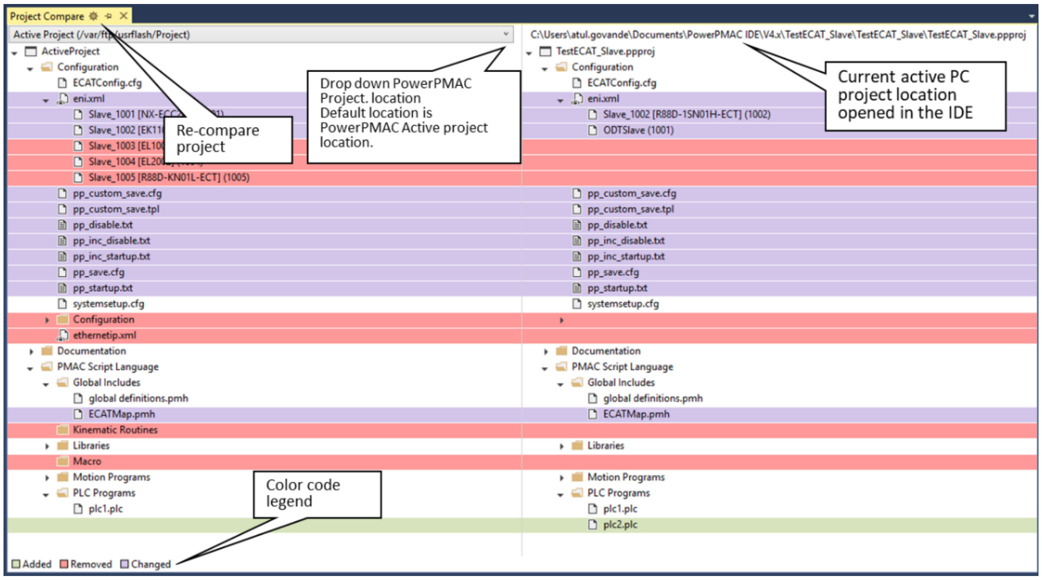
The User can enable the compare dialog again by going to Tools-Options-Power PMAC-General Settings. The option is marked in Red square. Check the box so next time when the IDE opens the project it compared on load and pop-up the compare dialog. The tool option looks like this…



If the User clicks 'Yes' it will open Project compare dialog. The same Project compare dialog can be opened by right clicking on the solution file and then selecting Compare project context menu. The context menu looks like this…



The Project Compare view looks like this…

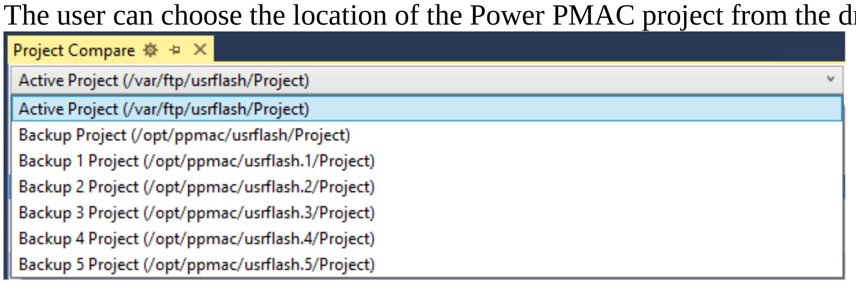


Indicates file(s)/folder(s) are different

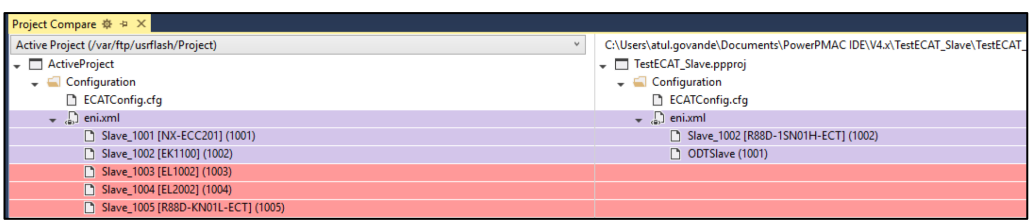
Indicates file(s)/folder(s) are only on PowerPMAC

Indicates file(s)/folder(s) are only on PC

The user can choose the location of the Power PMAC project from the drop-down list, like this…



If the EtherCAT .eni files are different then the .eni file will be expanded, and the user will see the comparison of slaves that are part of the eni file on Power PMAC vs PC. It is displayed like this…

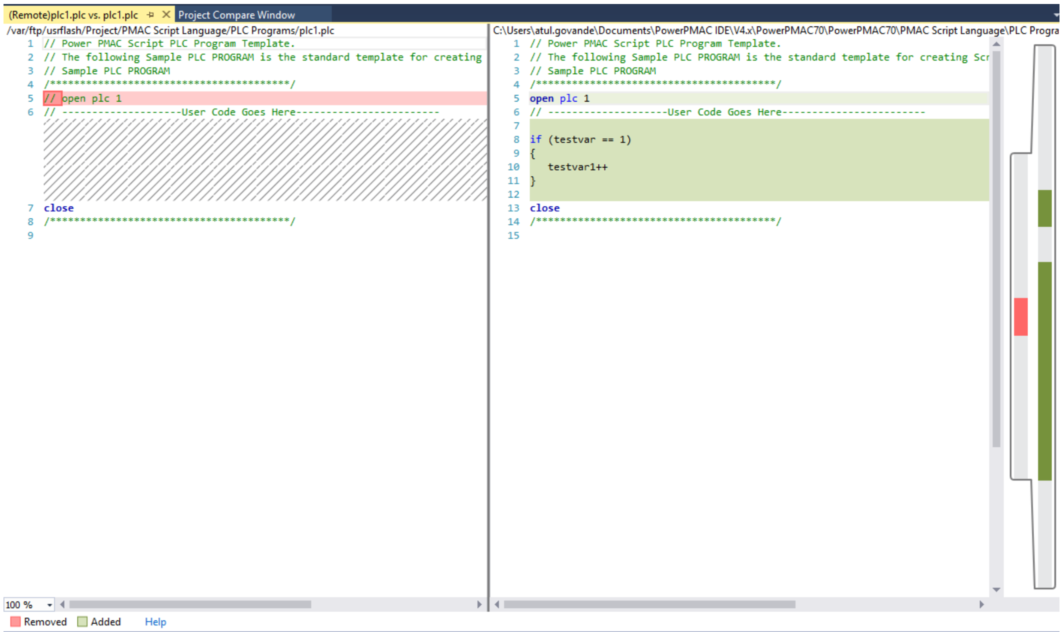


Comparing a File

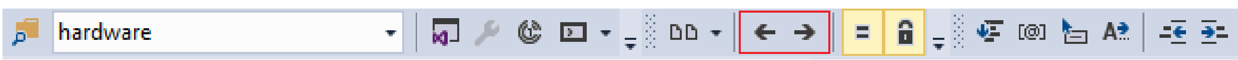
IDE V4.3 onwards will support file comparison directly from the Compare Project dialog. The user can compare .eni files, as well as navigate files and folders. The user can compare the project opened in the IDE with a Power PMAC backup project (userflash, userflash.1, userflash.2 etc.).

As described in the compare project, this color indicates the files are different.

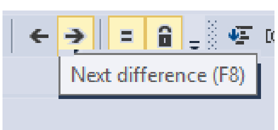
The user can double click on the file to view the difference or right click on the file to open the compare context menu. The file compare view looks like this…



The user can scroll up/down the file to see the differences. The user can also use the arrow keys from the IDE menu to jump to the next/previous difference.



Hover the mouse over the arrow keys to see the tooltip.

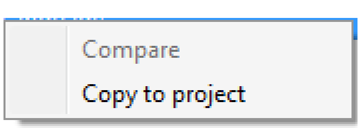


Limitation of file compare:

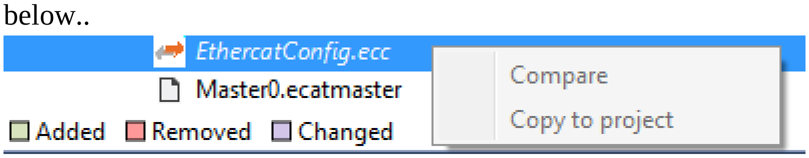
1. The user cannot compare bin files. For example capp1.out
2. The user cannot see the differences between encrypted files (.gpg).

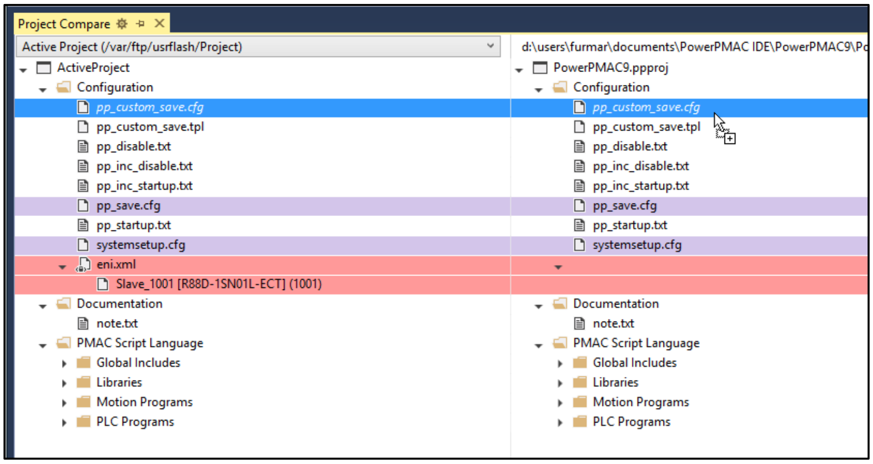
Copying files/folder

It is possible to copy files or copy folder from Active project on PMAC to PC.
On right click to File or folder the context menu will change depending on the permission level.



For example .ecc file cannot be compare (Binary file) and copied so menu will be disabled as shown below..

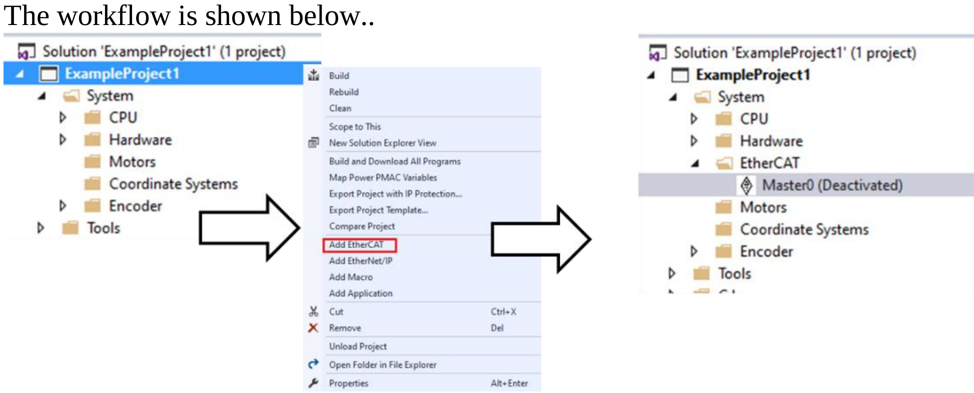




Note	1. The copy is only one direction, **Power PMAC to PC (Power PMAC IDE).** 2. Slaves under EtherCAT cannot be copied from Power PMAC to PC project.

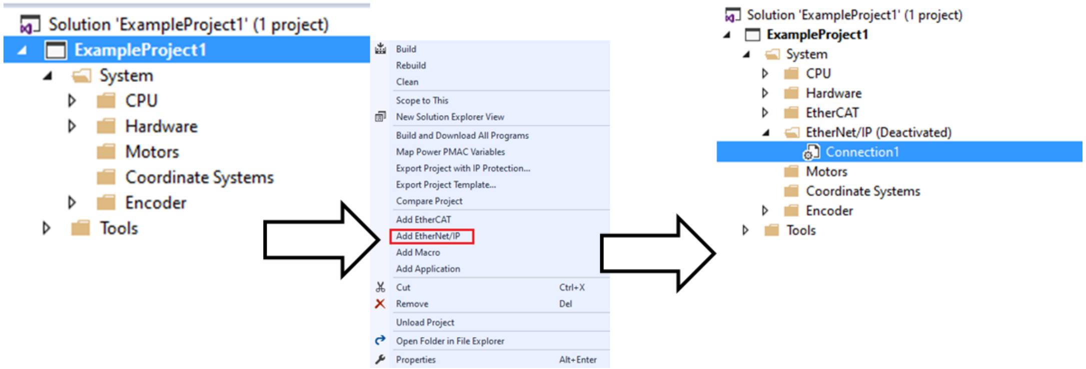
Add EtherCAT

Add EtherCAT context menu adds EtherCAT Master so user can add EtherCAT devices. This option is dynamic option. If user opens the Basic project where EtherCAT node is not present in the project tree. Under this case if user required to add EtherCAT network to existing project this option is used. The workflow is shown below..



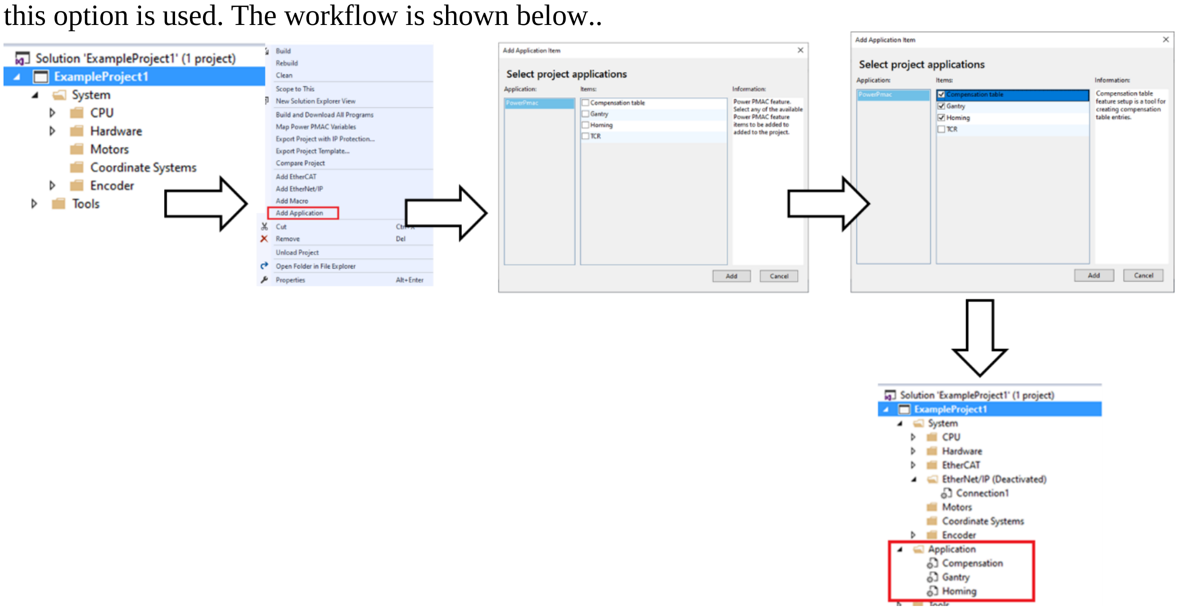
Add EtherNet/IP

Add EtherNet/IP context menu adds EtherNet/IP Node so user can configure EtherNet/IP connections. This option is dynamic option. If user opens the Basic project where EtherNet/IP node is not present in the project tree. Under this case if user required to add EtherNet/IP network to existing project this option is used. The workflow is shown below..



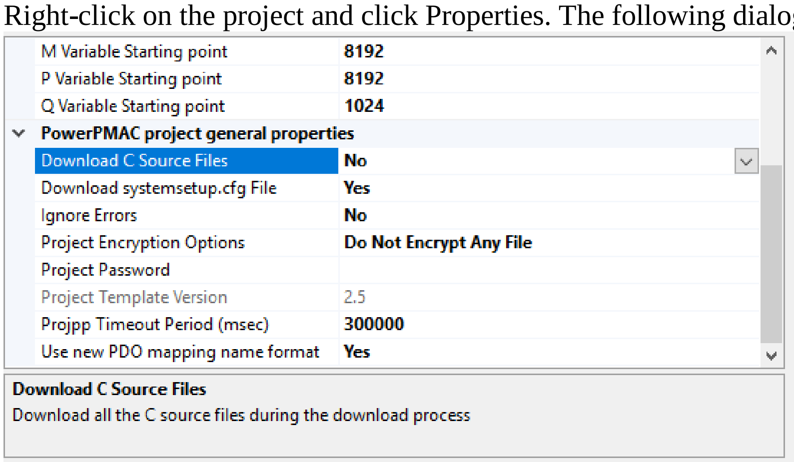
Add Application

Add Application context menu adds Application Node with selected application so user can setup and configure. This option is dynamic option. If user opens the Basic project where Application node is not present in the project tree. Under this case if user required to add Application support to existing project this option is used. The workflow is shown below..



Properties

Right-click on the project and click Properties. The following dialog shows.



The properties are self-explanatory.

Note

For IDE V4.1.x and above the new property 'Download systemsetup.cfg file' is set to No when the project is upgraded from V3.x project. For any New Project this property is set to Yes as we recommend user to use the cfg file. The IDE automatically maintains the file and saves any changes from IDE domain to this file.
This property is not available for IDE V4.0.x.

The new property "Use new PDO mappingname format" is added to support new naming method for EtherCAT PDO's. There is no longer a limitation for EtherCAT PDO names. This property is from V4.3 so if the project is upgraded from previous version this property is set to No and any new Load PDO mapping command will use the old mechanism of PDO naming.

Project – Common operation

Adding and Removing Files

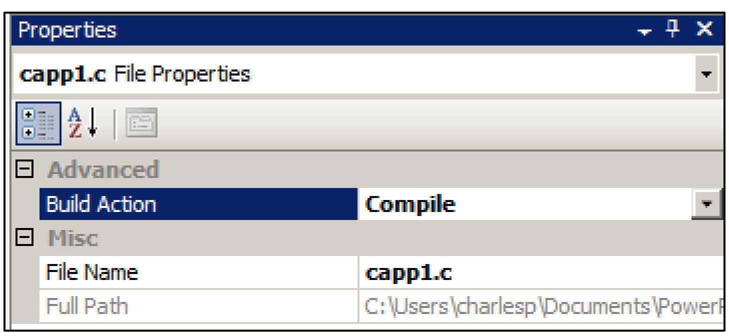
Add new or existing files to any subfolder by right-clicking it and selecting Add and then either New Item or Existing Item:

Then browse to the item to be added or included. From IDE Ver. 4.2 Script Language files added to the project will be displayed in a natural order. The script files can be moved up or down by right click and opening context menu.

In the previous version of the IDE these files were displayed in alphabetical order.
To remove file simply select file and Delete file.

File Properties

Right-click any file and click Properties. The following dialog is shown:

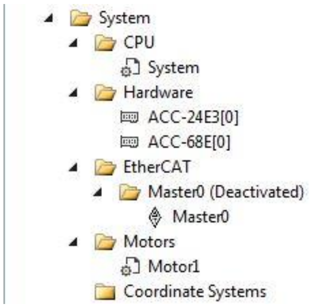


The Build Action box can be set to either Compile, Content, Embedded Resource or None. For Script Files none of these options have any effect. For C program files (*.c file extension) setting this to Compile will cause the file to be compiled. Any other setting will cause the file not to be compiled. All C header files (*.h file extension) should be set to Content to be linked with the *.c files but not themselves compiled.

System

Layout

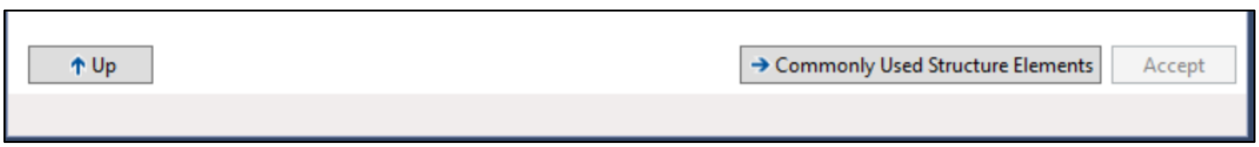
The system folders store the CPU, Motor, Coordinate system and Encoder settings.



Common for all the views from system folder items

The following button strip is common to all the system folder view. When user clicks the Global Clock block under System-CPU-System folder node.

Type 1

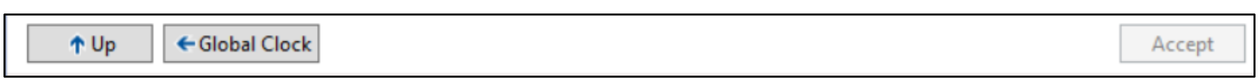


Up button: Returns back to the originator of the view.

Commonly Used Structure Elements button (Next): This is dynamic Next button and text that appears is the next logical choice for parameter setup. This button appears if the Next option is available.

Accept: Clicking this accepts the parameter settings once they have been selected. Not clicking on the Accept means that the settings will not be downloaded to Power PMAC.

Type 2



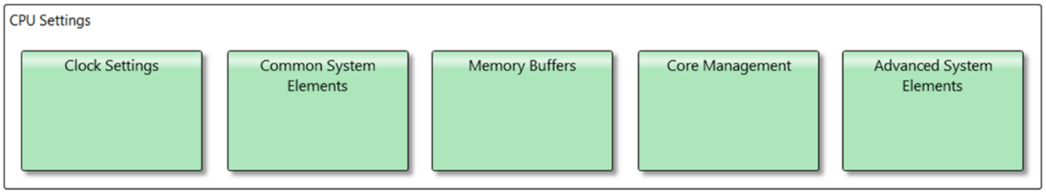
Up button: Returns back to the originator of the view.

Global Clock (Prev): This is dynamic Prev button and text that appears is the Previous logical choice for parameter setup. This button appears if the Prev option is available.

Accept: Clicking this accepts the parameter settings once they have been selected. Not clicking on the Accept means that the settings will not be downloaded to Power PMAC.

CPU

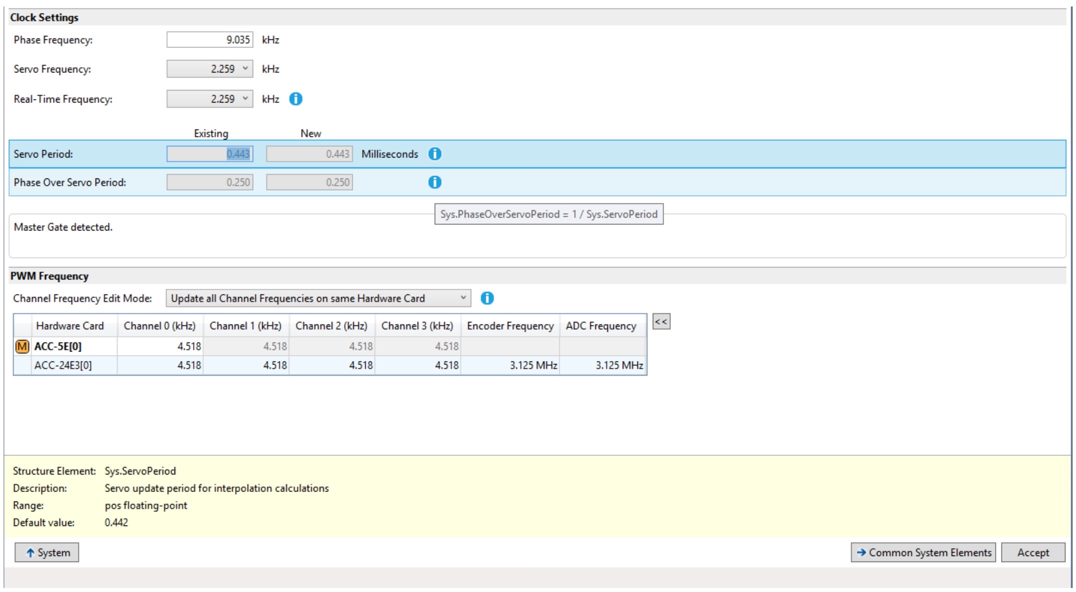
This contains the Power PMAC system setting such as global clock and commonly used Power PMAC system settings. This is a new view available after V4.2. The commonly used system block is broken down into separate blocks based on function, for improved usability.



Clock Settings

The Clock Settings window is used to configure the Global Clock

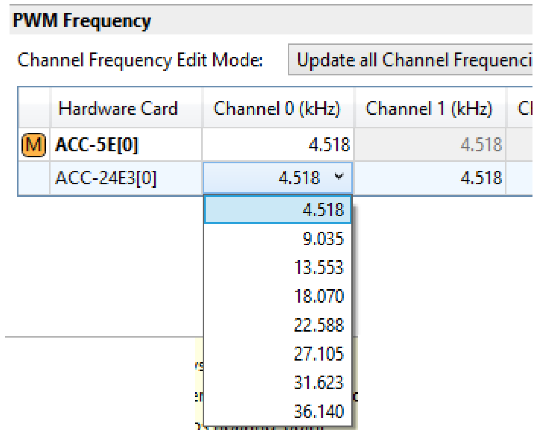
The first screen is used to set up the global clock frequencies for the system. Type in the frequency required, in kHz and click “Accept”. The Up arrow navigates back to System block or Left arrow can take to the next block that is common System elements. The Symbol M indicates the master clock source. Hovering the mouse over i symbol will provide more information about settings.



Note

If the software detects the EtherCAT® option but does not detect a Master Gate it will automatically force Power PMAC to use its internal clock by setting **Sys.CPUTimerIntr = 1**. For EtherCAT® the servo period must be multiples of 62.5 µsec. Upon accepting the clock settings issue, a **save** and **$$$** in the Terminal Window for changes to take effect and then check the value of **Sys.ServoTime** in the Watch Window to ensure it is counting continuously before proceeding.

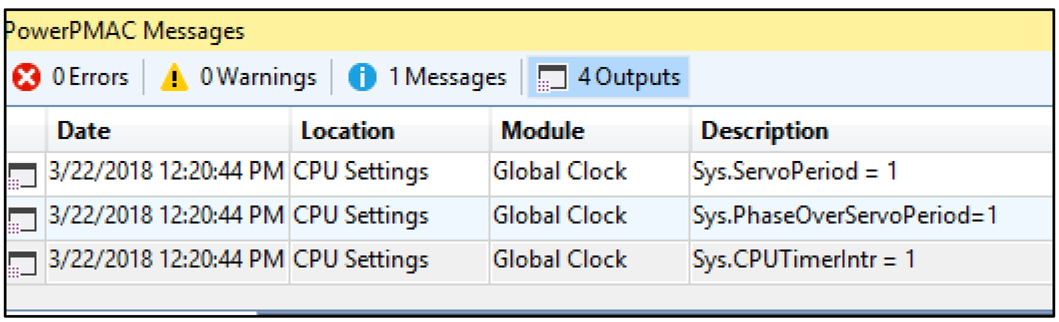
The PWM frequency for each channel on the axis interface cards can also be set if there are any. To change this right-click on the channel of the PWM frequency to change and then select one of the possible options displayed as shown below:



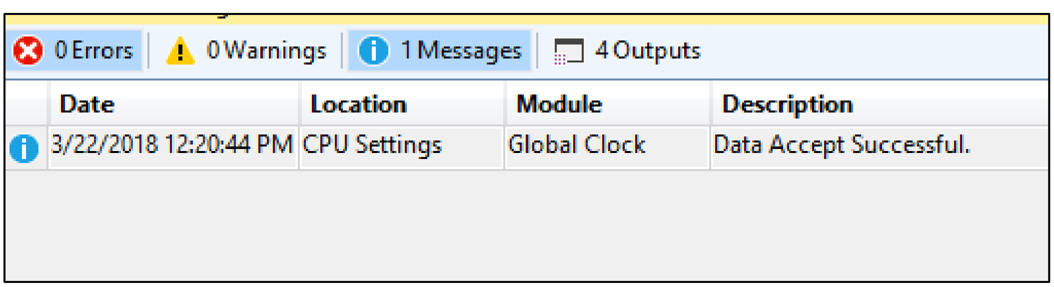
The Window containing four tabs at the bottom of the screen display's useful information as the system is configured.

The four columns give further detail as to the origination of the information i.e. the Location and Module.

The Output tab shows every command that is being sent to Power PMAC.

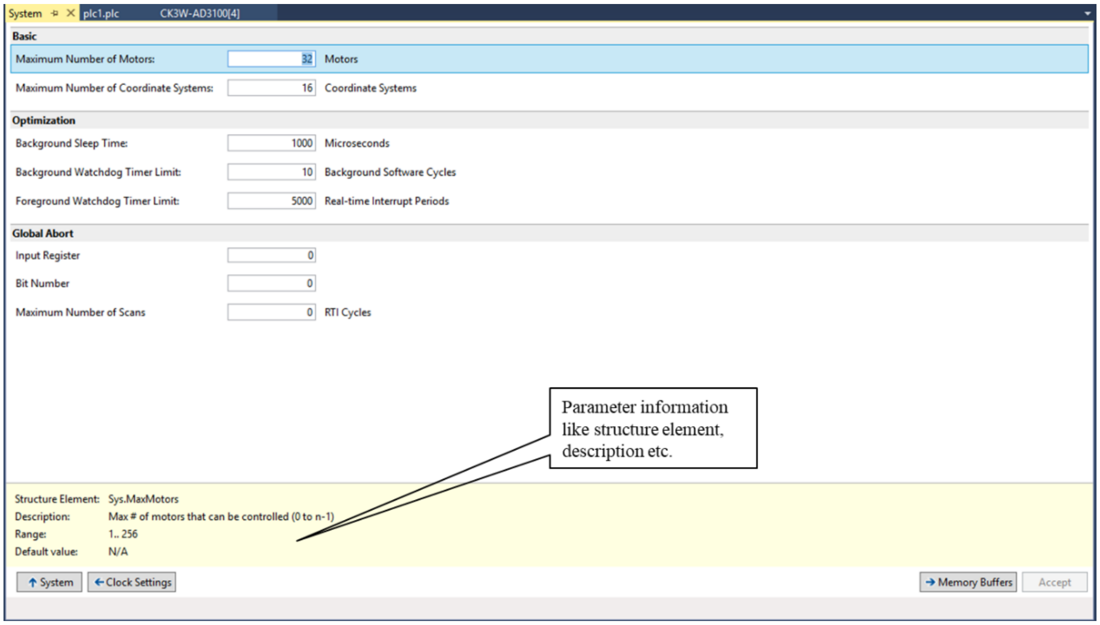


The Messages Tab displays setup-related parameters that have been changed, in this case in the Global Clock.



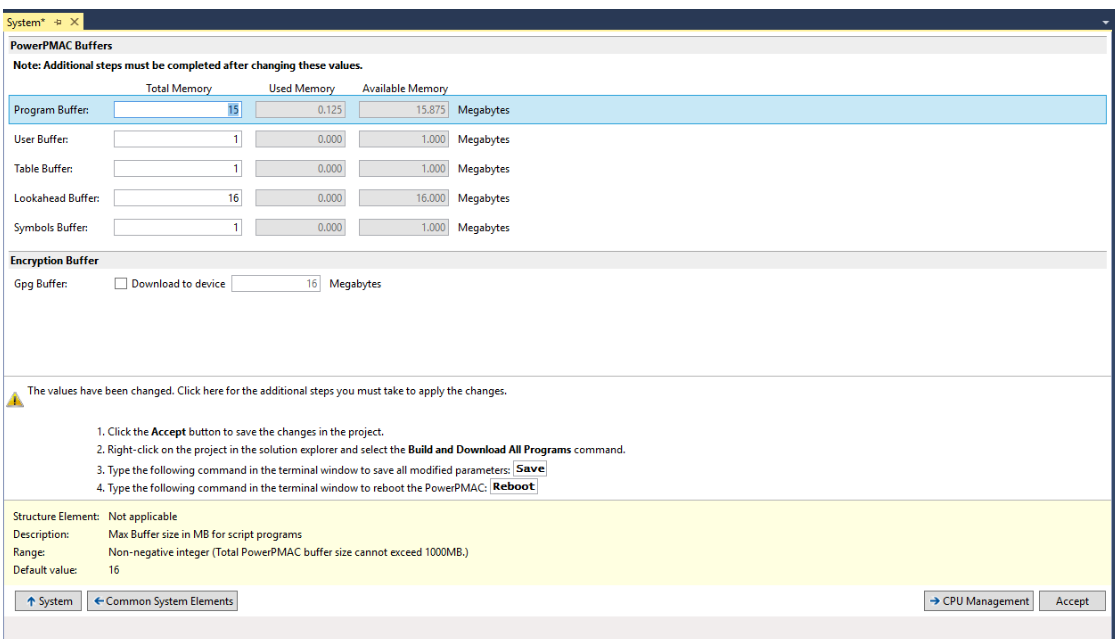
Commonly System Elements

This page shows the typical system parameter. Most of the times user will change settings in this page. Global Abort section will be prefilled for PowerBrick. For regular Gate 3 (Acc24E3 or CK3WAX) user will need to select.



When the parameter is selected the details about that parameter will be displayed in the parameter information panel at the bottom of the page. If any parameter needs additional steps these will also displayed at the bottom of the page, as shown below.

Memory Buffers



Note

Saved System data structure element values set from Global clock and Commonly Used Structure Elements are automatically stored and maintained in the file under the CPU node. Similarly, all the gate values are stored and maintained automatically under the Hardware node. These values are used when the build is performed to generate the systemsetup.cfg file.

Core Management

Here user can set the relation between task and CPU core. This functionality is not supported on the FW version lower than 2.6.x.x. here is the core management function support table:

FW Version	ARM Dual Core CPU (CK3E,UMAC)	CK3M (ARM Dual Core)	ARM Quad Core CPU(UMAC)	Other CPU
Less than 2.6.x.x	Not supported and option grayed out	Not supported and option grayed out	Not supported and option grayed out	Not supported and option grayed out
2.6.x.x. or above	Supported	Supported	Supported	Not supported and option grayed out

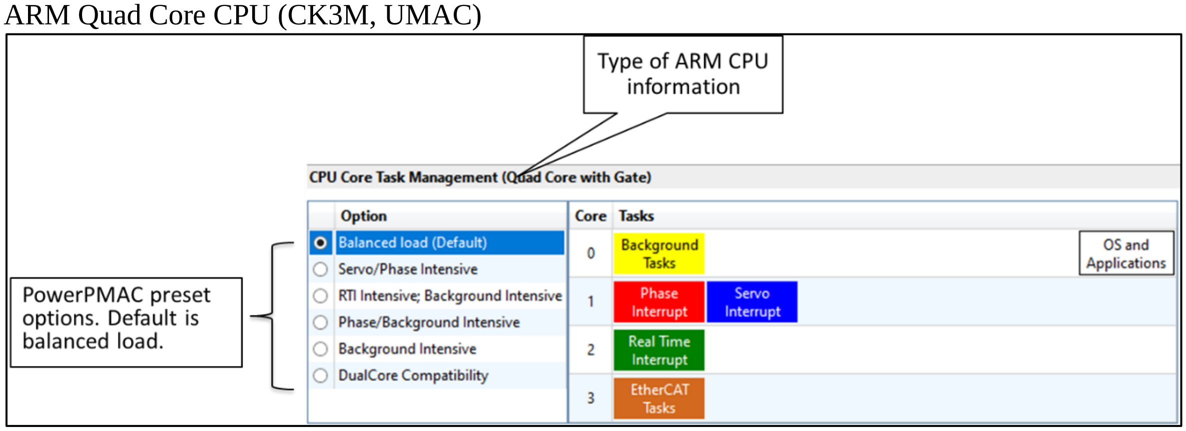
Note

Most of the application will never need to change the default balanced core management settings. Core management settings are not general settings.

On clicking the core management, a new dialog will open showing the current settings from the project. The dialog looks like this..

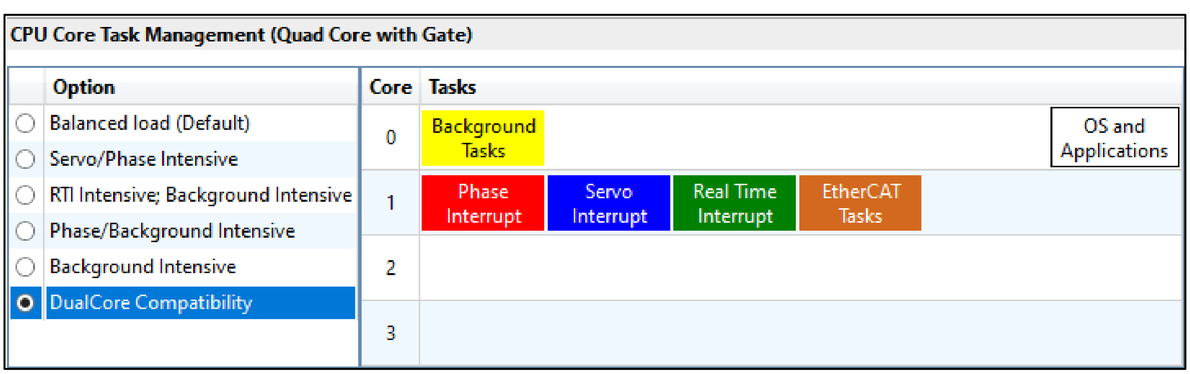
This dialog is for Quad core. The preset options varies depending on type of the CPU and gate availability.

ARM Quad Core CPU (CK3M, UMAC)



Dual core compatibility option selection under QUAD core management is designed for users who are porting from Dual core to Quad core but do not want to change core management.

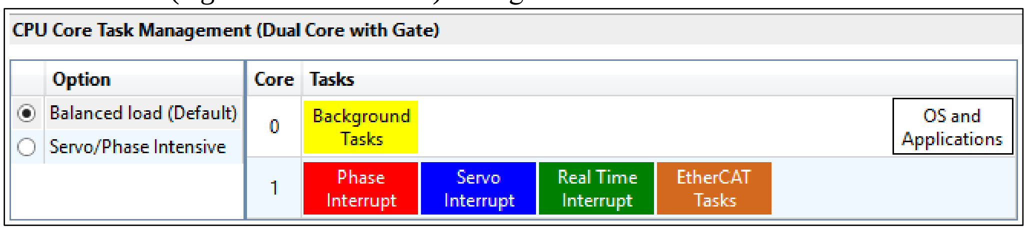
Dual core compatibility sets the cores similar to Dual core Balanced load configuration as shown below.



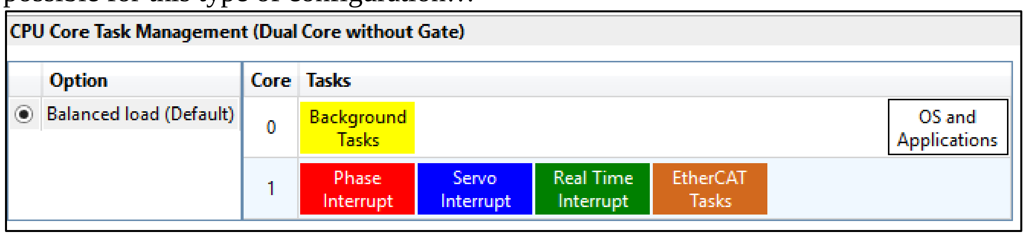
A project must be open to manage core assignment. The new settings are stored in the project.

Note

The dual core (e.g. CK3M with Gate) dialog looks like this…



The dual core (e.g. CK3E or CK3M with No Gate) dialog looks like this. There is only one setting possible for this type of configuration…



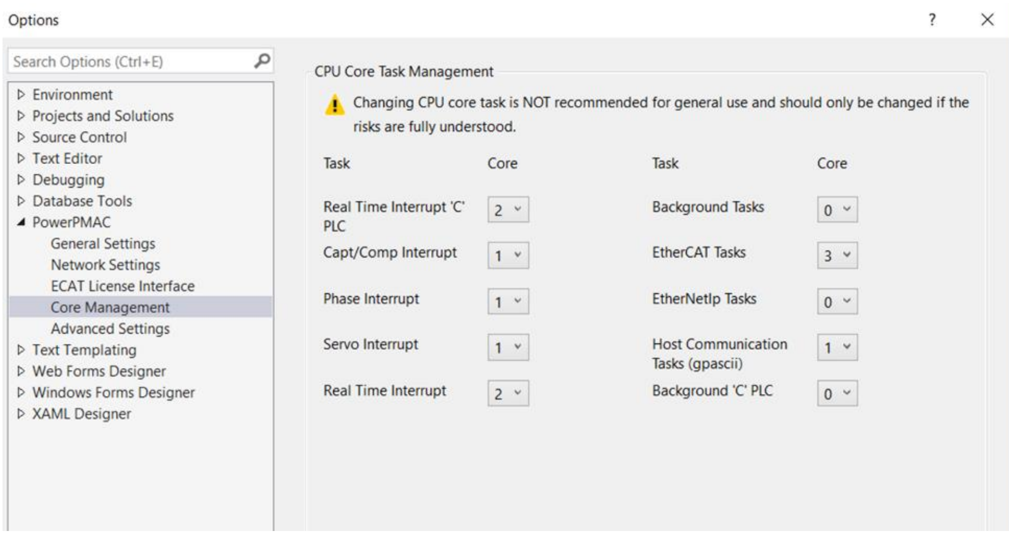
Balanced load (Default) option is selected when a new project is open. All available preset options are factory recommended. The settings will reflects how you will see the tasks in the task manager.

Any change to CPU core task management option from current option requires following steps for successfully applying the change…

1. Build and Download the project
2. Save the project
3. Reboot or power cycle Power PMAC.

Note

There is a possibility that the settings read from the project may not match the preset settings; in this case an additional Custom entry will be appeended to the table. If the User wants to set different core assignments than preset settings, then select Tools-Option-Power PMAC-Core Management.
The advanced settings are available for QUAD core CPU only.
The factory recommendation is not to change core assignment settings from this screen unless all the associated risks are understood. It is recommended to use the selection from preset settings on the core management screen. The Advanced options dialog looks like this…



Any change to core management settings are stored in the project, so next time the project is opened these settings are available.

Note

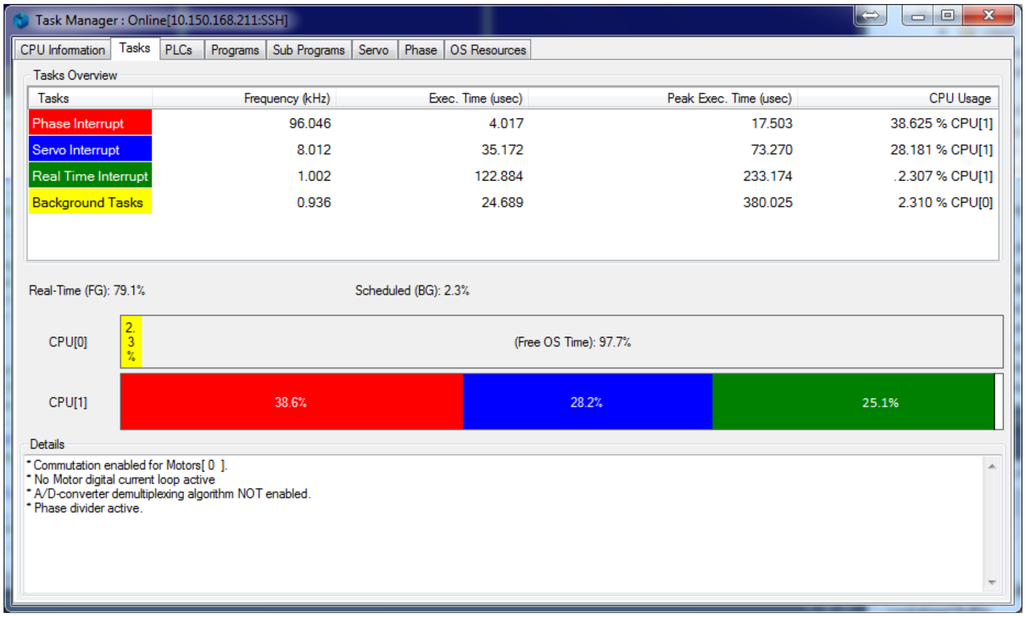
System difference will display the core management differences between Power PMAC system (Device) and project, as indicated by a flashing warning sign at the right bottom corner.

The Tool software makes sure that the Sys.CoreBgcplc = 1, always matches as Sys.CoreBackground and Sys.corertiplc always matches with sys.corerti. The mismatch is only possible in QUAD core only if user uses Tools/Option/Core management as a advanced user. Dual core mismatch is not possible from Tool software.

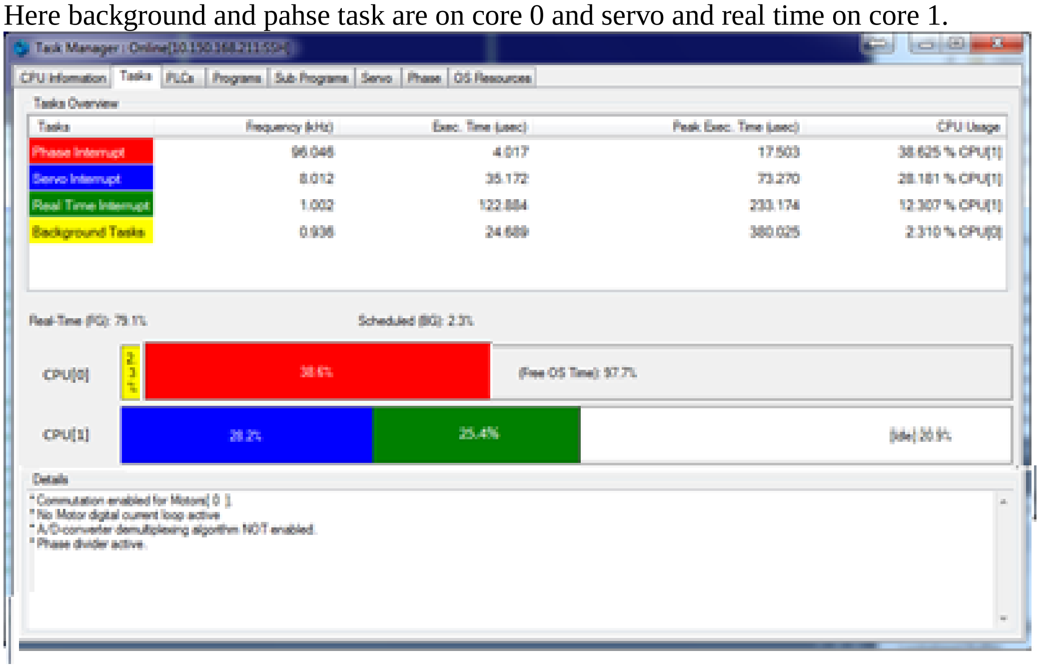
Example of using Core management

Case 1: UMAC/CK3M ARM Dual core CPU with Gate Hardware Phase/Servo Mode and many commutated axis

In the situation where we are using many commutated axis with no core management CPU0 is overloaded. (See below picture) The result could be Phase Errors, Servo Errors, Runtime Errors occur in customer applications

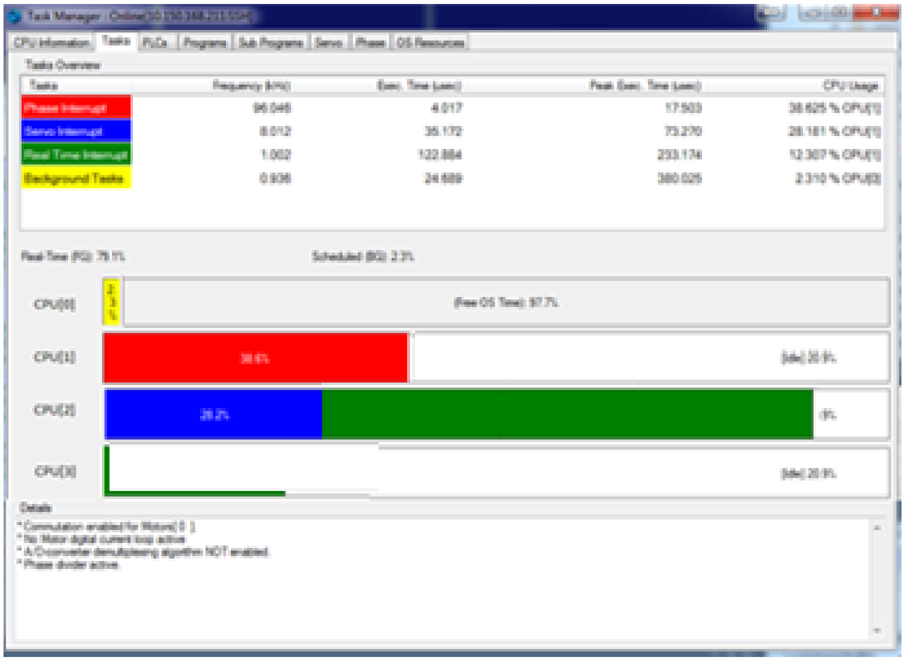


Now use core managemt and you will see Phase Errors, Servo Errors and Runtime Errors no longer occur Here background and pahse task are on core 0 and servo and real time on core 1.

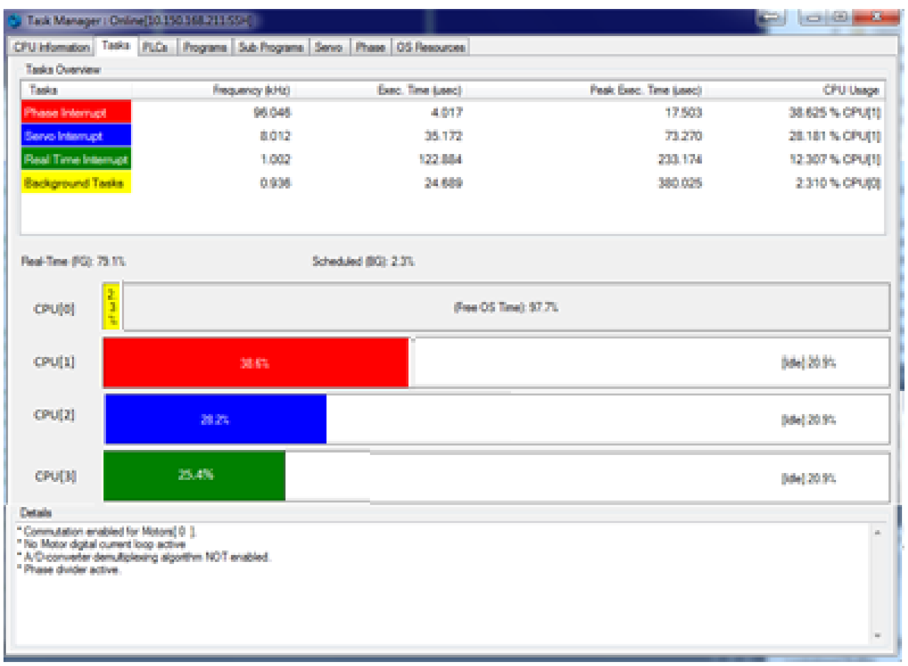


Case 2: UMAC ARM QUAD core CPU with Gate Hardware Phase/Servo Mode with sophisticated kinematics

In this case real-time interrupt handles motion planning. Application demands large robotic kinematic application with large computation in real-time. If core management is not used then there is a possibility of run time error.

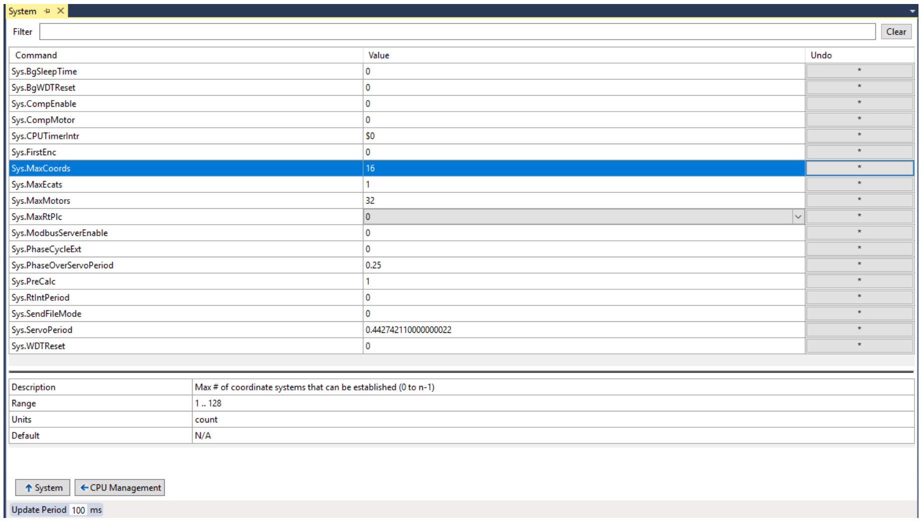


Now use core managemt and you will no longer see run time error or any other calculation error because of no time.



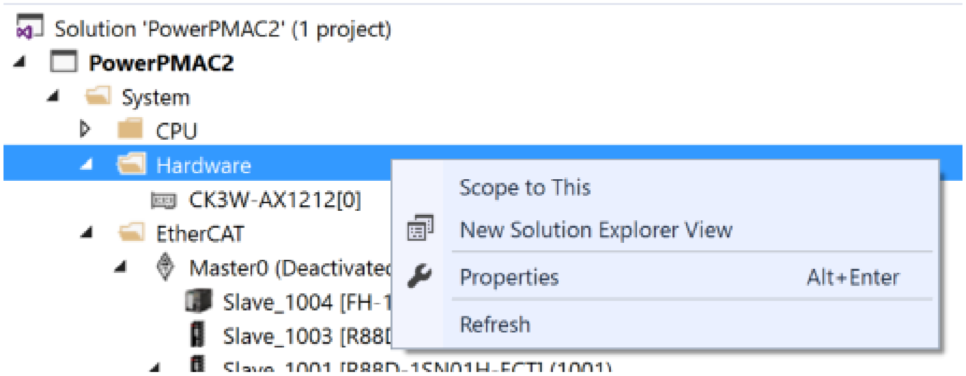
Advanced System Elements

This dialog allows user to make change at one place. This is designed for advance user and keeps the backward compatibility with previous IDE (V2.x and V3.x) Setup Variable screen menu.


System
Filter
Clear
Command
Value
Undo
Sys.BgSleepTime
0
Sys.BgWDTReset
0
Sys.CompEnable
0
Sys.CompMotor
0
Sys.CPUTimerIntr
$0
Sys.FirstEnc
0
Sys.MaxCoords
16
Sys.MaxEcats
1
Sys.MaxMotors
32
Sys.MaxRtPlc
0
Sys.ModbusServerEnable
0
Sys.PhaseCycleExt
0
Sys.PhaseOverServoPeriod
0.25
Sys.PreCalc
1
Sys.RtIntPeriod
0
Sys.SendFileMode
0
Sys.ServoPeriod
0.442742110000000022
Sys.WDTReset
0
Description
Max # of coordinate systems that can be established (0 to n-1)
Range
1 .. 128
Units
count
Default
N/A
System
CPU Management
Update Period 100 ms

Hardware

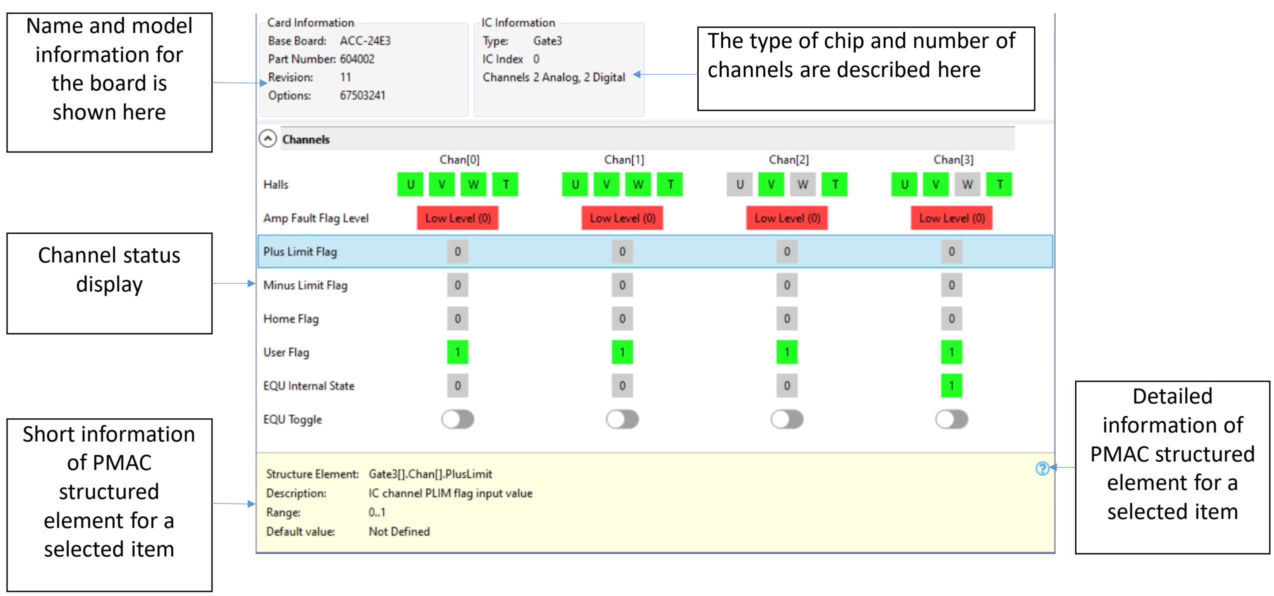
This folder contains the Hardware Diagnosis part of the System Setup. The System Setup displays various parameters associated with the accessory cards in the system in a graphical manner in order that parameters may be adjusted for setup or diagnosis purposes.
Right click the Hardware Node to refresh the card list as shown below. This menu is available for IDE version 4.3.2.x and above.



Axis Interface Cards

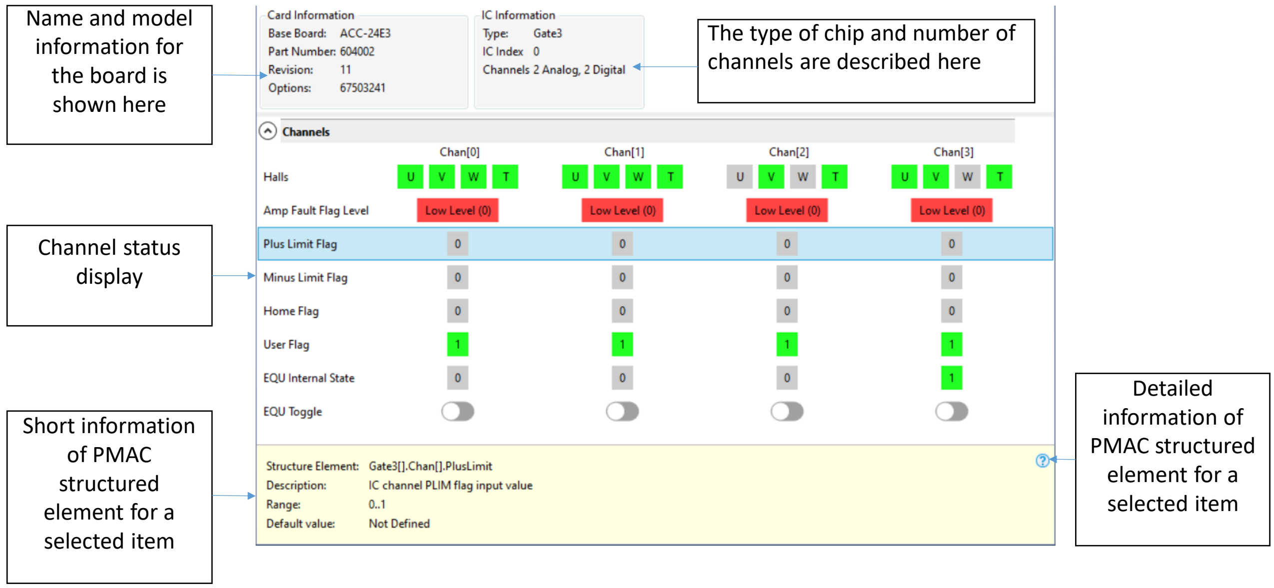
The following are the supported Axis interface cards: For example the ACC-24E3's Hardware Diagnosis page appears as follows:

ACC-24E3:



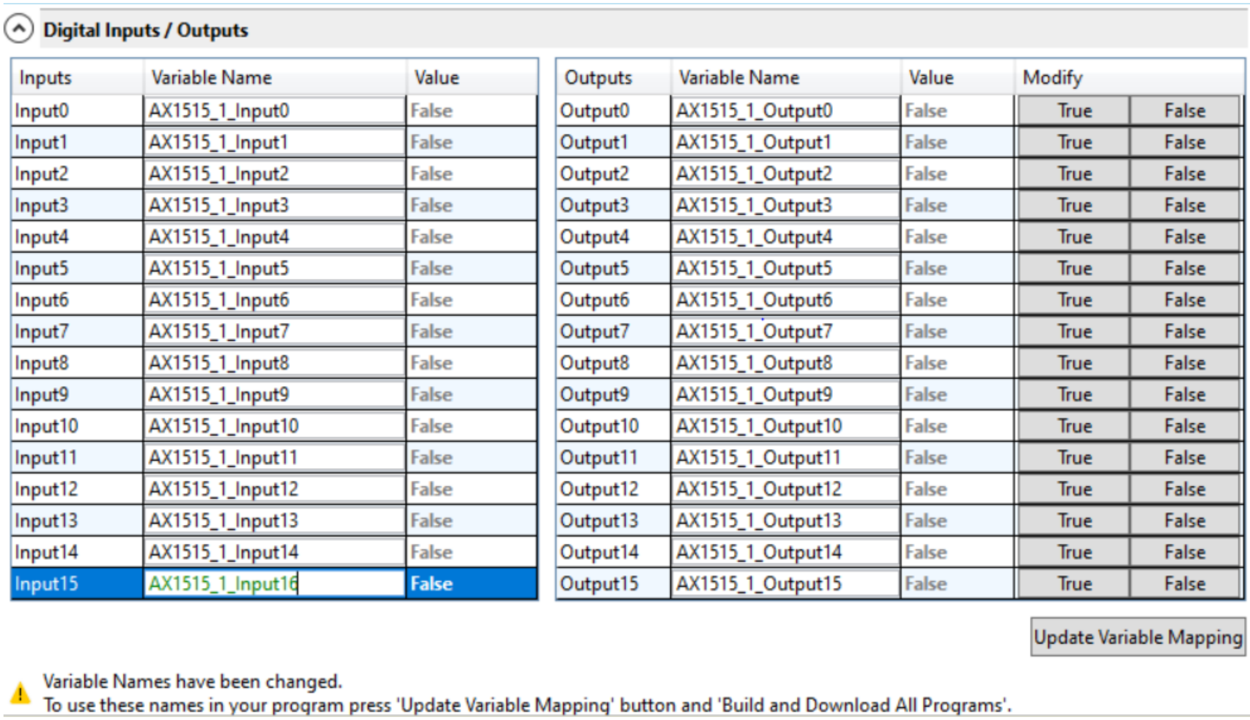
The Acc-24E3 page displays the card information and the Channel Status. The channel status will toggle between red and green and 1 / 0. The EQU state can be toggled by the EQU toggle switch. Each row of this page can be selected and the information about the selected row will appear at the bottom of the page. This information is a short description of the selected item structure element. To get more detailed information about the selected item, the user can click on the icon, which will open up a help manual.

CK3W-AX1515:



The CK3W-AX1515 page displays the card information, the Channel Status and the Digital Inputs / Outputs. The above picture represents the hardware information and channel status. Channel status will toggle between red and green and 1 / 0. The EQU state can be toggled by the EQU toggle switch. Each row of this page can be selected and the information about the selected row will appear at the bottom of the page. This information is a short description of the selected item structure element. To get more detailed information about the selected item, the user can click on the icon, which will open up a help manual.

The Digital Input / Output section of this page allows the user to toggle the outputs and also see the status of the input. It also allows the user to change the predefined Input and output variable names. The default variable names are also added to the AX1515_[Index]_Mapping.pmh file located in the Global Includes folder. The [Index] will be replace by the actual index of the Hardware card. The user can change the default variables, and after the variables are changed the user should click on the Update Variable Mapping button to update the AX1515_[Index]_Mapping.pmh with modified definitions.



Digital Inputs / Outputs

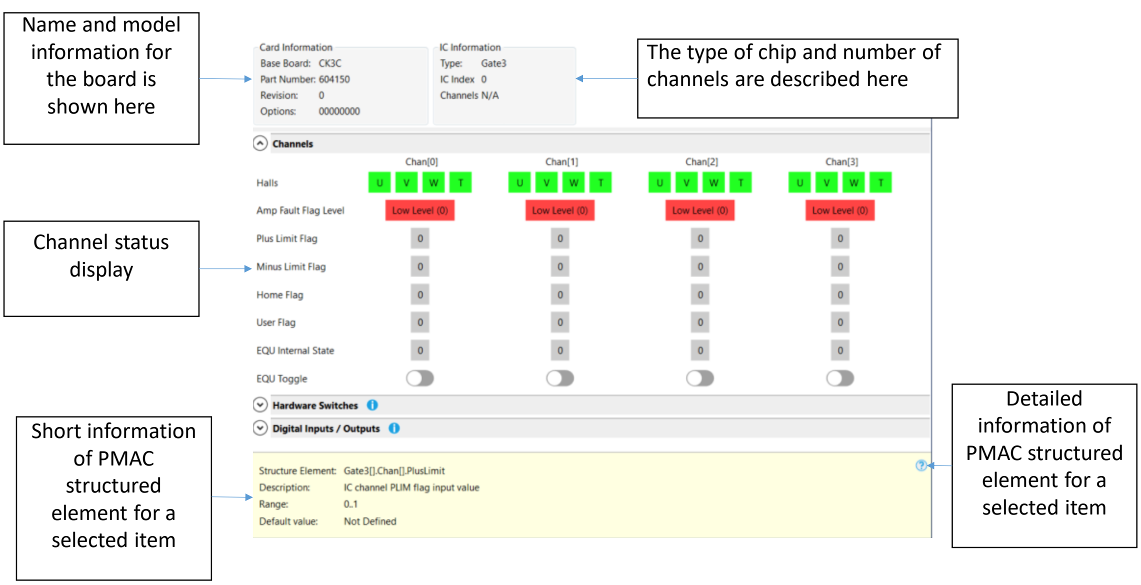
Inputs	Variable Name	Value	Outputs	Variable Name	Value	Modify	
Input0	AX1515_1_Input0	False	Output0	AX1515_1_Output0	False	True	False
Input1	AX1515_1_Input1	False	Output1	AX1515_1_Output1	False	True	False
Input2	AX1515_1_Input2	False	Output2	AX1515_1_Output2	False	True	False
Input3	AX1515_1_Input3	False	Output3	AX1515_1_Output3	False	True	False
Input4	AX1515_1_Input4	False	Output4	AX1515_1_Output4	False	True	False
Input5	AX1515_1_Input5	False	Output5	AX1515_1_Output5	False	True	False
Input6	AX1515_1_Input6	False	Output6	AX1515_1_Output6	False	True	False
Input7	AX1515_1_Input7	False	Output7	AX1515_1_Output7	False	True	False
Input8	AX1515_1_Input8	False	Output8	AX1515_1_Output8	False	True	False
Input9	AX1515_1_Input9	False	Output9	AX1515_1_Output9	False	True	False
Input10	AX1515_1_Input10	False	Output10	AX1515_1_Output10	False	True	False
Input11	AX1515_1_Input11	False	Output11	AX1515_1_Output11	False	True	False
Input12	AX1515_1_Input12	False	Output12	AX1515_1_Output12	False	True	False
Input13	AX1515_1_Input13	False	Output13	AX1515_1_Output13	False	True	False
Input14	AX1515_1_Input14	False	Output14	AX1515_1_Output14	False	True	False
Input15	AX1515_1_Input16	False	Output15	AX1515_1_Output15	False	True	False

Update Variable Mapping

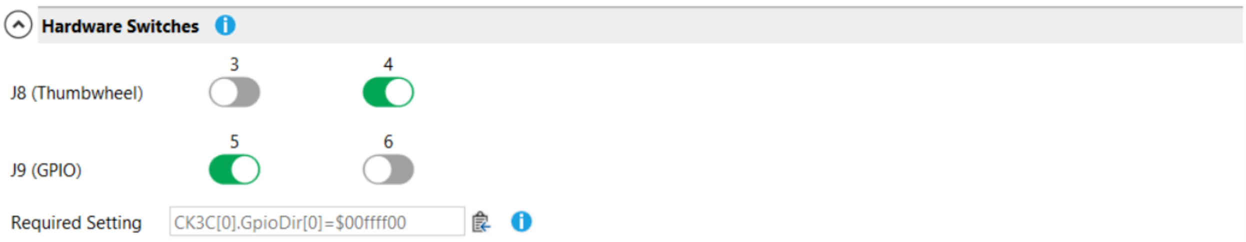
Variable Names have been changed.
To use these names in your program press 'Update Variable Mapping' button and 'Build and Download All Programs'.

CK3C:

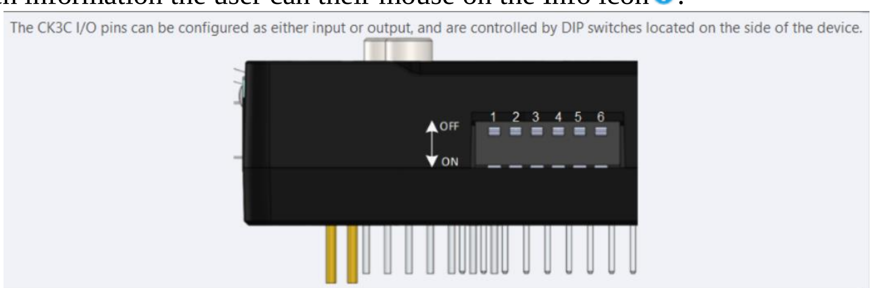
The CK3C page displays the card information, the Channel Status, Hardware switch sections and the Digital Inputs / Outputs. The picture below represents the hardware information and channel status. Channel status will toggle between red and green and 1 / 0. The EQU state can be toggled by the EQU toggle switch. Each row of this page can be selected and the information about the selected row will appear at the bottom of the page. This information is a short description of the selected item structure element. To get more detailed information about the selected item, the user can click on the icon, which will open up a help manual.



The Hardware Switched section allows the user to match the CK3I/O dip switch pins to properly display the number of configured Inputs and Outputs based on the selection. The dip switch pins are located at the side of the device.



To see the switch information the user can their mouse on the Info icon.



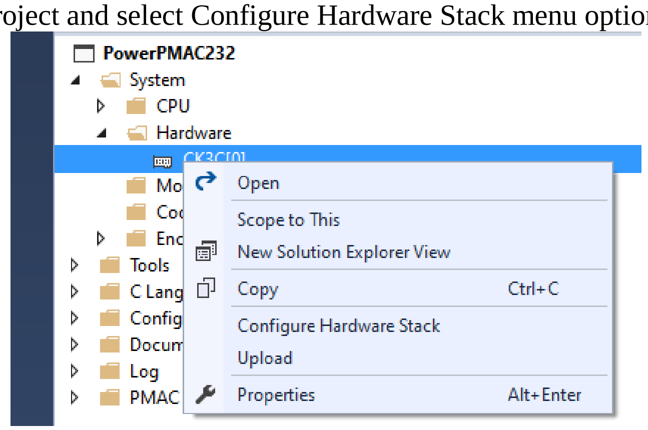
The following table shows the configured pins and the available Inputs and Outputs:

IDE V4.2 onwards the Hardware Card's will be displayed as detected and not in Alpha/numeric order.

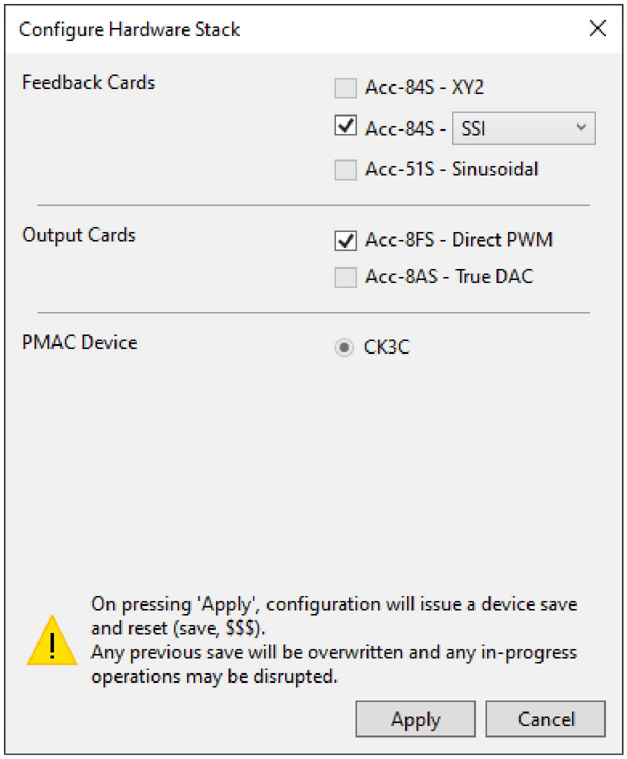
Note

Configuring CK3C Hardware Stack:

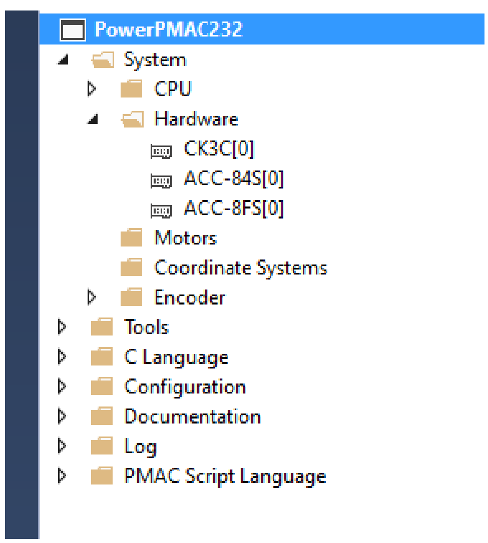
To configure the CK3C hardware stack, the user should right click on the detected CK3C hardware in the Hardware folder of the project and select Configure Hardware Stack menu option.



The Hardware stack configuration page will allow the user to select one of the available hardware accessories supported by the CK3C.

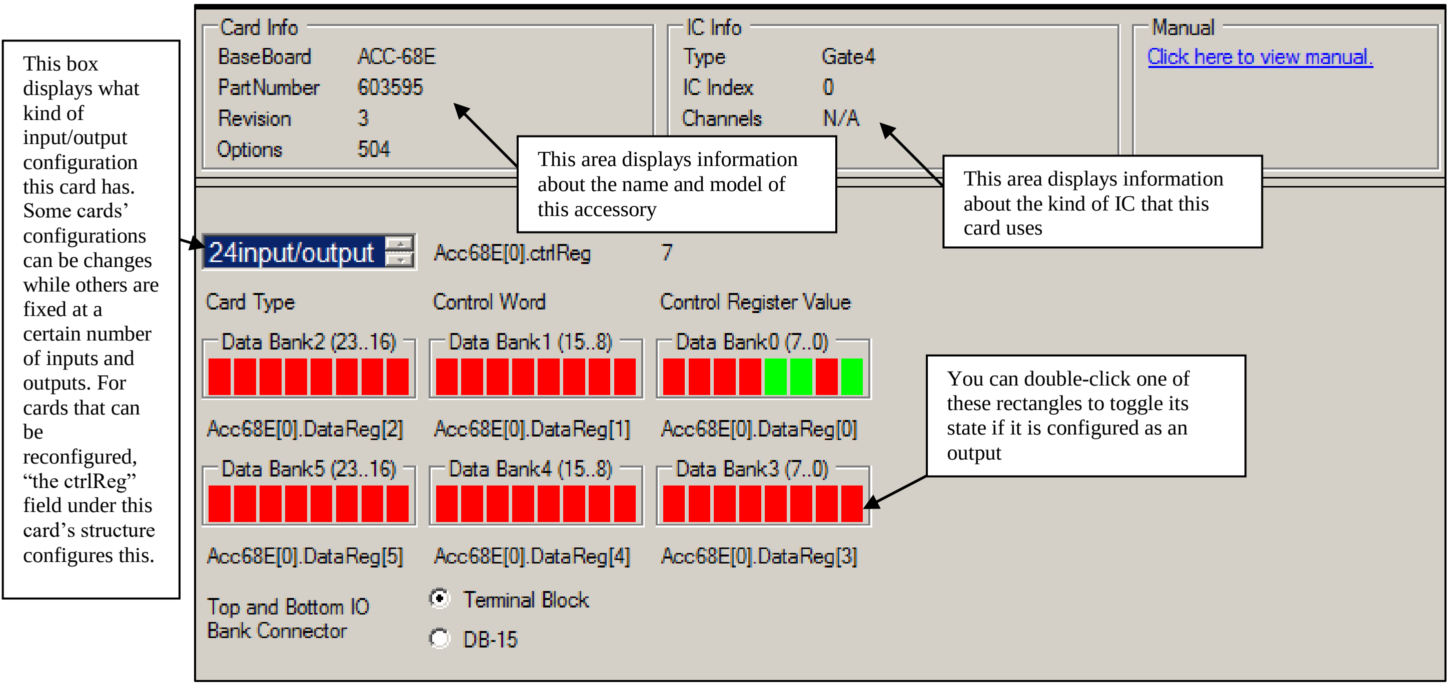


Once the user selected any accessory card, the user should apply for configuration for the changes to apply. The selected accessory cards will be added to the hardware folder.



Digital I/O Cards

The Hardware Diagnosis page for digital I/O cards is much simpler than for axis interface cards. This screen shows the data registers in the card displaying each bit in each bank with a green light if the bit is high or with a red light if the bit is low:

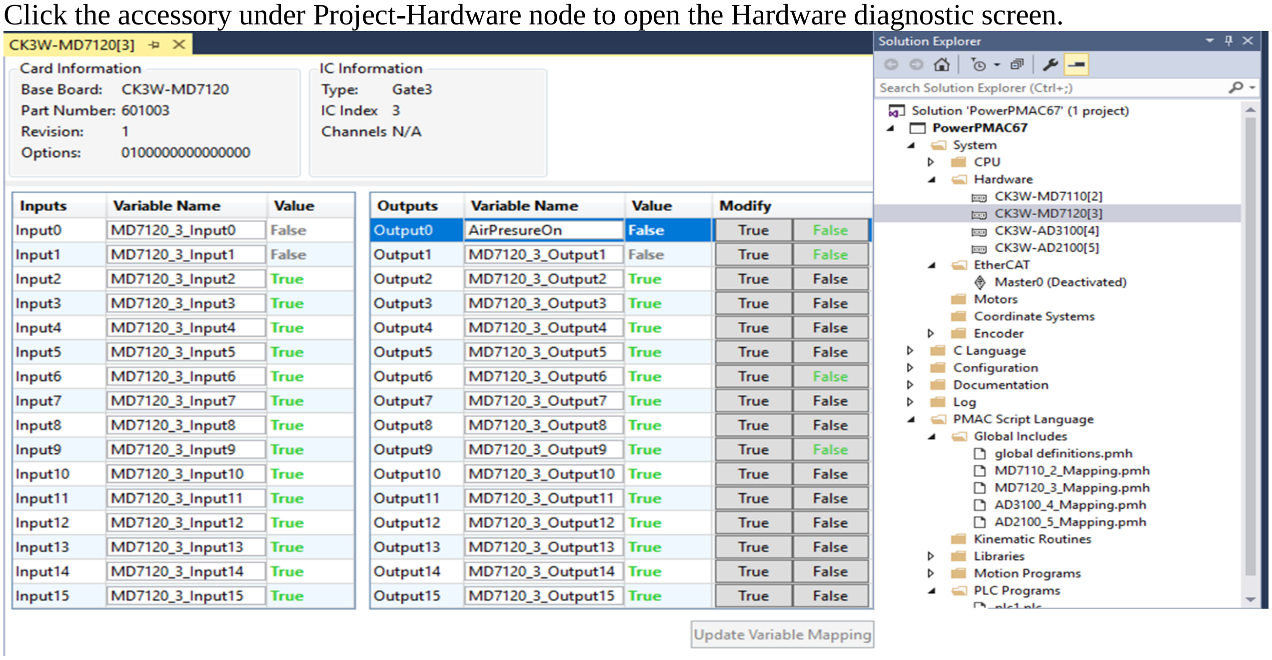


MD71xx

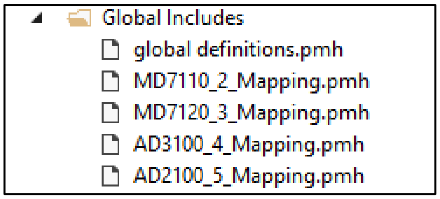
CK3M IO accessories the screens are more aligned with SYSMAC Studio.

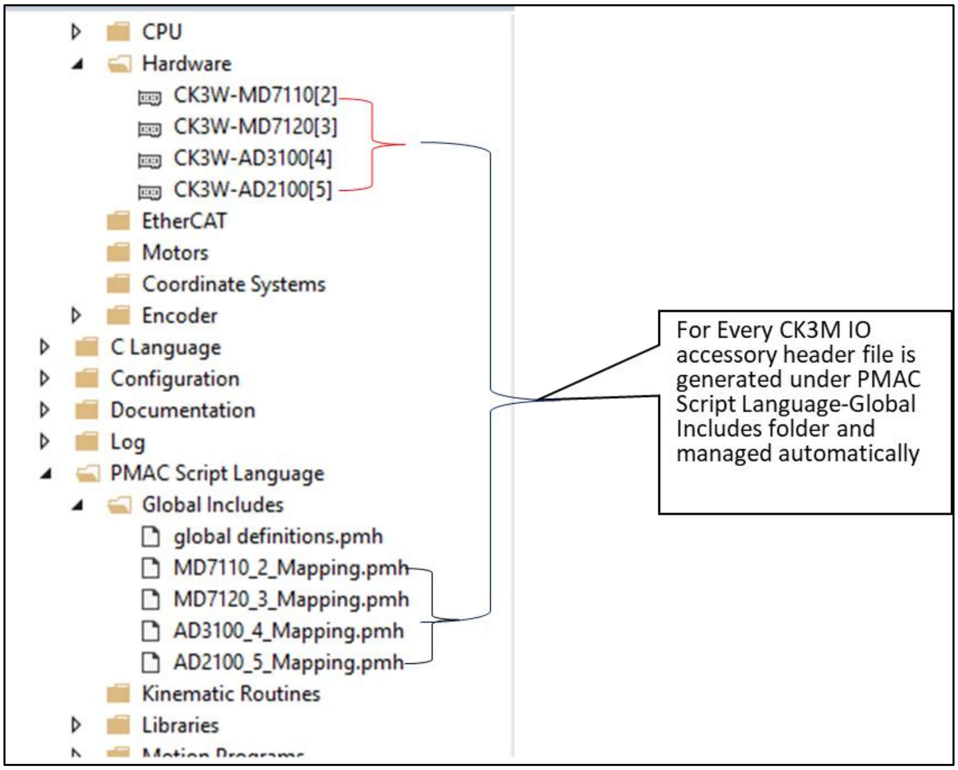
There are two digital output accessories MD7110 and MD7120. User can Modify Output and read Input.

Click the accessory under Project-Hardware node to open the Hardware diagnostic screen.



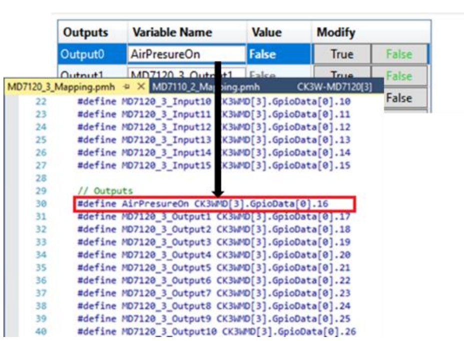
When the accessory is detected header file will be automatically added to the project under Global Includes.





User can change the Name of the Input or Output. For example, user can change the name from Input0 to Switch0 and from Output0 to AirPressureOn. The change in these names are associated with the file under Global Includes. After the name is change select "Update variable mapping" to regenerate respective file.

Project intellisense will show the names of these variables that can be used in PLC or Motion or C code.

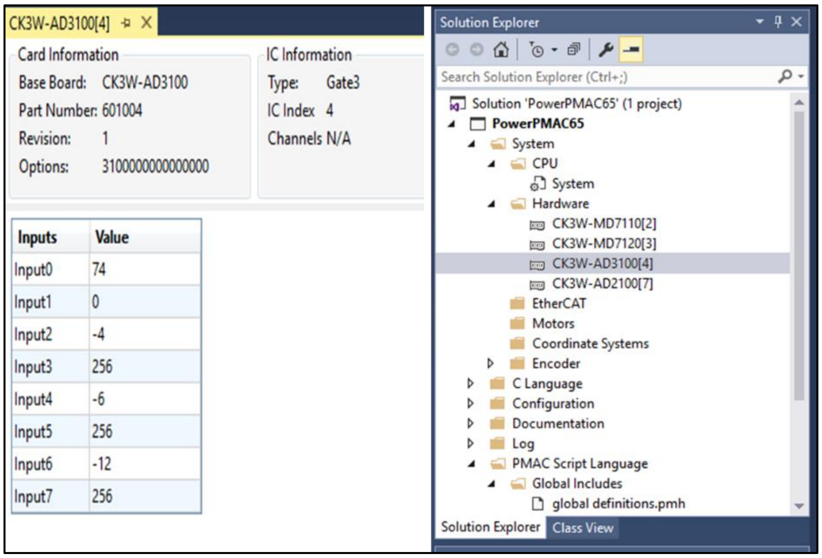


Note

Please do not modify the MD7xxx or AD2xxx file. These files are maintained by IDE.

AD31xx

The Hardware Diagnosis Screens for CK3M Analog accessory AD3100 and AD2100 are accessed by clicking the accessory under Project-Hardware node to open the Hardware diagnostic screen.



When the accessory is detected a header file will be automatically added to the project under Global Includes. The User can change the Name of the Input or Output. For example, the User can change the name from Input0 to Voltage0 and so on. The change in these names are associated with the file under Global Includes. After the name is changed, select “Update variable mapping” to regenerate respective file.

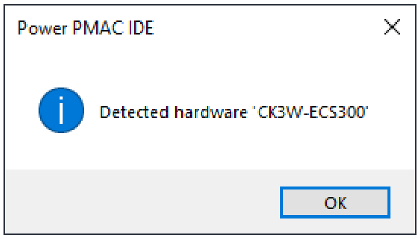
Project intellisense will show the names of the variables that can be used in PLC or Motion or C code.

Note

Please do not modify the MD7xxx or AD2xxx file. These files are maintained by IDE.

CK3WECSxxxx

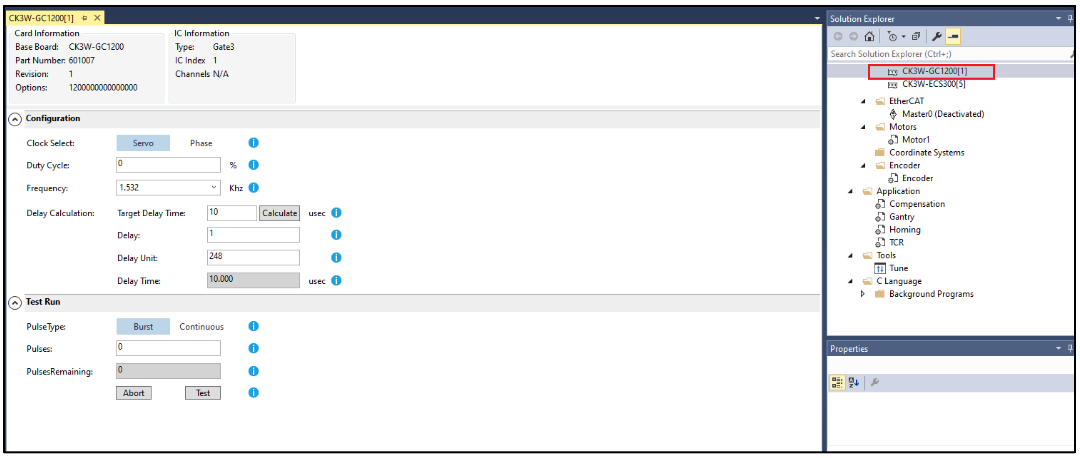
This hardware does not have the diagnostics screen but used in the Motor topology. On double clicking it will give following message.



CK3WGCxxxx

The Hardware Diagnosis/configuration Screens for CK3WGC accessory is accessed by clicking the accessory under Project-Hardware node.

It shows like this..



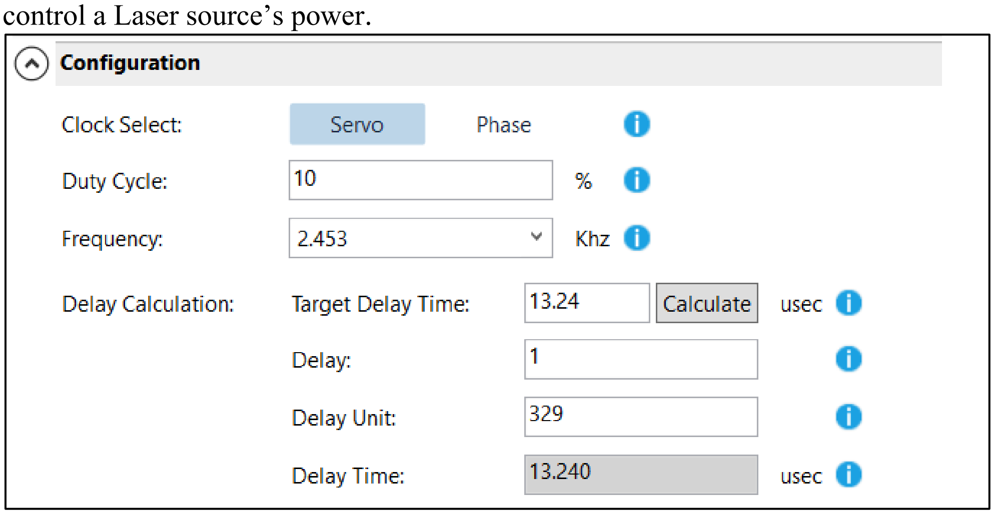
The top section shows the detected hardware card and option.

Using this user can setup and diagnose PWM output. It is done in two steps Configuration and Test Run.

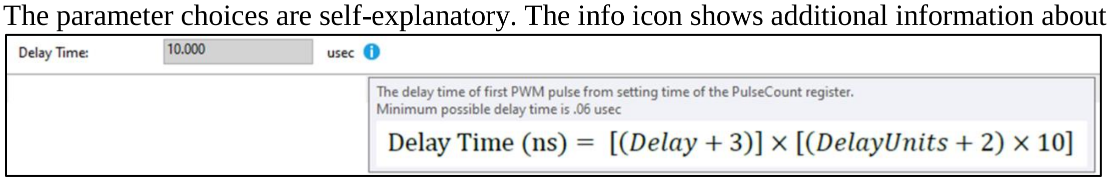
Once testing complete, user can press Accept and it generates Ck3WGC Definitions.pmh file under Global Includes that can be used in Motion program and plc script.

Configuration

Configuration section looks like this.. This is for configuring PWM output. PWM output is intended to control a Laser source's power.



The parameter choices are self-explanatory. The info icon shows additional information about parameter.



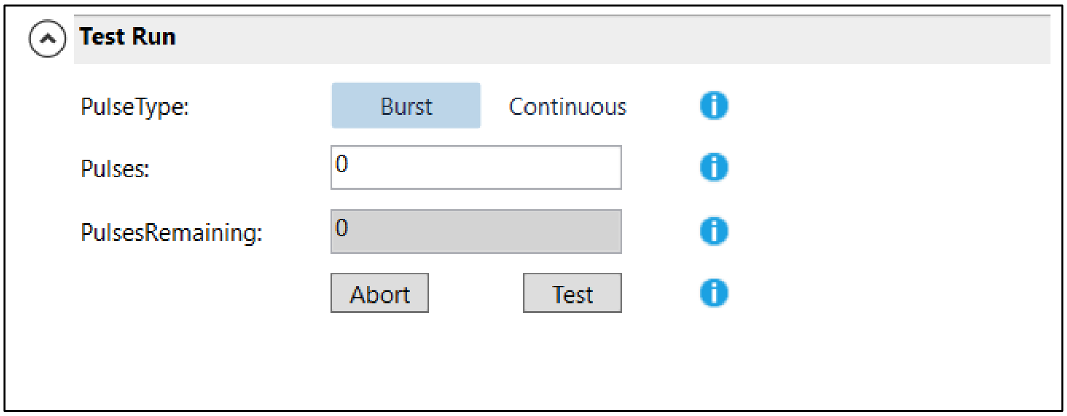
User have choice of entering Target Delay Time in usec and then press calculate to fill Delay and Delay unit box or enter the Delay and Delay unit to get Delay Time in ns.
PWM frequency (Frequency) I drop down list calculated using following formula…

$$f_{PWM}(\mathrm{kHz}) = \frac{10^5}{16 \times PWMPeriod}$$

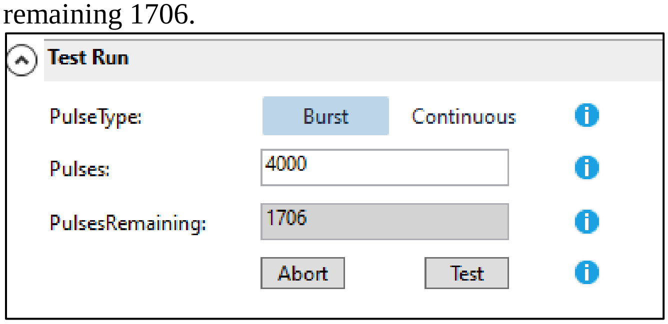
User can start typing and the list will filter based on the input.

Test Run

Test Run section looks like..



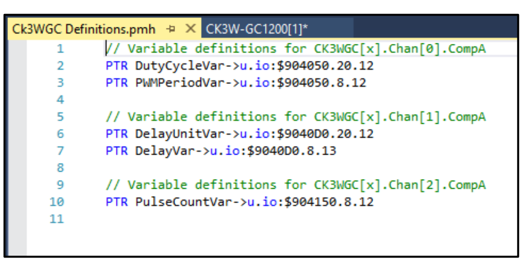
As the section name this will allow user to test the configuration setting. On pressing Test it will configure the card settings and depending on type of the Pulse the user can verify output. If the burst mode selected then on Test it will show you Pulsesremaining as shown below…Asked for 4000 and remaining 1706.



The second option is [Continuous], which does not allow the user to enter a number. When Continuous is selected and when Test is pressed, the number inside PulsesRemaining is expected to become 4095 and not decrease until the user selects the Abort button.

Accept

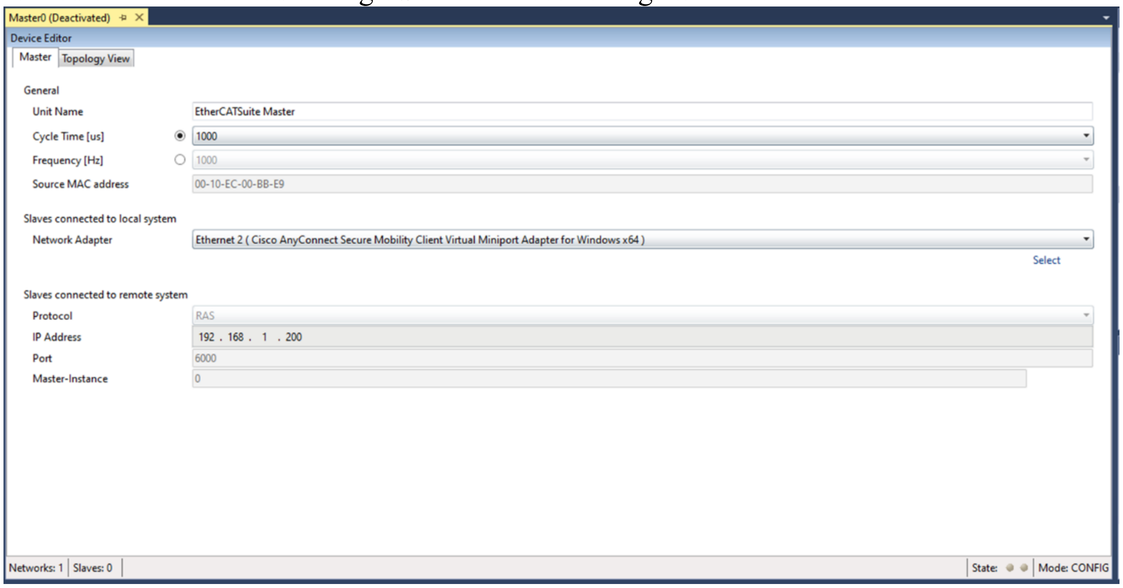
On accept generates Ck3WGC Definitions.pmh file under Global Includes that can be used in Motion program and plc script. The file contains PTR variable for read and write the hardware register of the CK3WGC card.



EtherCAT

Master0

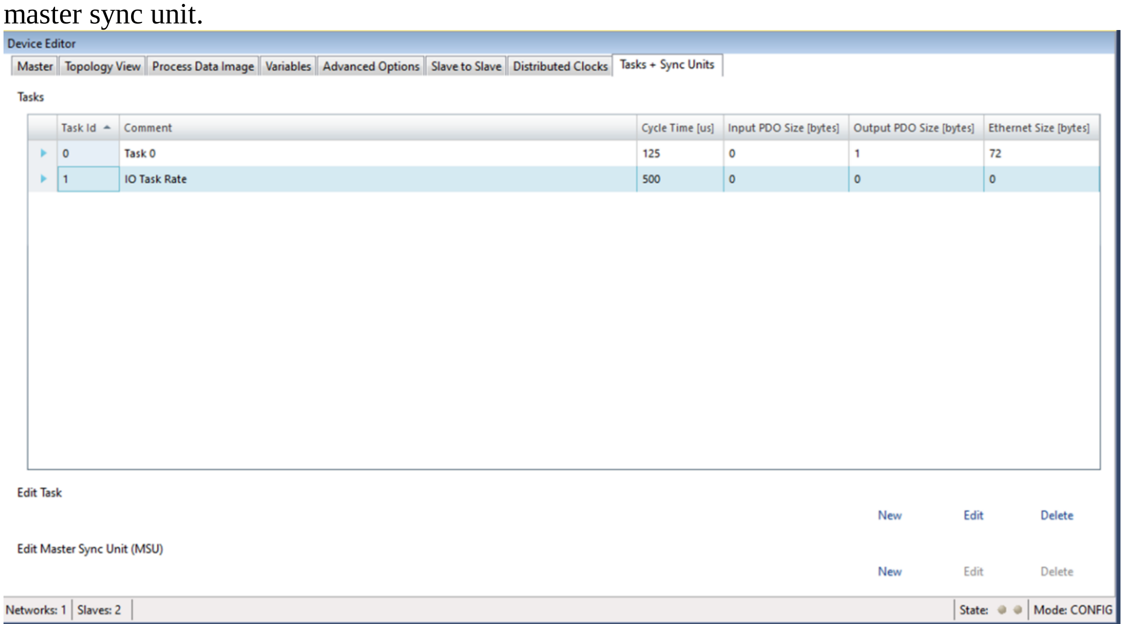
On clicking the Master node0 it will open the Master0 device editor. This section includes network related or master related settings. Some of those settings will also affect the “Master” section of the ENI.



Under General, user can choose to setup the ECAT clock either using time or freq mode. For 16 kHz clock user must select frequency mode.

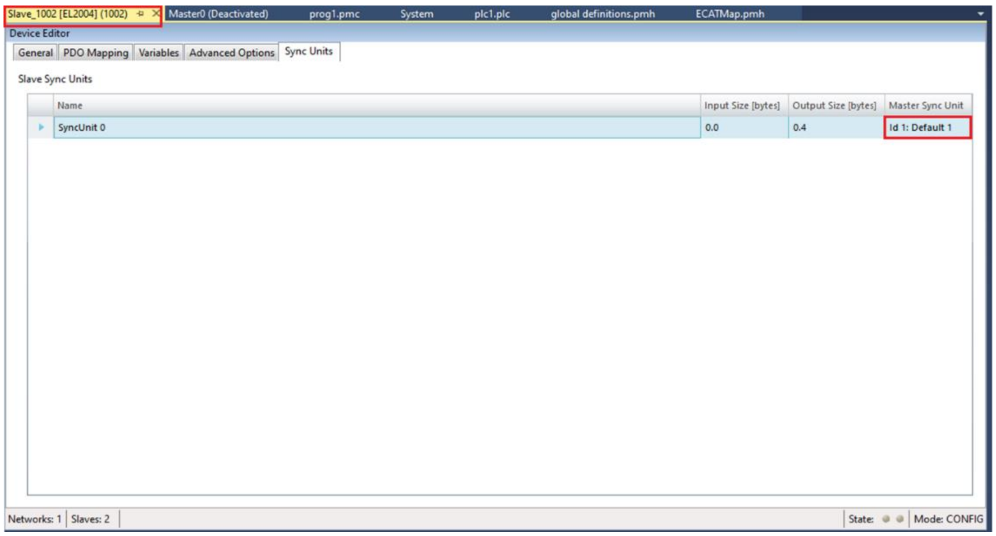
Tasks + Sync Units

In this tab, the user can define additional cyclic tasks and master sync units. After adding a new Master sync unit, the user can assign one or more slave sync units on tab “Slave -→ Sync Units” to this master sync unit.



As shown above a new Task IO Task Rate is added. From FW 2.6 a new EtherCAT structure element is added to handle the additional task. At this point only one additional task can be added per master.

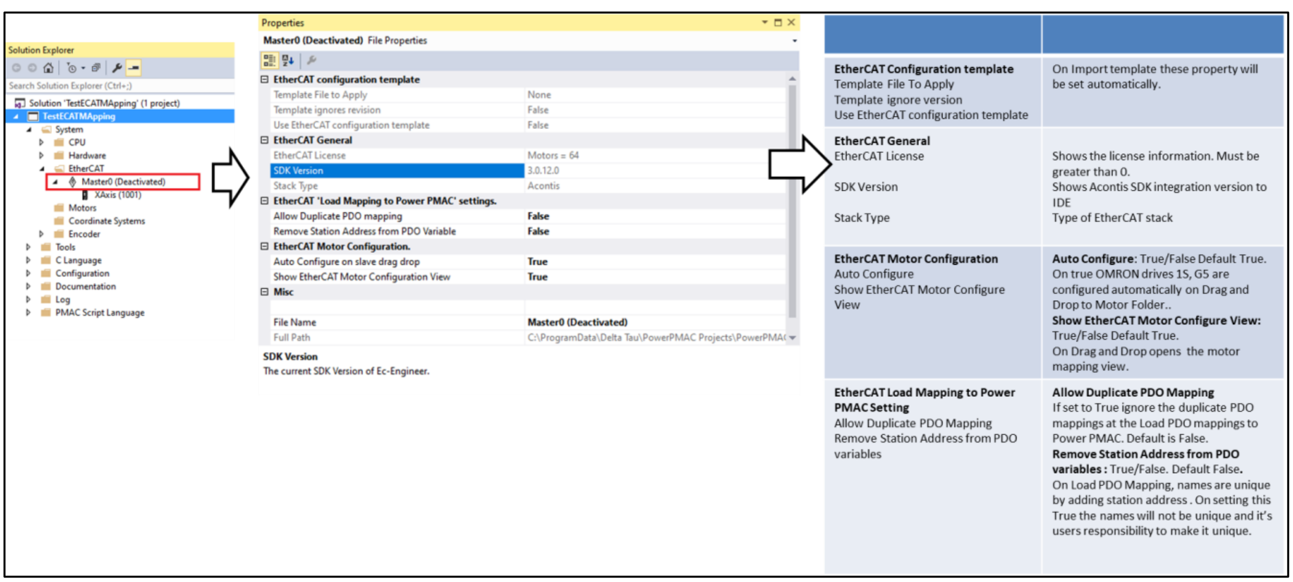
On adding the task, it will set ECAT[i].TaskID1Ext. User can verify the value in the Power PMAC Message – Output Tab.
Example: Typical use of the additional cyclic task….
In a high-performance application, the EtherCAT servo drives must be updated at 8 kHz (125 µsec) to get enough response. There is an I/O device on the network that cannot reliably be updated faster than 2 kHz (500 µsec). This I/O device is assigned to Task ID 1 and ECAT[i].TaskID1Ext is set to 3 so it is only updated every 4th cycle.
To use the new Cyclic task user can click on the slave that needs to assign the task and then select Sync Units Tab. (Slave--→Sync Units Tab) Under Master Sync Unit select the Id1 from drop down as shown below… Red square indicate the slave and the Master Sync ID.



EtherCAT Master-Node Properties

Master-Node Properties

On selecting the Master Node view the EtherCAT properties as shown below.

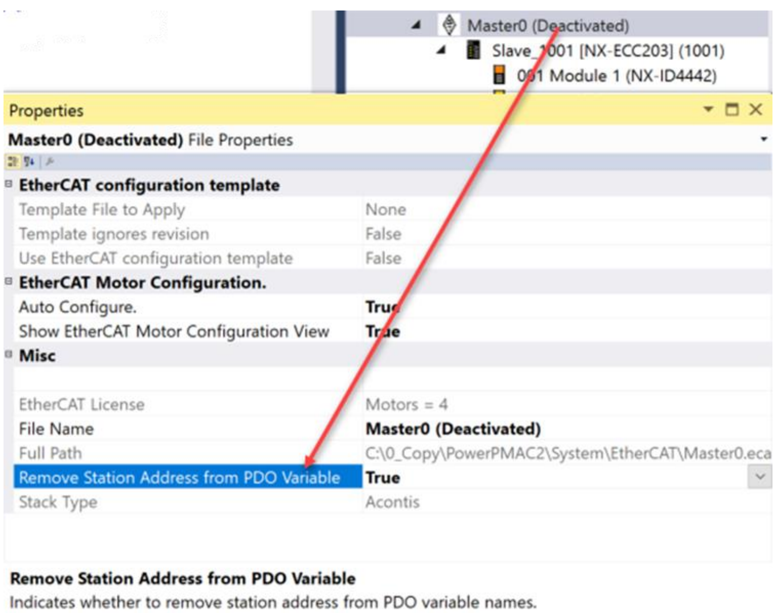


Allow Duplicate PDO Mapping
Default: False
If set to true then Load PDO mapping will ignore the duplicate mapping found. It will print found duplicate mappings with addresses in the Power PMAC Message window as Warning. If EtherCAT network is unable to activate this could be one of the reason.
In the default state Load PDO mapping will fail if it finds duplicate PDO mappings.

Remove Station Address from PDO Variable
Default: False
Note: need to have the Solution Property "***Use new PDO mapping name format***" set to **Yes**
- this is selected in the Solution - Property page of Explorer Tree (Rt-Clk on Solution name)
If set to **True** when PDO variable names are generated by IDE the slave station address will NOT be appended as part of the variable name. The slave name (assigned or custom edited) in the Device General Tab Name field will be used as the PDO variable name created in the ECATMap.pmh file.
EXAMPLE: below a drive was given the custom name of "Drive3"

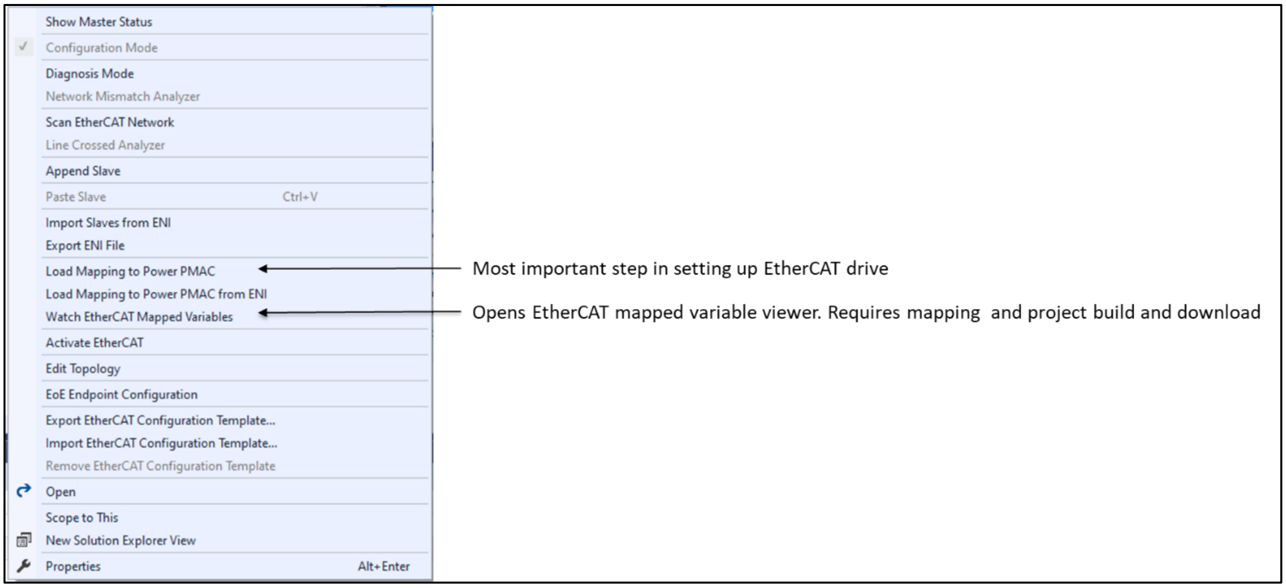
Before: `#define Slave_1002_R88D_1SN01L_ECT_6040_0_Controlword ECAT[0].IO[16].Data`

After: `#define Drive3_6040_0_Controlword ECAT[0].IO[12].Data`



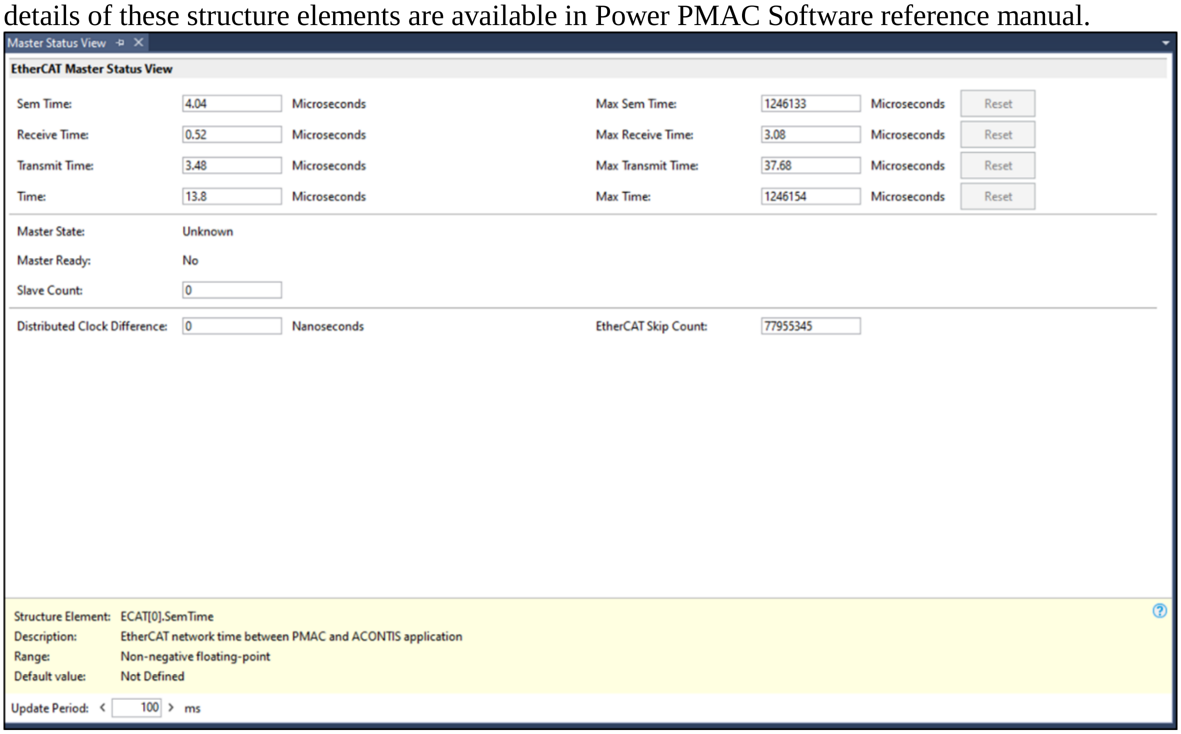
EtherCAT Master-Node Context Menu

The EtherCAT folder stores the EtherCAT master and Slave information. Use the context menu to configure and setup an EtherCAT network. The context menu is self explanatory.



Show Master Status

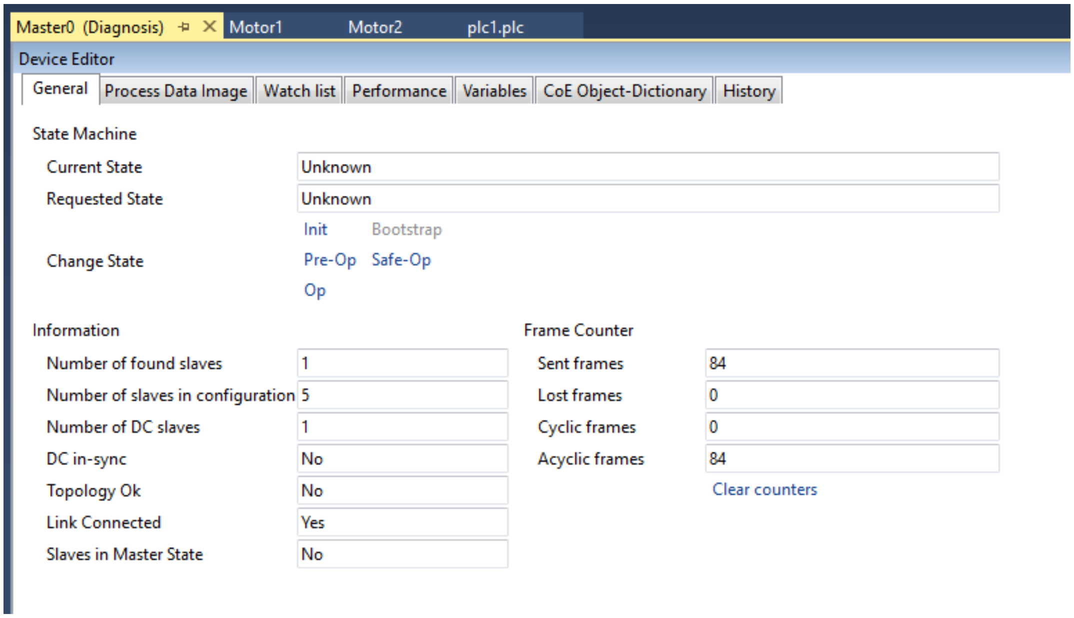
This menu will open Master status view, displaying important network registers. The bottom info ribbon will display respective structure element, description, range and default value. The default update period is 100 msec. that can be change. For measurement purpose user can reset all the Max. Time settings. The details of these structure elements are available in Power PMAC Software reference manual.



Diagnosis Mode

(Ref: EC-Engineer manual)

This mode is available to analyze EtherCAT networks that are controlled by the Acontis Master Stack. Automated control systems usually require high availability of the whole system and, due to the rough industrial environment, this is often hard to achieve.
High availability should be guaranteed for an automated control system so it is important to verify and maintain the field bus. In this mode it is possible to look into the "health" of the EtherCAT system. Detection of signs of system degradation prior to running into a system failure will be of great benefit. In that case it is possible to exchange the problematic components (cables, slave devices).
When the Diagnosis mode is selected it will detect the devices on network. The screen will look like this:



The General Tab displays information on the current state of the state machine of the master which can be modified.

The Process Data Image Tab displays information on the process variables which can be modified. The variables will be forced to the value the entered. The user can press the release button to stop forcing the user-entered value to the variable. Selecting a process variable will show a chart of the values. This chart will be updated every 250 milliseconds.

The Watch list enables the monitoring of selected variables.

The Performance Tab displays information like the busload per cycle and per second.

The Variable Tab displays information on the the trace variable which can be modified. The chart will update every 250msec.

The CoE Object-Dictionary Tab displays information on the values of the object dictionary of the master which can be modified.

The History Tab contains the diagnosis history.

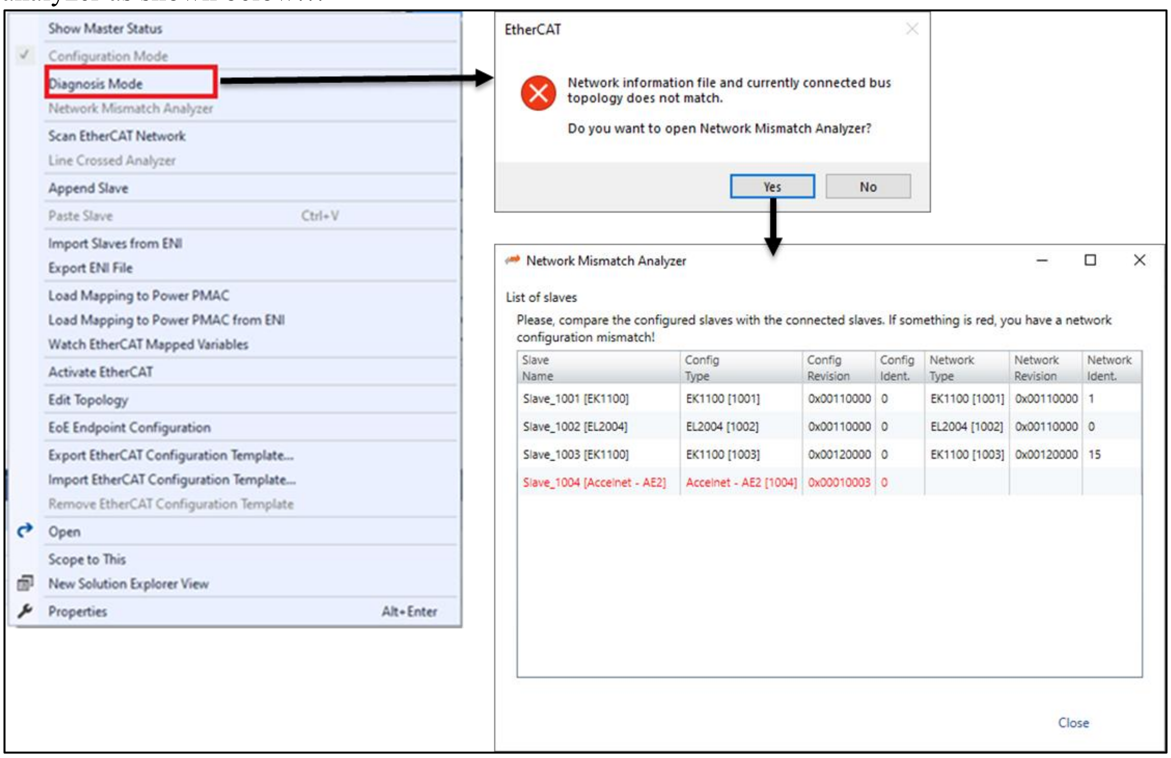
Network Mismatch Analyzer

This option is useful when there is a mismatch between the eni file from the project and actual devices on the network. It is difficult to figure out where the mismatch is and for this this tool is useful that identifies mismatch.

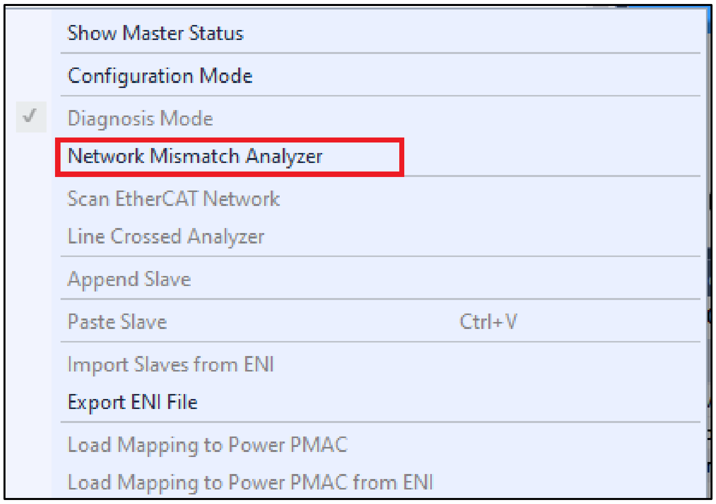
This option only active if the Diagnosis mode is active. If user switches to Diagnosis mode and gets the following error …

0x9811001E EC_E_BUSCONFIG_MISMATCH	Bus Config Mismatch	ENI	Network information file and currently connected bus topology does not match.

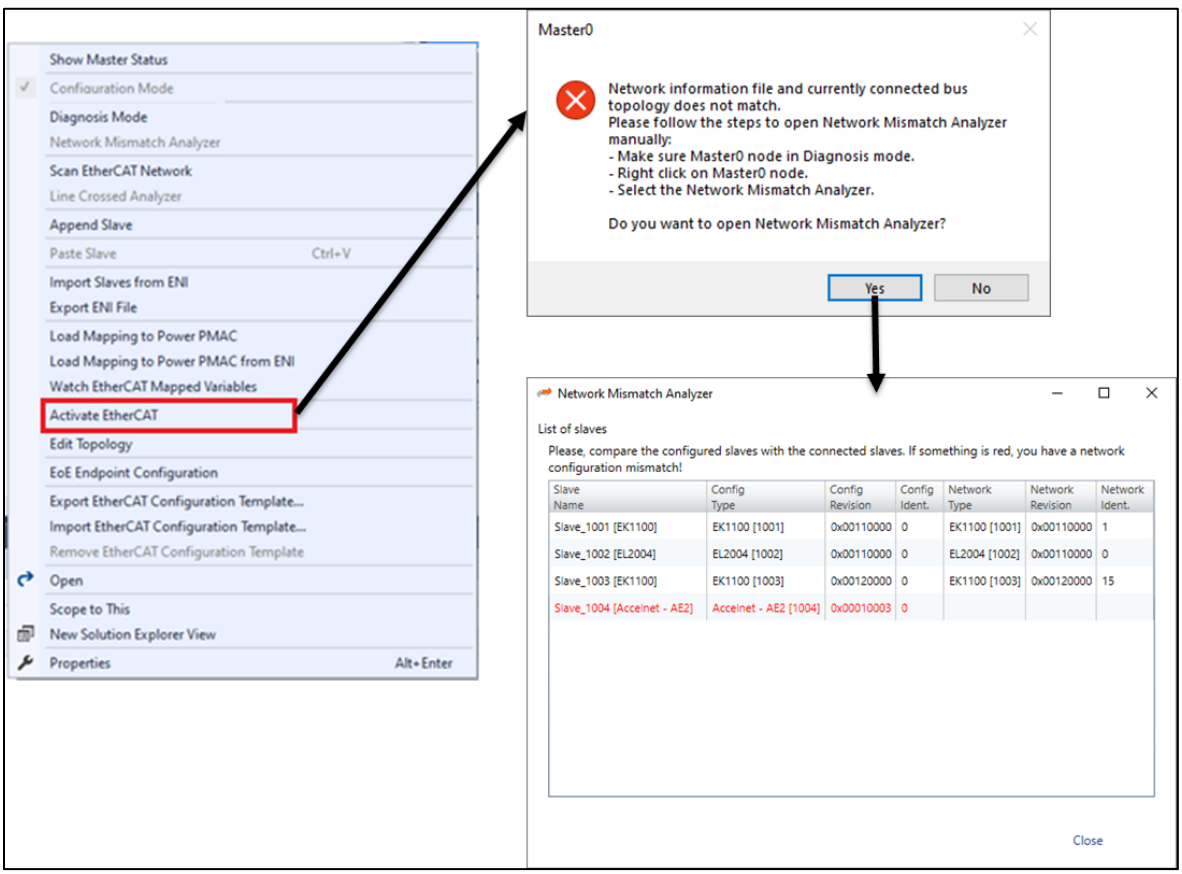
The diagnosis option will ask to open Network Mismatch analyzer. On selecting OK it will open the analyzer as shown below…



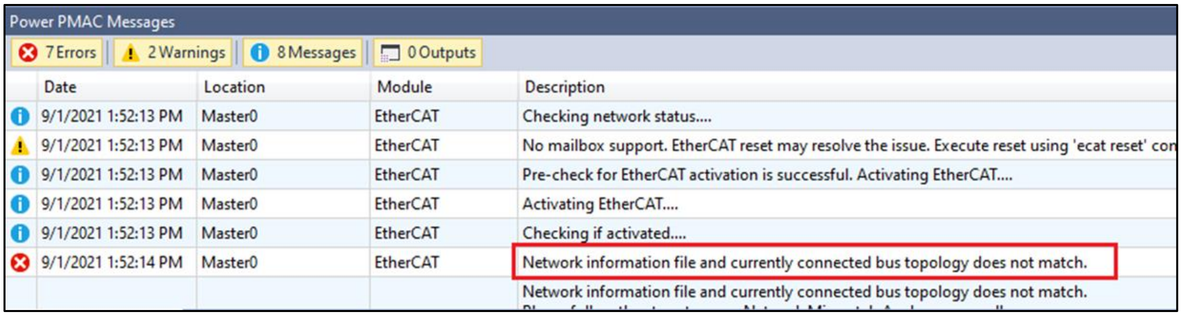
If 'No' option selected then user will need to open the analyzer manually using the Network Mismatch Analyzer option from Master Node Context menu as shown below…Please see it is enabled only in Diagnosis mode.



The second way of opening the Mismatch Analyzer is when user uses Activate EtherCAT option to enable EtherCAT network and if there is a mismatch between eni and actual devices then software will capture the bus mismatch error and will ask user to open the mismatch analyzer automatically as shown below work flow….



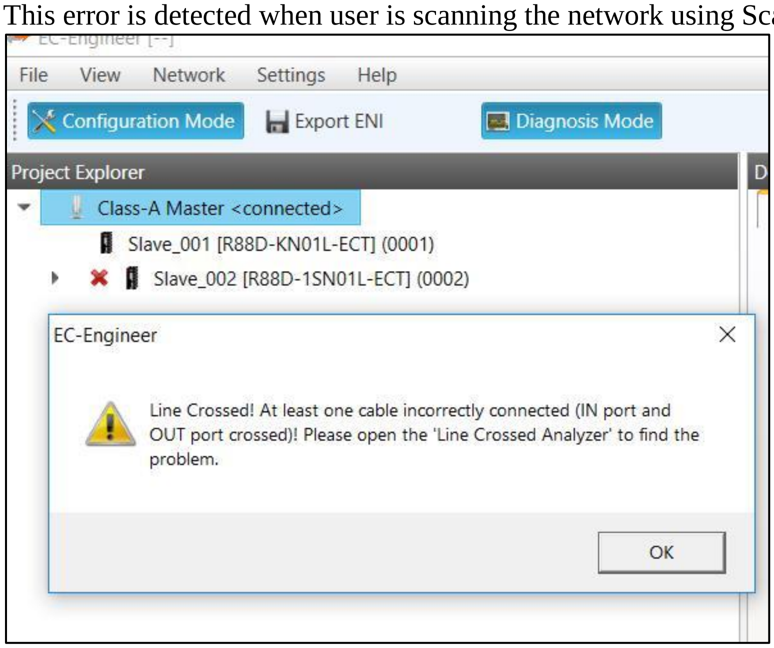
The error is also displayed in the Power PMAC message window like this…



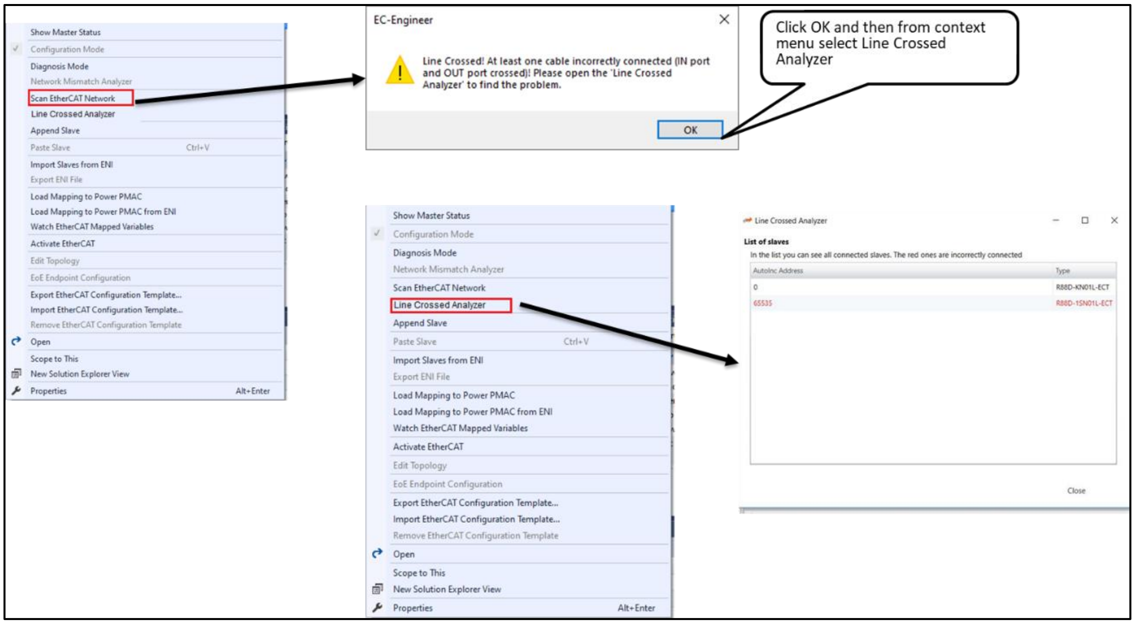
Line Crossed Analyzer

If user have connected a line to a wrong port, you can see in the Line Crossed Analyzer which slave is Incorrectly connected. The wrong entries will be red. It is very difficult to identify wrong connection on a bigger network. This tool is useful for identifying. If there is a line error pop up message will be displayed like this and then user can open the Line cross analyzer.

This error is detected when user is scanning the network using Scan EtherCAT Network context menu.



Here is the typical workflow …



Scan EtherCAT Network / Append Slave

Select Scan EtherCAT network it will issue scan command and detect the connected EtherCAT devices on the netwrok. If the EtherCAT devices are not connected user can still configure EtherCAT network using Append Slave menu. When the slave devices are added to the Master, either using Scan or Append, then the Master node looks like this:

Note

From IDE V4.2 onwards ECAT devices will be displayed as the information is received from the scan/Append slave and not in Alpha/numeric order

Enable/Disable cable redundancy

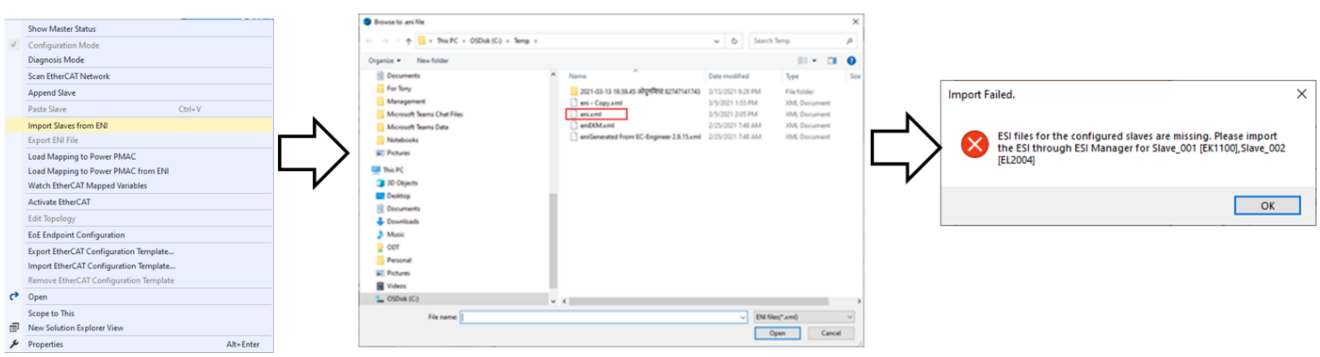
This feature will allow you to enable or disable the ethrecat cable redundancy feature. This command will set the Ecat[0].Redundancy structure element to 1 or 0 to enable or disable the cable redundancy.

Note

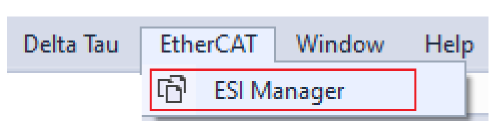
This feature is available starting in IDE 4.6.0.x and firmware version 2.7.0.0

Import Slaves from ENI

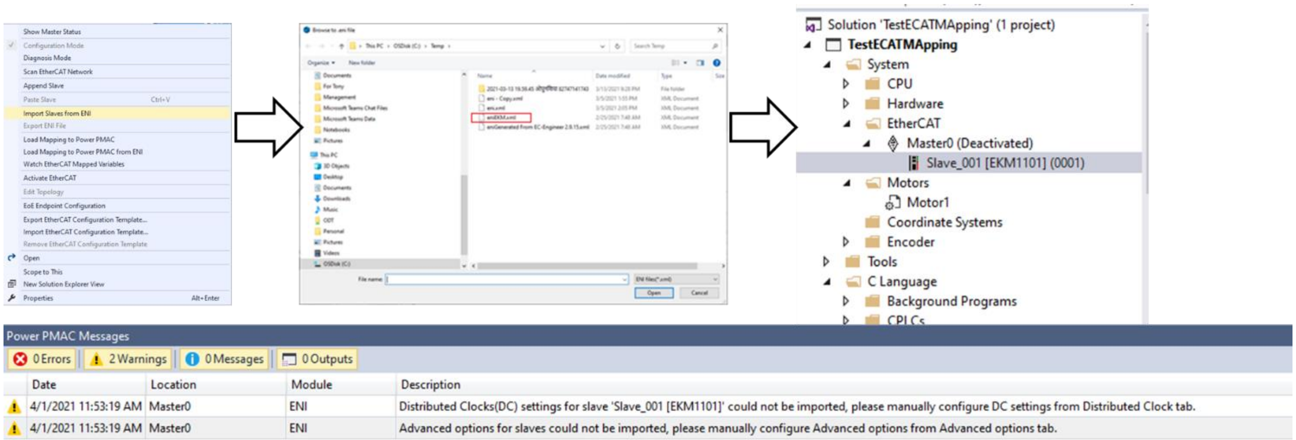
This feature allows you to import slaves from the eni file, that was either created with different eni tool generator or eni file was generated previously from the Power PMAC IDE software.
On selecting the menu it will open file selection dialog.
Case 1: Here is the example of import slave from eni file where the esi file is not present in the Power PMAC IDE. Under this case the error will be displayed requesting importing esi file using ESI manager from EtherCAT menu.



To access esi manger choose



Case 2: Here is the example of import slave from eni file where the esi file is present in the Power PMAC IDE system. Under this case the import slave will be successful and slave will be listed under master node.



Please check the warning message under Power PMAC Messages. There is one know limitation with import eni feature, it will not import distributed clock settings and advanced option settings from Slave Advanced tab if the slave was configured with these settings. The reason for limitation is specification of eni file do not store this information so it is not available.

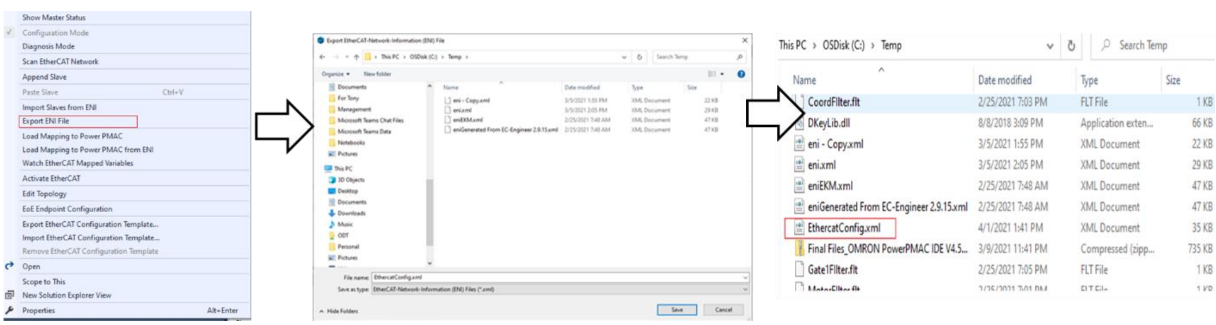
This feature is best if number of slaves importing from the file are less than 5!

Export ENI file

This option allows user to generate the eni file for available connected to the Master node and export it to folder. This menu option is helpful when you have generate the eni file and share with other users. After exporting the eni file you can again import it to the Power PMAC project as explained above.

The practical use of this feature is configuring the EtherCAT network without physical devices and share with other users.

Here is typical export process.

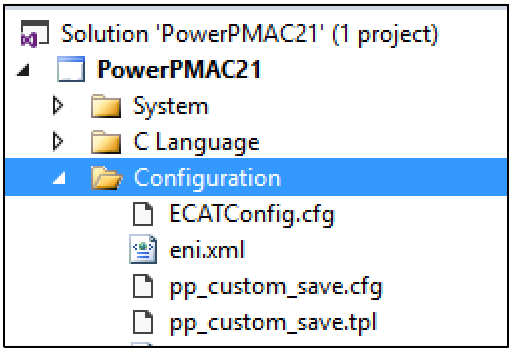


Load Mapping to Power PMAC

As the name says, this command read the mapped variables from currently connected EtherCAT device and generate following files and add it to the project.

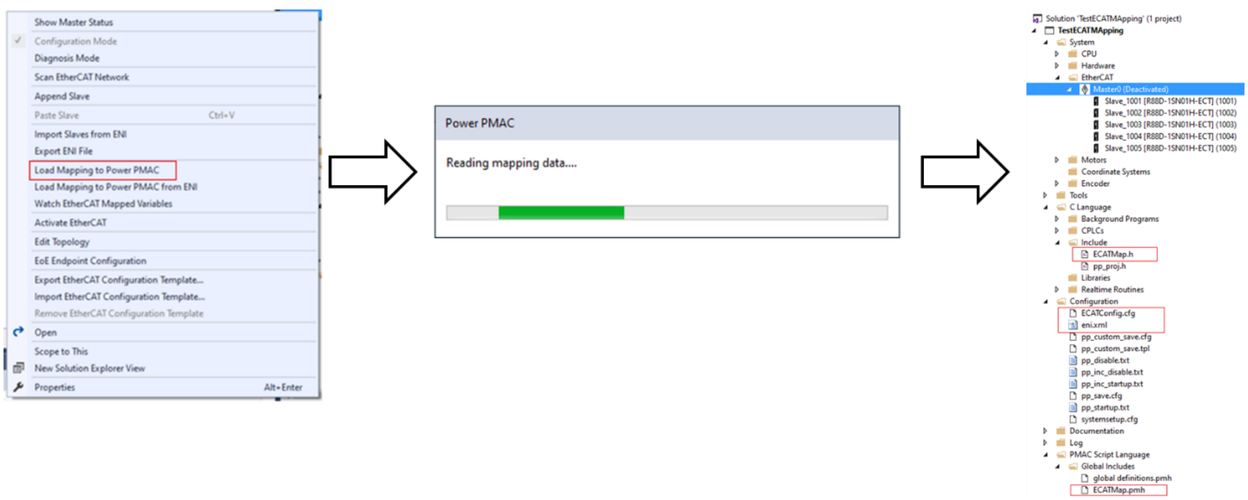
1. The eni.xml (EtherCAT network information) is generated and copied to the Project-Configuration folder. This file is copied to Power PMAC from Build and Download project process. On the Power PMAC after Build and Download, the files is placed under /var/ftp/usrflash/Project/Configuration folder.

2. The mapping file ECATConfig.cfg is created and copied to the Project-Configuration folder. This file is copied to Power PMAC from Build and Download project process. On the Power PMAC after Build and Download, the files is placed under /var/ftp/usrflash/Project/Configuration folder.

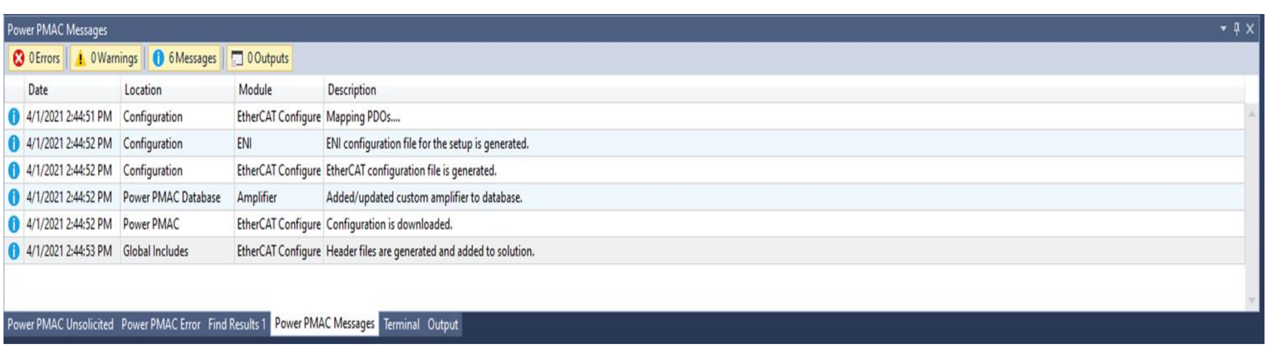


3. The ECATMap.pmh and ECATMap.h files are created and copied to the Power PMAC Script Language-Global Includes and C Language-Includes folders for use in C app and script languages. These header files consist of #defines values to access ECAT mappings in C app or script languages.

Typical flow will look like this…



On selecting Load mapping to Power PMAC, the process indicates its progress by showing a dialog and a message in the Power PMAC message box.

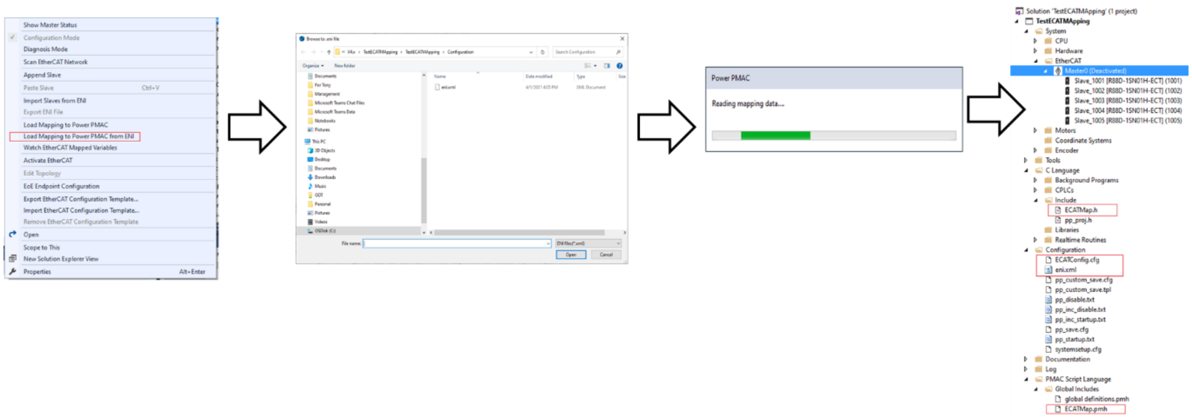


Note

1. It is not necessary to copy the EtherCAT files manually to the project like in V2.x and V3.x; V4.x automatically manages these files.
2. EtherCAT header files collapse/expand feature is available in te IDE 4.3.2.x and above

Load Mapping to Power PMAC from ENI

Similar to Import slave from eni this option allows user to generate mapping from eni and add it to the project similar to Load Mapping to Power PMAC. The process is identical to “Load Mapping to Power PMAC” except user will need to input the eni file from file dialog.



Export EtherCAT Configuration Template

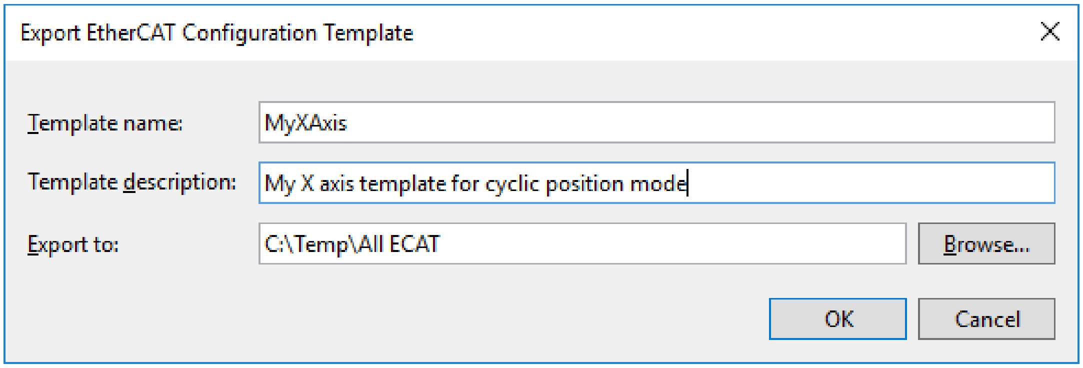
This context menu allows the User to set the EtherCAT slave slave/slaves network and export as a template to be used in the future. If the User has a lot of slaves with the same configuration (e.g. PDOs, InitCmds) then the User can use this feature to speed up development.

Note

It is possible to have slave network of commonly used EtherCAT devices and export it as one template for future use.

Steps to export EtherCAT configuration template

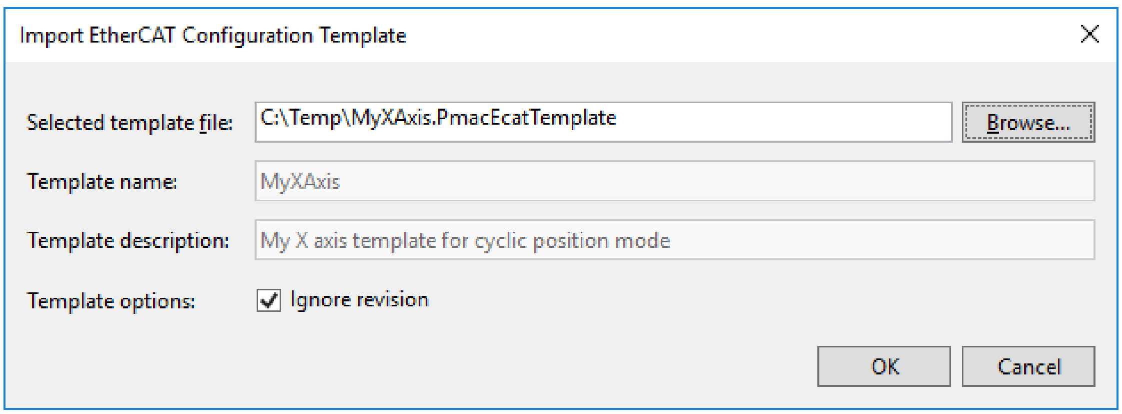
1. Configure EtherCAT Slave/Slaves network by either using Append slave or scan slave
2. Load PDO mappings
3. Make sure the EtherCAT network can be activated.
4. On success deactivate the network, right click on the Master node and select Export EtherCAT Configuration Template menu. The following dialog will open…



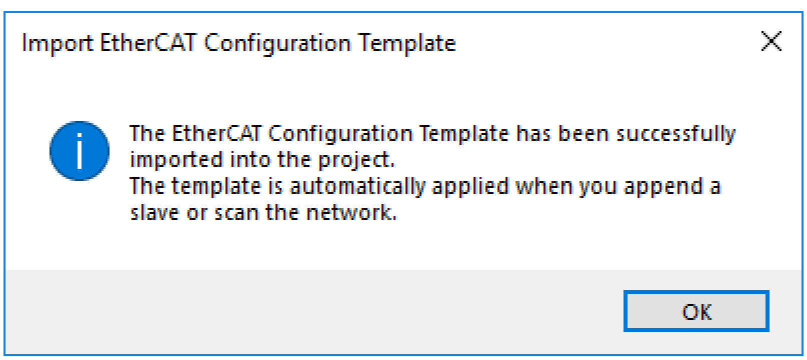
Enter all the necessary field's

Import EtherCAT Configuration Template

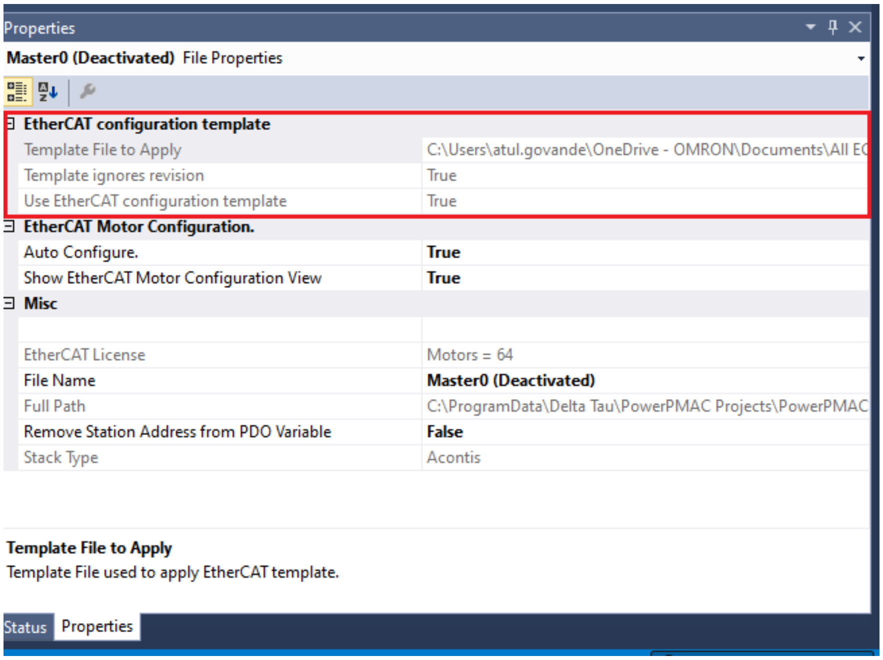
This context menu allows the User to apply the exported template. Right click on Master node and select the Import EtherCAT Configuration Template menu. The following dialog will open…



Enter all the necessary field's and click OK to import the template. On success the User will see the following message.

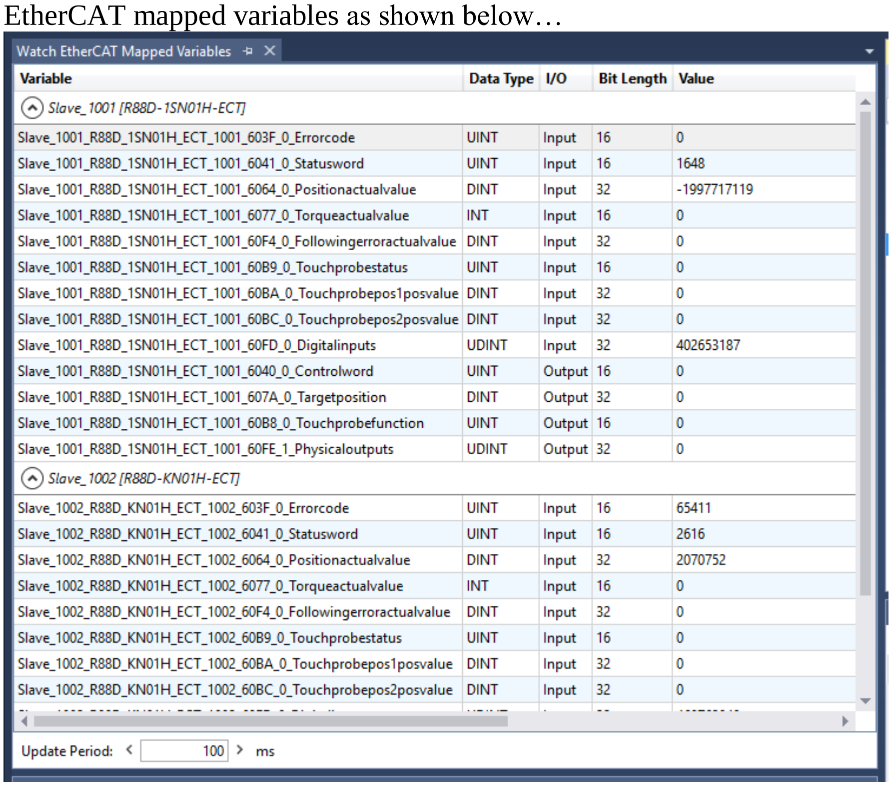


Once imported the project system will automatically apply and the new slaves will be copied from this template (if available) and from the ESI cache. This behavior is also used for the bus scan.
Select Master EtherCAT node and check the Properties. If the template is imported successfully then the Master node properties (shown below) will show name of the Template file, whether revision will be ignored or not and the Use EtherCAT template for matching slave. The property



Watch EtherCAT mapped variable

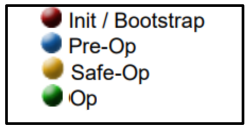
This context menu option allows the user to monitor/set (write-only) EtherCAT mapped variables. On clicking it the Watch EtherCAT Mapped Variables dialog will be opened, displaying current downloaded EtherCAT mapped variables as shown below…



The variable list is slave based and the user can collpase and expand the slaves to monitor the variables. Read-only variables cannot be altered and are grayed out. Write-only variables can be altered by the user and the new value will be downloaded to the Power PMAC.

Activate/Deactivate EtherCAT

This is used to Activating the EtherCAT network. Ecat[m].Enable = 1 command send to Power PMAC, where m is master index. The command is successful if the eni file and actual physical network matches. If there is mismatch an error will be thrown. At this point user can open Network mismatch analyzer to identify missing device.
On Activate EtherCAT the visualization in the project tree for the ECAT devices will be change according to the state of the ECAT device. Possible ECAT state visualizations are…



On successful Activation of the ECAT network this context menu command will change to Deactivate EtherCAT. This command is used for deactivating EtherCAT network, Ecat[m].Enable = 0 command is send to Power PMAC.

EtherCAT - Slave-Node Context Menu

Right click on any slave and it will open the context menu with commands associated with that slave. IDE V4.3.2.x and above will support Hot Connect Group.
The context menu looks like this…

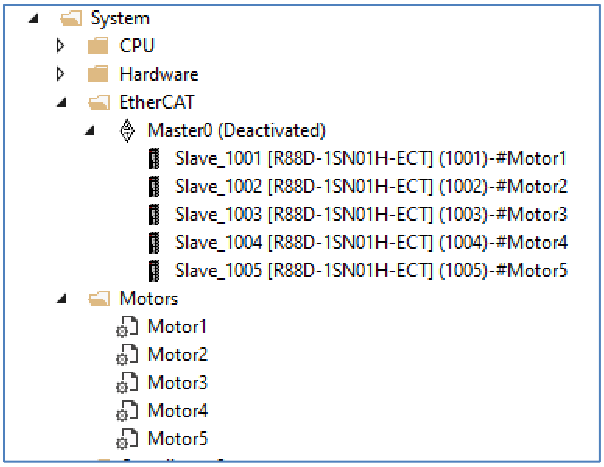
The following sections will explain the menu features in more detail.

Disable/Enable Slave

This feature allows the user to disable/enable the slave in the EtherCAT network. If, in the motion or PLC files, the PDO names are used then user will not be required to change the program even if the slave is disabled. The user must use the #define keyword in the programs instead of the actual EtherCAT structure element as these are managed automatically.

The following steps are needed for successful disabling of a slave.

1. Select a configured EtherCAT network that can be activated like below.

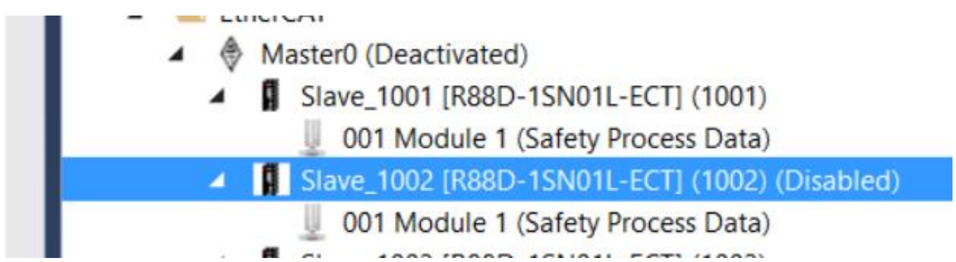


2. Load the mappings to the Power PMAC
3. To disable a slave, right click on a slave and select Disable Slave menu option. Once a slave is disabled, the menu option will change to Enable Slave, which the user can click to enable the slave.

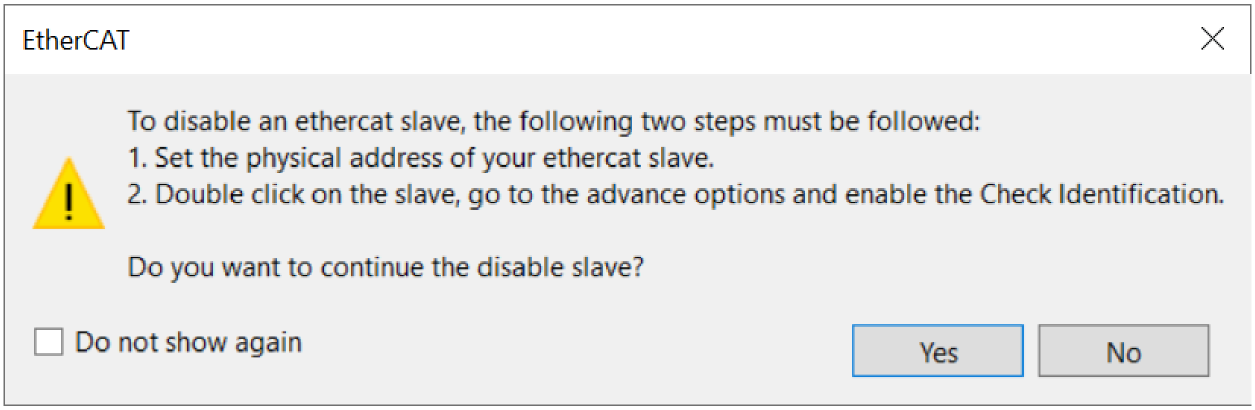
The first slave on the node cannot be disabled

Note

If a slave is disabled, the caption for the node will be appended with (Disabled) text.

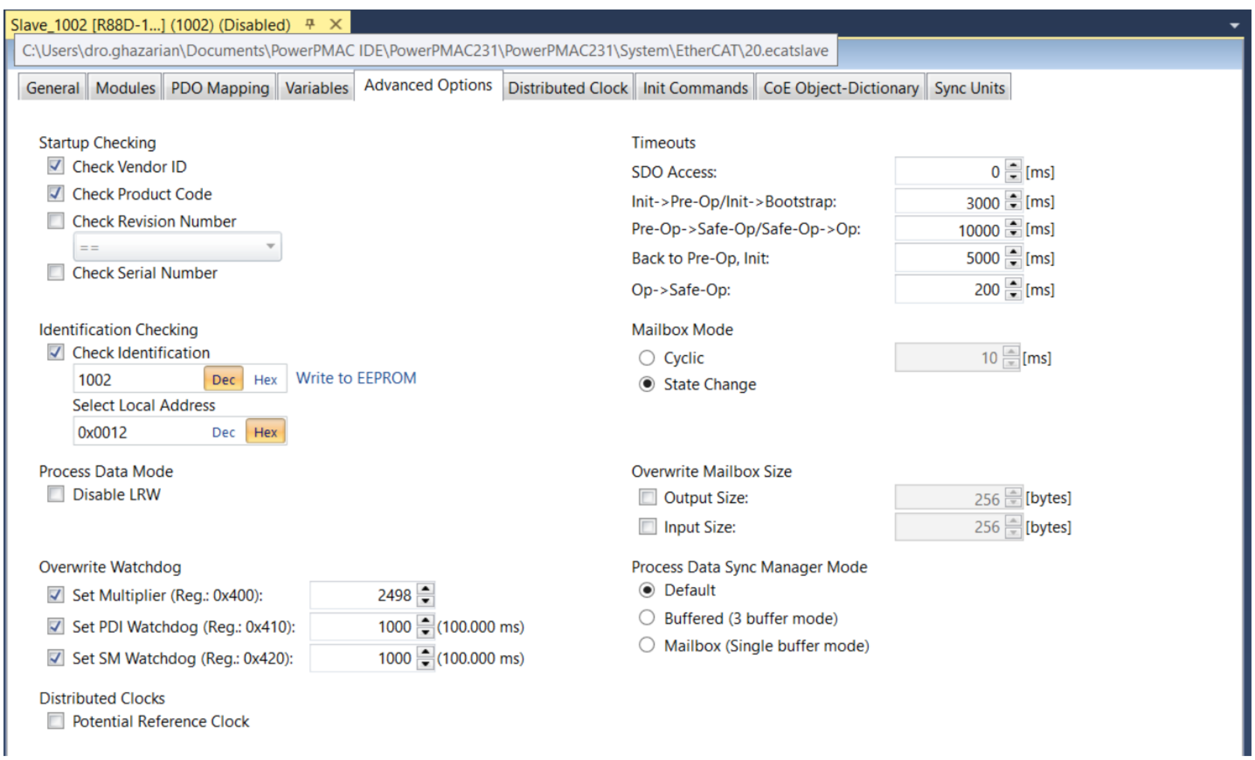


Once the Disable slave menu option is selected a message will popup which will guide the user to take the following steps to successfully disable a slave.



The following are the steps:

a. Set the Physical address of the Ethercat slave: for the disable feature to work correctly, each slave on the ring must have a unique slave address. This address is set by setting the rotary switches on the Ethercat slave to a unique address.
b. Double click on the slave and enable Check Identification: the user must also enable the check identification if the slave as well. To enable, double click on the slave to open up the property page. On this page select the Advanced Options tab and check the Check Identification.



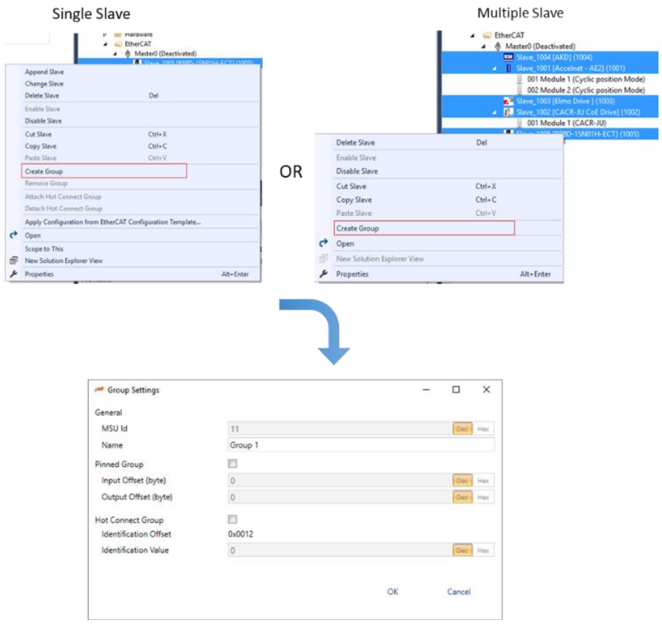
After setting these parameters a slave can be disabled/enabled. The disable enable command sets the Ecat[0].Slave[x].Enabled structure element to 0 for disable and 1 to enable.

Ethercat slave disable/enable feature requires a minimum IDE version of 4.6.0.x and firmware version 2.7.0.0

Note

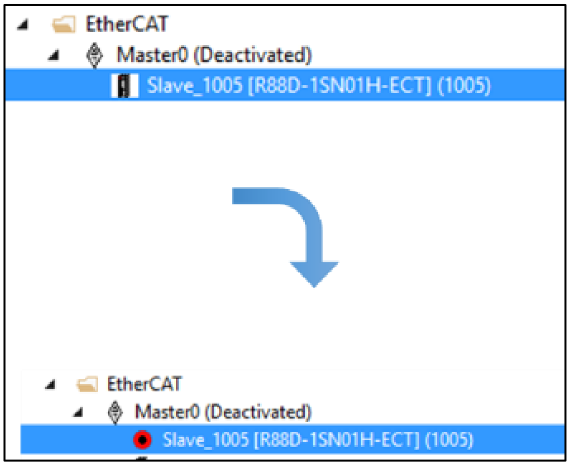
Hot (Connect) Create Group

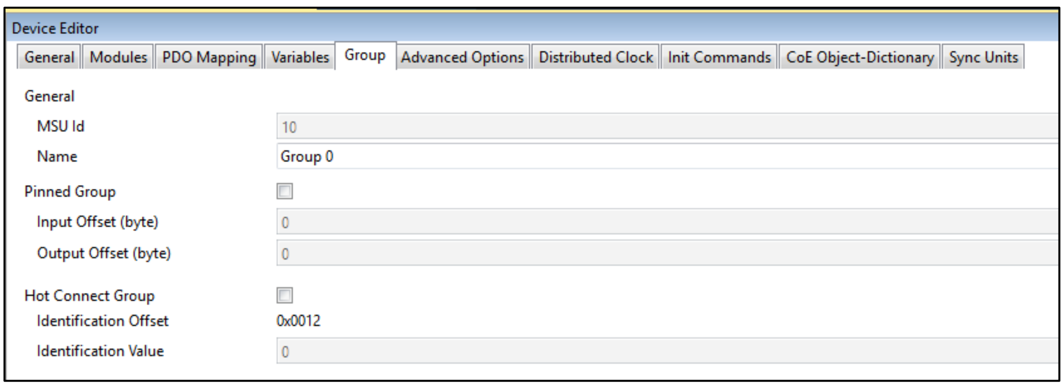
To create the hot connect group the user needs to select the Create Group context menu option by the right-clicking the slave. The user can select multiple slaves using CTRL+Mouse to add them to the group. On selecting a group, a dialog will be displayed as below…



- **General**
  - MSU Id: Generated Master Sync Unit Id
  - Name: Name of the group
- **Pinned Group**
  - Input Offset: Fixed input offset of the group in the process data image in bytes
  - Output Offset: Fixed output offset of the group in the process data image in bytes
- **Hot Connect Group**
  - Identification Offset: Register offset where the identification can be read from the slave
  - Identification Value: Hardware identification value or configured station alias address can be used.

As soon as the group is formed, the icon for the slave in the solution explorer will change and the group tab will be added to the slave editor, as shown below…

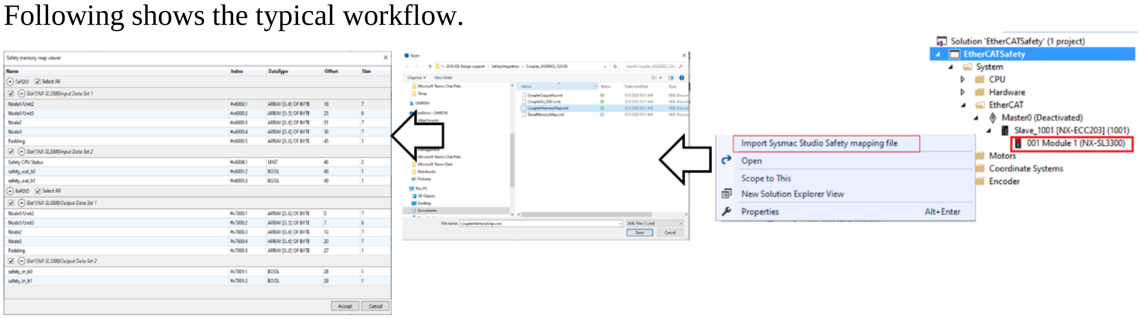




EtherCAT – Import SYSMAC Studio safety mapping file

This is a special slave context menu available only for OMRON Safety Module NX-SL3300, NX-SL3500, NX-SL5500, NX-SL5700.

This menu improves the setup time and ease of integration of safety controller with Power PMAC. Following shows the typical workflow.



On accept the mapping will be imported and added and available under PDO mapping.

Example - Safety Controller integration with Power PMAC IDE

Scope

Commissioning Safety PLC (NX-SL3300 or NX-SL3500) with 1S servo drive under the control of PMAC. Steps involving SYSMAC studio are out of the scope and this document assumes user has completed necessary steps involving SYSMAC studio.

Power PMAC IDE4.5.x or above

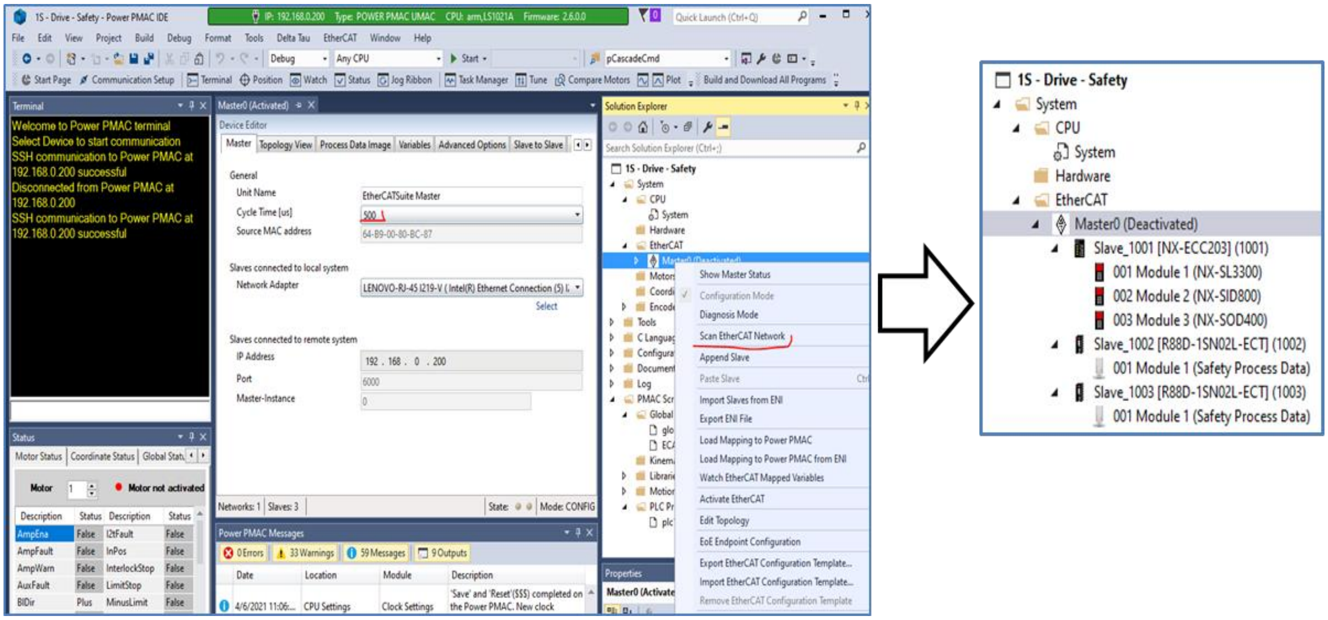
ITEM	NUMBER	DESCRIPTION	NOTES
1	CK3E-1310 / FW 2.6.0.0	Power PMAC	
2	NX-ECC203 / FW1.6	ECAT Coupler Unit	Use at least with FW1.6
3	SL3300	Safety PLC	
4	SID800	Safety Input Unit	
5	SOD400	Safety Output Unit	
6	R88D-1SN02L	1S Servo Drive / Motor	
7	R88D-1SN02L	1S Servo Drive / Motor	

Steps

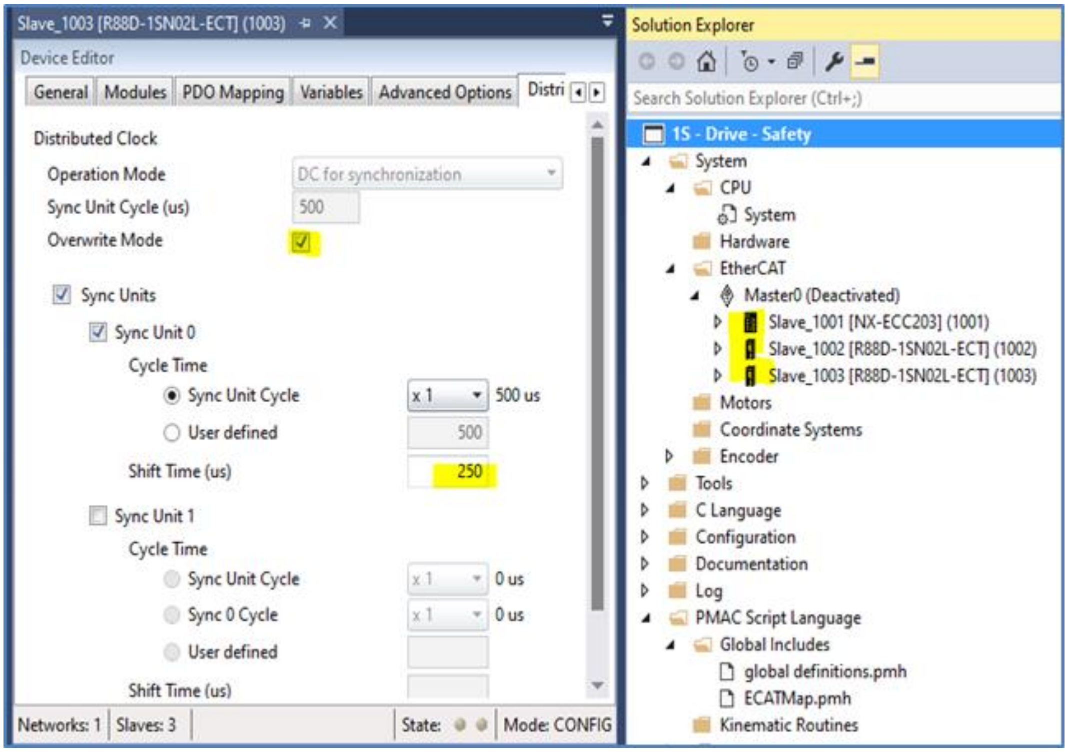
1. Sysmac Configuration
2. Download Sysmac project to ECC203 and SL3300
3. Export Sysmac PDO configuration
4. PMAC-IDE configuration

Power PMAC IDE Configuration

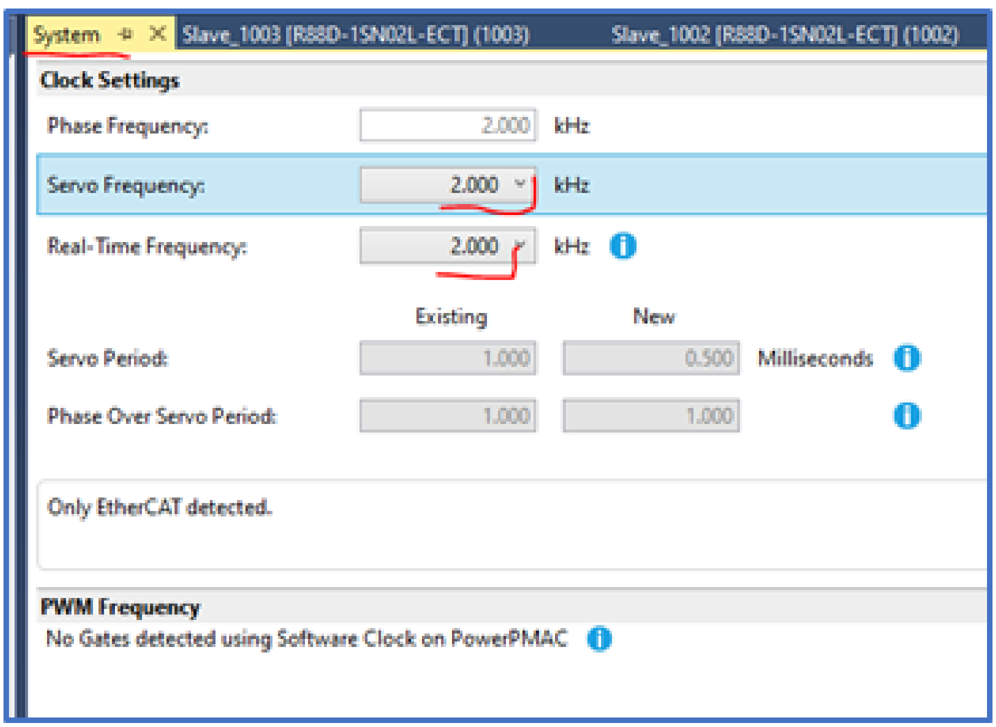
4.1 Reset & Re-Initialize Power PMAC. Scan EtherCAT Network. This will look like in the IDE



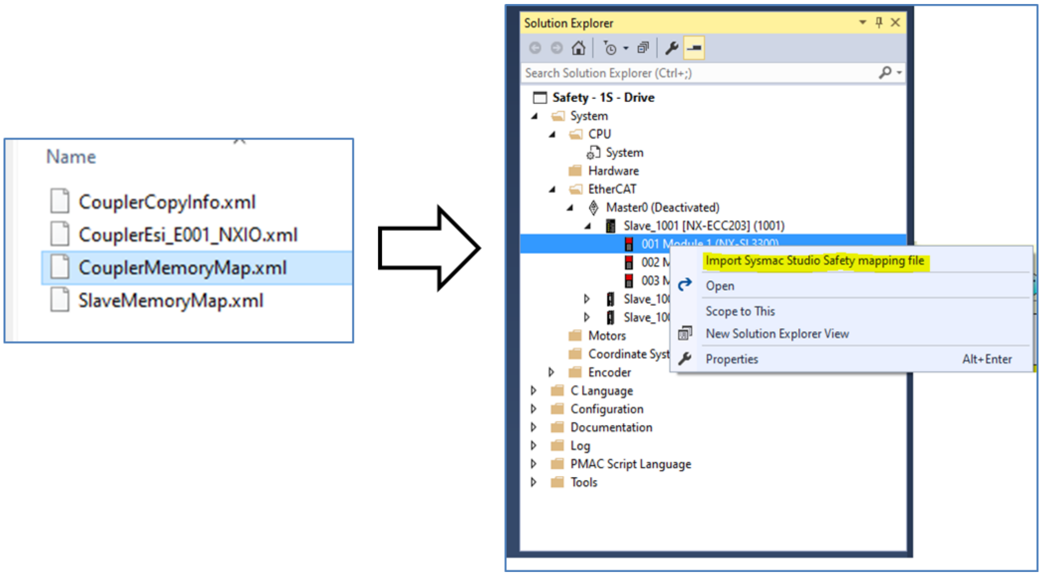
4.2 Set “Shift Time” to 250uS for all 3 ECAT devices (ECC203 and 2 drives)



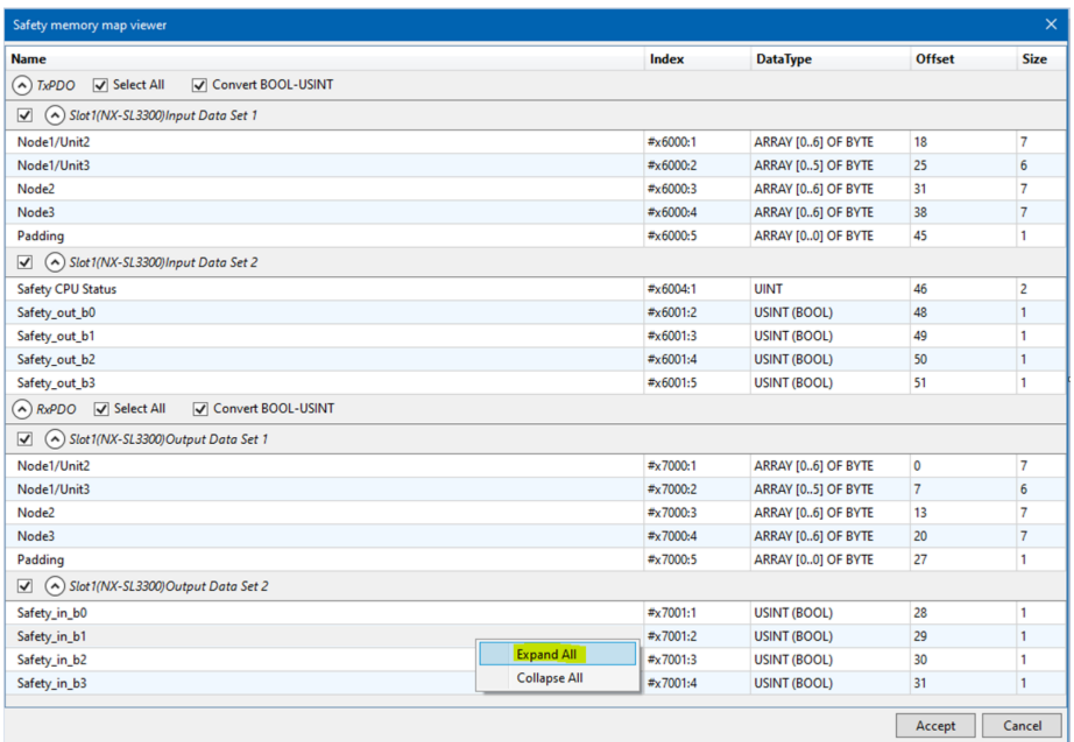
4.3 Set CPU speed @ 2 kHz. The example is tested up to 2kHz



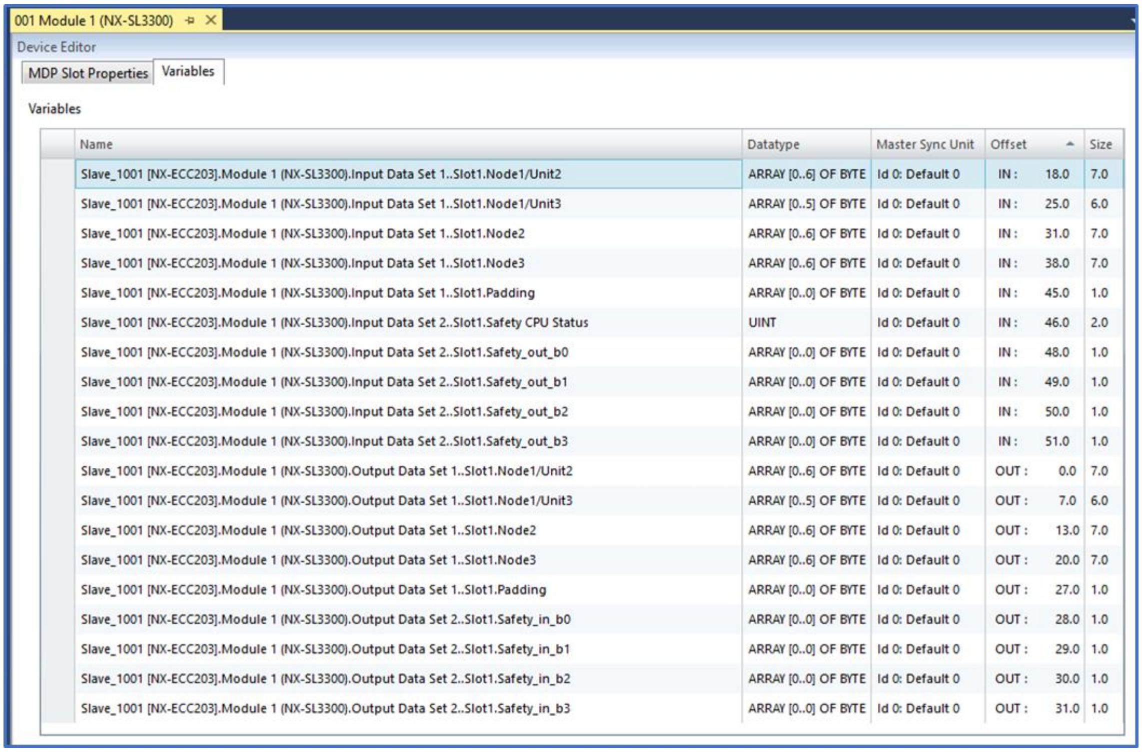
4.4 This is most important step , use the exported file from SYSMAC studio . The safety PDO map file name **CouplerMemoryMap.xml.** See below image of the import file using context menu on coupler. This option only available for OMRON safety controller and it's a dynamic context menu.



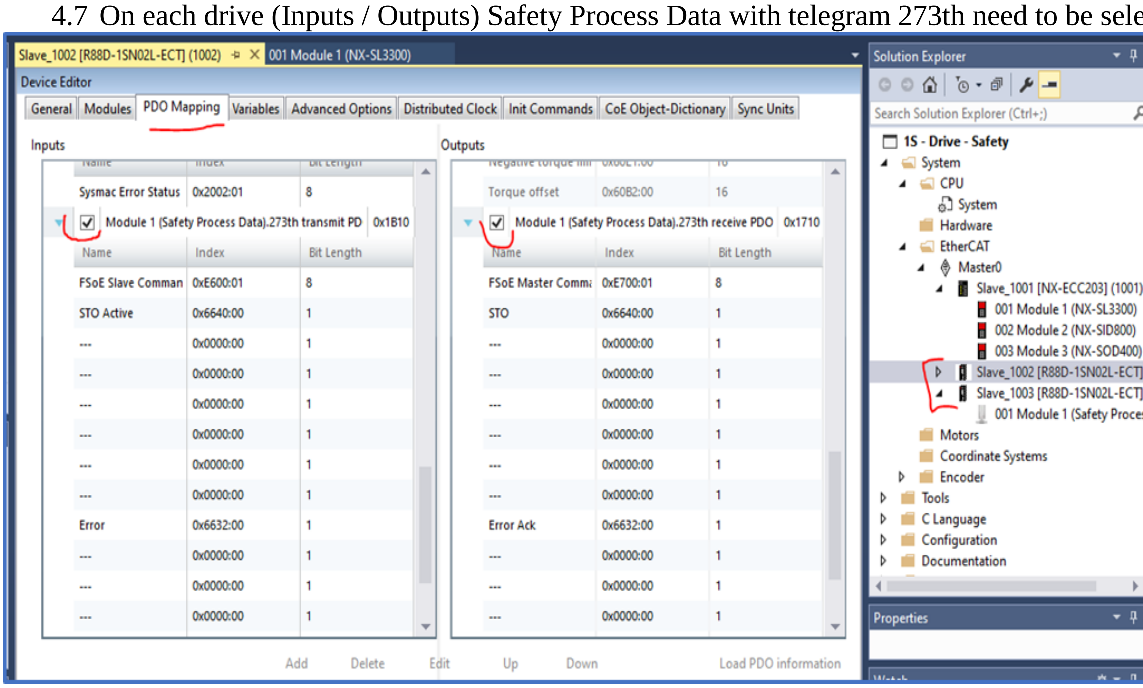
4.5 On successful import the viewer will be opened and shown below. Leave **Select All** selected and click Accept Leave checkbox **Convert BOOL-USINT** selected



4.6 After proper import, the Variables in Safety module should look like this

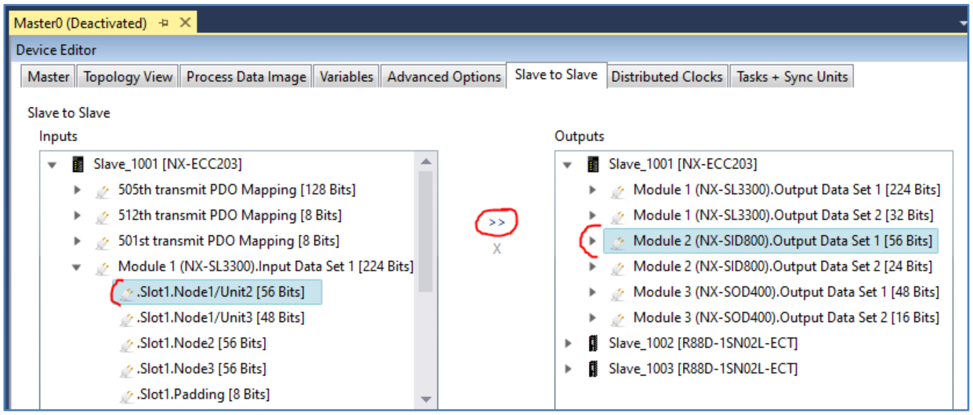


4.7 On each drive (Inputs / Outputs) Safety Process Data with telegram 273th need to be selected.

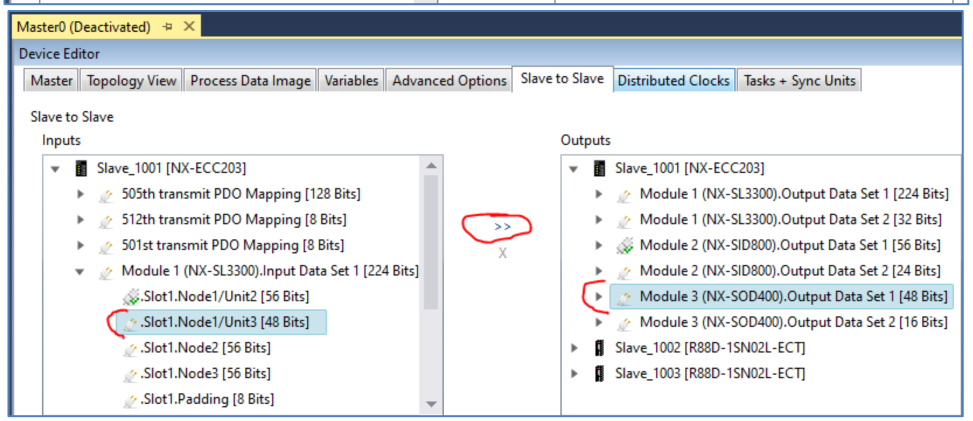


4.8 When PDO is complete, ***Slave to Slave*** communication need to be establish 4 connection for INPUTs - (this will vary with different configuration)

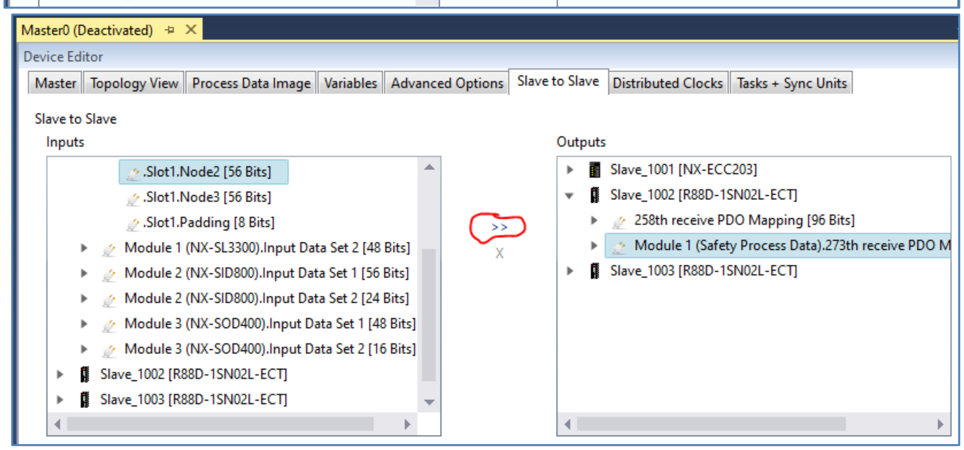
1


Master0 (Deactivated)
Device Editor
Master | Topology View | Process Data Image | Variables | Advanced Options | Slave to Slave | Distributed Clocks | Tasks + Sync Units
Slave to Slave
Inputs
Slave_1001 [NX-ECC203]
505th transmit PDO Mapping [128 Bits]
512th transmit PDO Mapping [8 Bits]
501st transmit PDO Mapping [8 Bits]
Module 1 (NX-SL3300).Input Data Set 1 [224 Bits]
.Slot1.Node1/Unit2 [56 Bits]
.Slot1.Node1/Unit3 [48 Bits]
.Slot1.Node2 [56 Bits]
.Slot1.Node3 [56 Bits]
.Slot1.Padding [8 Bits]
>>
X
Outputs
Slave_1001 [NX-ECC203]
Module 1 (NX-SL3300).Output Data Set 1 [224 Bits]
Module 1 (NX-SL3300).Output Data Set 2 [32 Bits]
Module 2 (NX-SID800).Output Data Set 1 [56 Bits]
Module 2 (NX-SID800).Output Data Set 2 [24 Bits]
Module 3 (NX-SOD400).Output Data Set 1 [48 Bits]
Module 3 (NX-SOD400).Output Data Set 2 [16 Bits]
Slave_1002 [R88D-1SN02L-ECT]
Slave_1003 [R88D-1SN02L-ECT]

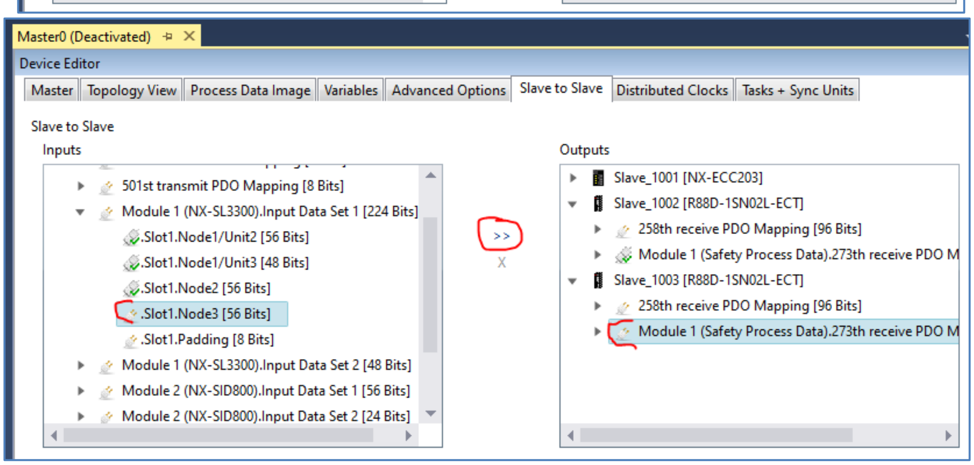
2


Master0 (Deactivated)
Device Editor
Master | Topology View | Process Data Image | Variables | Advanced Options | Slave to Slave | Distributed Clocks | Tasks + Sync Units
Slave to Slave
Inputs
Slave_1001 [NX-ECC203]
505th transmit PDO Mapping [128 Bits]
512th transmit PDO Mapping [8 Bits]
501st transmit PDO Mapping [8 Bits]
Module 1 (NX-SL3300).Input Data Set 1 [224 Bits]
.Slot1.Node1/Unit2 [56 Bits]
.Slot1.Node1/Unit3 [48 Bits]
.Slot1.Node2 [56 Bits]
.Slot1.Node3 [56 Bits]
.Slot1.Padding [8 Bits]
>>
X
Outputs
Slave_1001 [NX-ECC203]
Module 1 (NX-SL3300).Output Data Set 1 [224 Bits]
Module 1 (NX-SL3300).Output Data Set 2 [32 Bits]
Module 2 (NX-SID800).Output Data Set 1 [56 Bits]
Module 2 (NX-SID800).Output Data Set 2 [24 Bits]
Module 3 (NX-SOD400).Output Data Set 1 [48 Bits]
Module 3 (NX-SOD400).Output Data Set 2 [16 Bits]
Slave_1002 [R88D-1SN02L-ECT]
Slave_1003 [R88D-1SN02L-ECT]

3

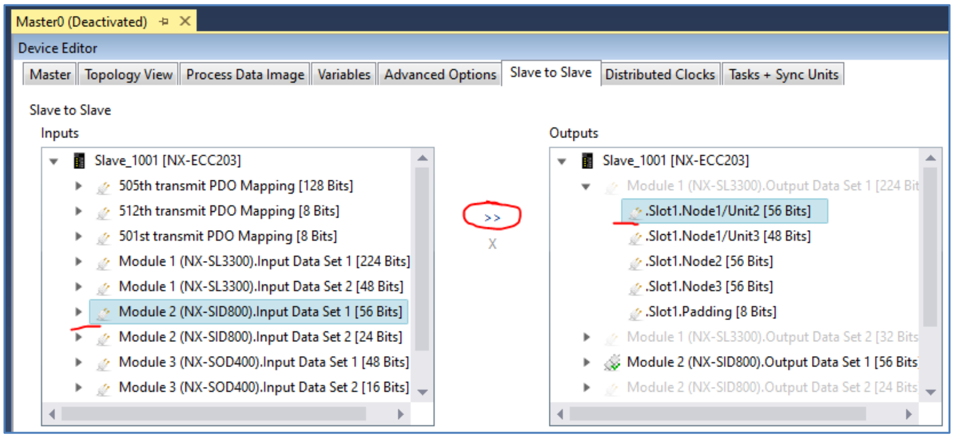

Master0 (Deactivated)
Device Editor
Master | Topology View | Process Data Image | Variables | Advanced Options | Slave to Slave | Distributed Clocks | Tasks + Sync Units
Slave to Slave
Inputs
.Slot1.Node2 [56 Bits]
.Slot1.Node3 [56 Bits]
.Slot1.Padding [8 Bits]
Module 1 (NX-SL3300).Input Data Set 2 [48 Bits]
Module 2 (NX-SID800).Input Data Set 1 [56 Bits]
Module 2 (NX-SID800).Input Data Set 2 [24 Bits]
Module 3 (NX-SOD400).Input Data Set 1 [48 Bits]
Module 3 (NX-SOD400).Input Data Set 2 [16 Bits]
Slave_1002 [R88D-1SN02L-ECT]
Slave_1003 [R88D-1SN02L-ECT]
>>
X
Outputs
Slave_1001 [NX-ECC203]
Slave_1002 [R88D-1SN02L-ECT]
258th receive PDO Mapping [96 Bits]
Module 1 (Safety Process Data).273th receive PDO M
Slave_1003 [R88D-1SN02L-ECT]

4

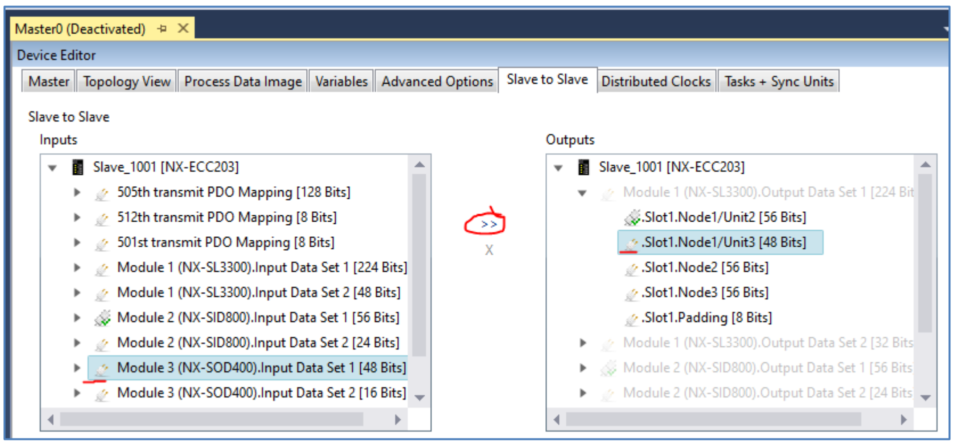

Master0 (Deactivated)
Device Editor
Master | Topology View | Process Data Image | Variables | Advanced Options | Slave to Slave | Distributed Clocks | Tasks + Sync Units
Slave to Slave
Inputs
501st transmit PDO Mapping [8 Bits]
Module 1 (NX-SL3300).Input Data Set 1 [224 Bits]
.Slot1.Node1/Unit2 [56 Bits]
.Slot1.Node1/Unit3 [48 Bits]
.Slot1.Node2 [56 Bits]
.Slot1.Node3 [56 Bits]
.Slot1.Padding [8 Bits]
Module 1 (NX-SL3300).Input Data Set 2 [48 Bits]
Module 2 (NX-SID800).Input Data Set 1 [56 Bits]
Module 2 (NX-SID800).Input Data Set 2 [24 Bits]
>>
X
Outputs
Slave_1001 [NX-ECC203]
Slave_1002 [R88D-1SN02L-ECT]
258th receive PDO Mapping [96 Bits]
Module 1 (Safety Process Data).273th receive PDO M
Slave_1003 [R88D-1SN02L-ECT]
258th receive PDO Mapping [96 Bits]
Module 1 (Safety Process Data).273th receive PDO M

4.9 4 connections for OUTPUTs - (this will vary if configuration is different).
Every time when modifying ECAT network Slave to Slave need to be Disconnected and Connected again.

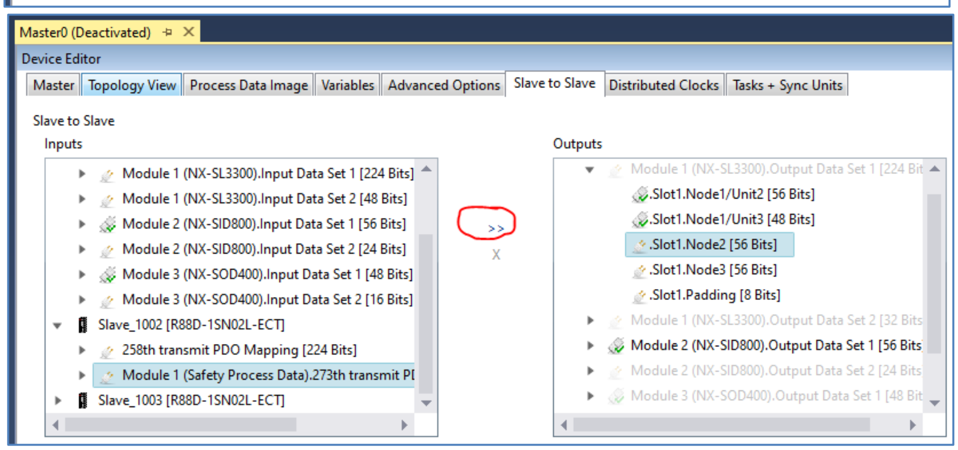
1



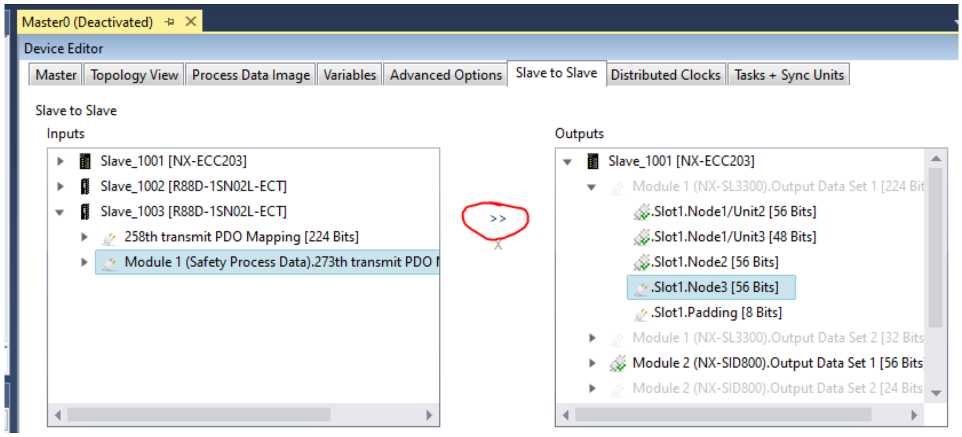
2



3



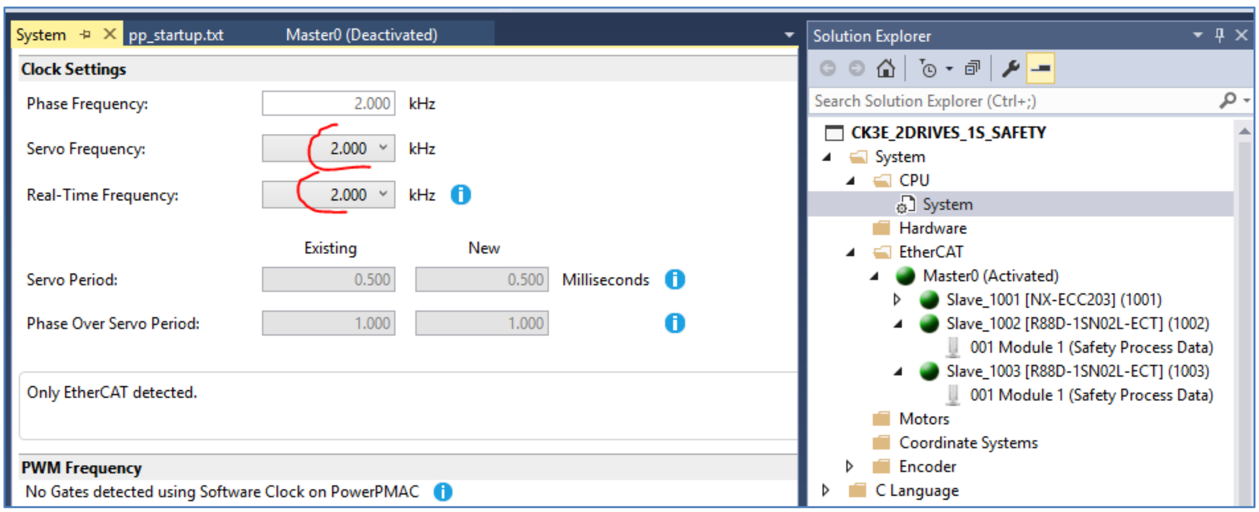
4



4.10 When completed, Connections menu should look like this

Input	Offset		Output	Offset	BitSize
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node1/Unit2	18.0	>>	Slave_1001 [NX-ECC203].Module 2 (NX-SID800).Output Data Set 1	32.0	56
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node1/Unit3	25.0	>>	Slave_1001 [NX-ECC203].Module 3 (NX-SOD400).Output Data Set 1	42.0	48
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node2	31.0	>>	Slave_1002 [R88D-1SN02L-ECT].Module 1 (Safety Process Data).273th receive PDO Mapping	82.0	56
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node3	38.0	>>	Slave_1003 [R88D-1SN02L-ECT].Module 1 (Safety Process Data).273th receive PDO Mapping	117.0	56
Slave_1001 [NX-ECC203].Module 2 (NX-SID800).Input Data Set 1	52.0	>>	Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node1/Unit2	0.0	56
Slave_1001 [NX-ECC203].Module 3 (NX-SOD400).Input Data Set 1	62.0	>>	Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node1/Unit3	7.0	48
Slave_1002 [R88D-1SN02L-ECT].Module 1 (Safety Process Data).273th transmit PDO Mapping	98.0	>>	Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node2	13.0	56
Slave_1003 [R88D-1SN02L-ECT].Module 1 (Safety Process Data).273th transmit PDO Mapping	133.0	>>	Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node3	20.0	56

4.11 Check the CPU clock to match the selected 2kHz for ECAT master



4.12 Load Mapping to PowerPMAC Enable the ECAT using right click context menu from Master node. Alternatively you can type in terminal window this command "ECAT[0].enable=1", though it is recommended to use clicking context menu. When RESET button is pressed, the CONTACTOR should enable and drives should remove STO ("St" on LED display) and go to normal operation ("—" on LED display).

EtherNet/IP

EtherNet/IP protocol is a member of the CIP network family of protocols, published by the ODVA. Power PMAC is an EtherNet/IP (EIP) adapter (slave) and will connect to an EtherNet/IP (EIP) scanner like NJ/NX controllers.

Prerequisite for Power PMAC EtherNet/IP adapter

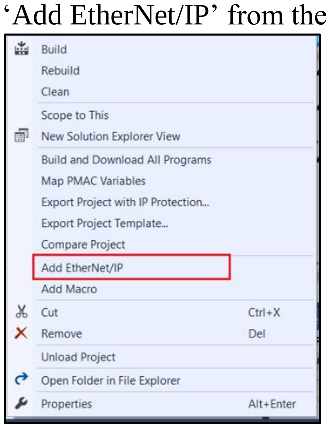
EtherNet/IP functionality support table…

FW Version	ARM Dual core CPU	CK3E/CK3M	ARM QUAD core CPU
2.5.4.x or above	EIP supported	EIP Not supported	EIP Not supported
2.6.x.x or above	EIP supported	EIP supported	EIP supported

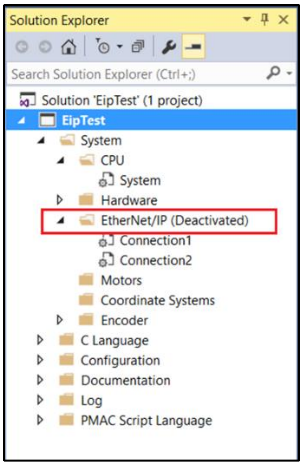
Note

EtherNet/IP must be enabled on the board. Upgrading the firmware to 2.5.4.x in the field on an existing board will not support the EtherNet/IP. In this case please contact to local support office.

The EtherNet/IP folder node in a Power PMAC project stores the EtherNet/IP connection information. An EtherNet/IP folder will be included when creating a new project using 'New Project' and select the project type 'Power PMAC project with EtherNet/IP'.
To add EtherNet/IP to an existing project, right click on the solution to open the context menu and select 'Add EtherNet/IP' from the menu as shown below…



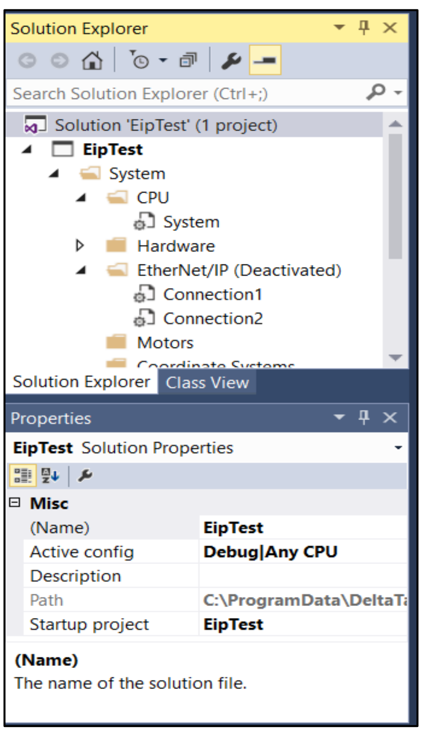
Once the EtherNet/IP node is added to the project, the user can add the connection to setup different variable data types to be shared with the scanner. The project looks like this with an EtherNet/IP node…



At the time of project loading the EtherNet/IP status is updated, provided an EtherNet/IP node is present. In the above image the current status of the network is “Deactivated”.
The User is encouraged to use context menu commands from the EtherNet/IP node to activate and deactivate EtherNet/IP functionality.

EtherNet/IP project node

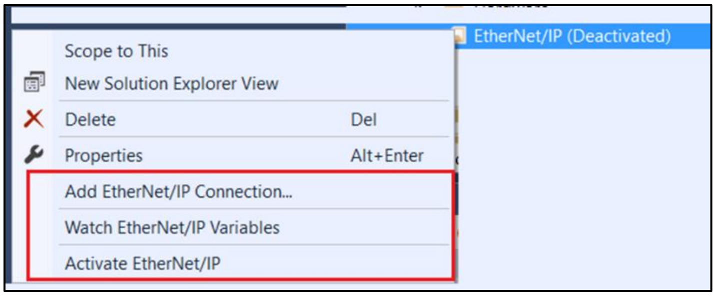
To set the EtherNet/IP update rate, select the EtherNet/IP project node. Update settings are available in the property window associated with the project node. It is displayed like this…



The range of the Update rate is 5 to 4294967295 uSec.

EtherNet/IP context menu

On right clicking the EtherNet/IP node following context menu is available.



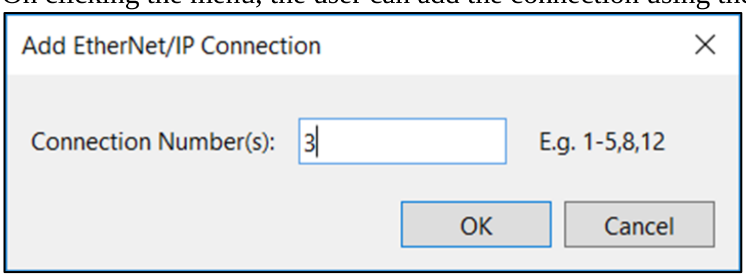
Add EtherNet/IP Connection:

As the name says, this menu allows the User to add a connection. A total of 32 connections can be added. Each input and output assembly per connection allows 504 bytes of data to be shared. The User can add only one data type per connection. In the current version of the IDE setup tool, the User cannot mix and match variable data types in one connection.

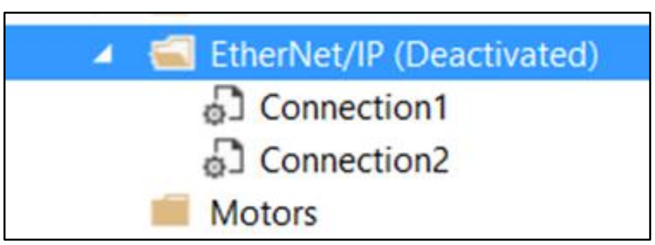
Note

User can add one data type variable per connection. There can be 32 connection possible and each can have one data type.

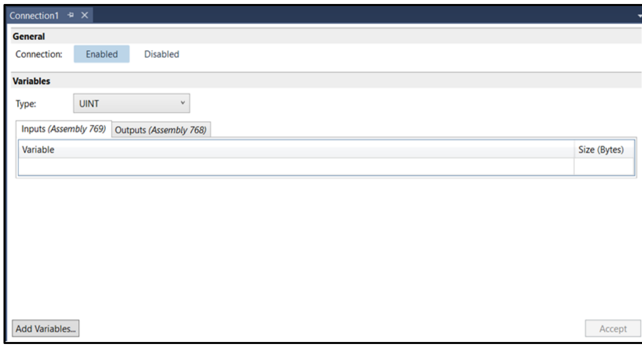
On clicking the menu, the user can add the connection using the following dialog…



On pressing OK, two connections will be added to the project tree. The project tree will look like this…



To setup EtherNet/IP, select the Connection1 or Connection2. When the EtherNet/IP configuration dialog opens it will look like this…

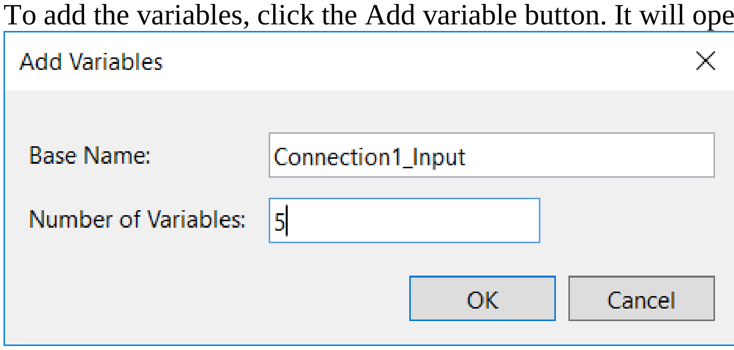


Connection: Sets the startup connection state. Default is Enabled
Type: Supported variable types one per connection.
Input/Output Assemblies: Direction from PowerPMAC point of view.
Add variable: Add number variable(s) to Input or Output assembly
Accept: Accept the connection variable settings.

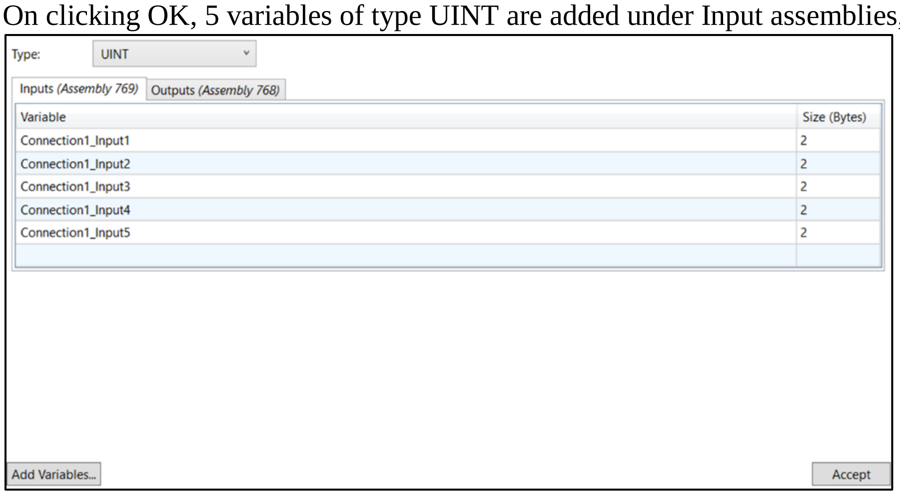
In the current setup tool, the following are the supported data types.

Data type supported	Data type size (byte)	Maximum number of variables
UINT	2	252
DINT	4	126
USINT	1	504
UDINT	4	126
REAL	4	126
LREAL	8	63
BYTE	1	504
WORD	2	252
DWORD	4	126

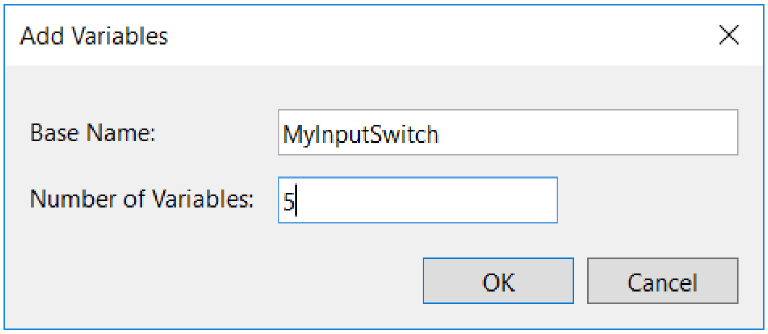
To add the variables, click the Add variable button. It will open the following dialog…



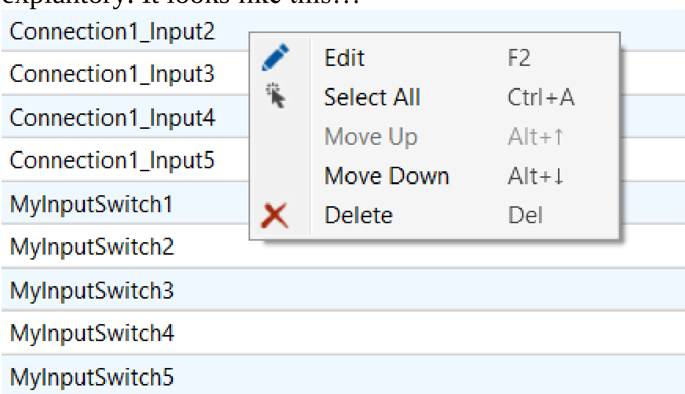
On clicking OK, 5 variables of type UINT are added under Input assemblies, as shown below…



The tool software will automatically generate the default names as shown. The User can change the variable name either by editing or at the time of creation as shown below…



Here the default Connection1_input base name is changed to MyInputSwitch. When customizing the names it is users reaponsibility to create unique names to avoid programming errors.
Each individual variable name is supported with a context menu. The menu is quite simple and self explantory. It looks like this…

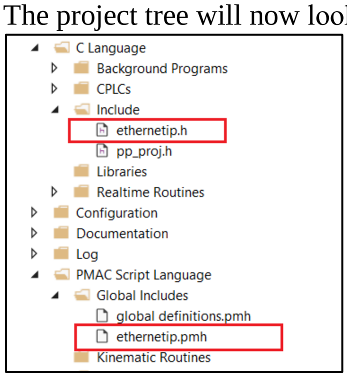


Similarly, the User can configure output assemblies and other necessary connections.
On completing the configuration of Input and Output assemblies, press Accept. The Accept button is per connction and will not be applied to all.

Note

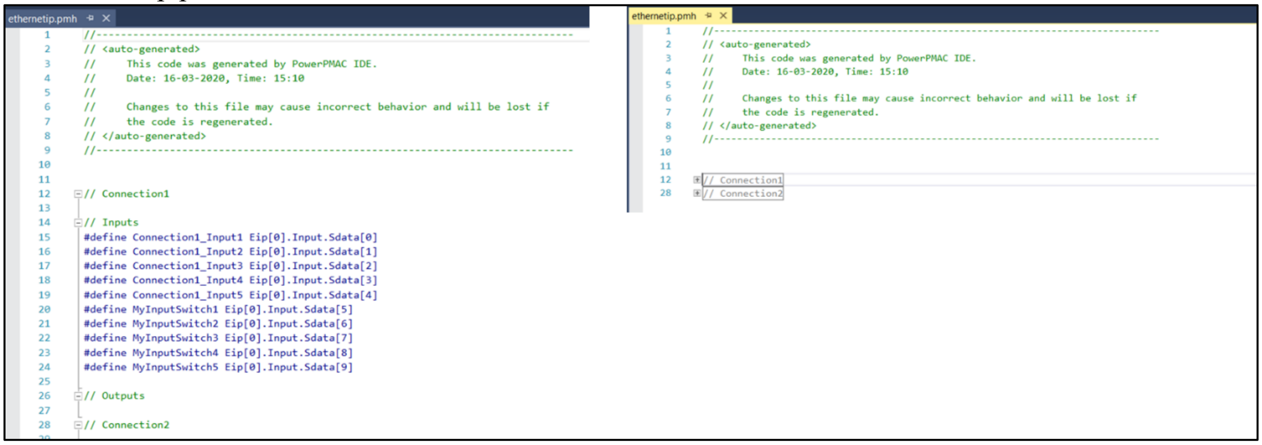
On completing the EIP configuration press Accept. Accept is per connection.

Once Accepted, it will create ethernetip.pmh and ethernetip.h to be used in programming the Power PMAC.
These newly created variables are available in the program editor and in the intellisense view.
The project tree will now look like this…



The variables are well organized per connection and can be collapsed or expanded per connection.

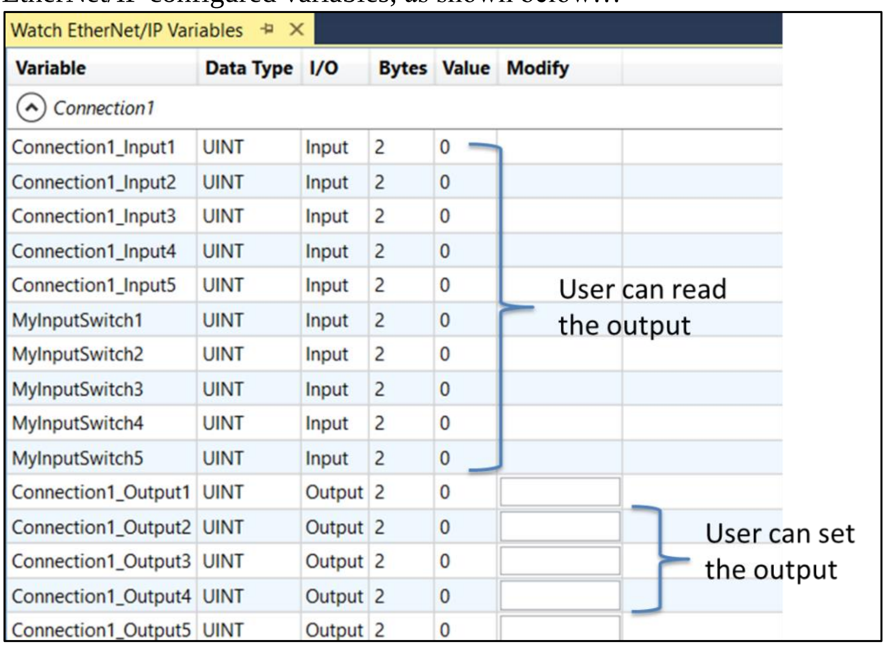
The ethernetip.pmh will look like this…


```
//---------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by PowerPMAC IDE.
//     Date: 16-03-2020, Time: 15:10
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//---------------------------------------------------------------------------


// Connection1

// Inputs
#define Connection1_Input1 Eip[0].Input.Sdata[0]
#define Connection1_Input2 Eip[0].Input.Sdata[1]
#define Connection1_Input3 Eip[0].Input.Sdata[2]
#define Connection1_Input4 Eip[0].Input.Sdata[3]
#define Connection1_Input5 Eip[0].Input.Sdata[4]
#define MyInputSwitch1 Eip[0].Input.Sdata[5]
#define MyInputSwitch2 Eip[0].Input.Sdata[6]
#define MyInputSwitch3 Eip[0].Input.Sdata[7]
#define MyInputSwitch4 Eip[0].Input.Sdata[8]
#define MyInputSwitch5 Eip[0].Input.Sdata[9]

// Outputs

// Connection2
```

```
//---------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by PowerPMAC IDE.
//     Date: 16-03-2020, Time: 15:10
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//---------------------------------------------------------------------------


// Connection1
// Connection2
```

Watch EtherNet/IP Variables

This context menu option allows the User to monitor/set (write-only) EtherNet/IP configured variables. On clicking it the Watch EtherNet/IP Variables dialog will be opened, displaying currently downloaded EtherNet/IP configured variables, as shown below…



Variable	Data Type	I/O	Bytes	Value	Modify
Connection1					
Connection1_Input1	UINT	Input	2	0	
Connection1_Input2	UINT	Input	2	0	
Connection1_Input3	UINT	Input	2	0	
Connection1_Input4	UINT	Input	2	0	
Connection1_Input5	UINT	Input	2	0	
MyInputSwitch1	UINT	Input	2	0	
MyInputSwitch2	UINT	Input	2	0	
MyInputSwitch3	UINT	Input	2	0	
MyInputSwitch4	UINT	Input	2	0	
MyInputSwitch5	UINT	Input	2	0	
Connection1_Output1	UINT	Output	2	0	
Connection1_Output2	UINT	Output	2	0	
Connection1_Output3	UINT	Output	2	0	
Connection1_Output4	UINT	Output	2	0	
Connection1_Output5	UINT	Output	2	0	

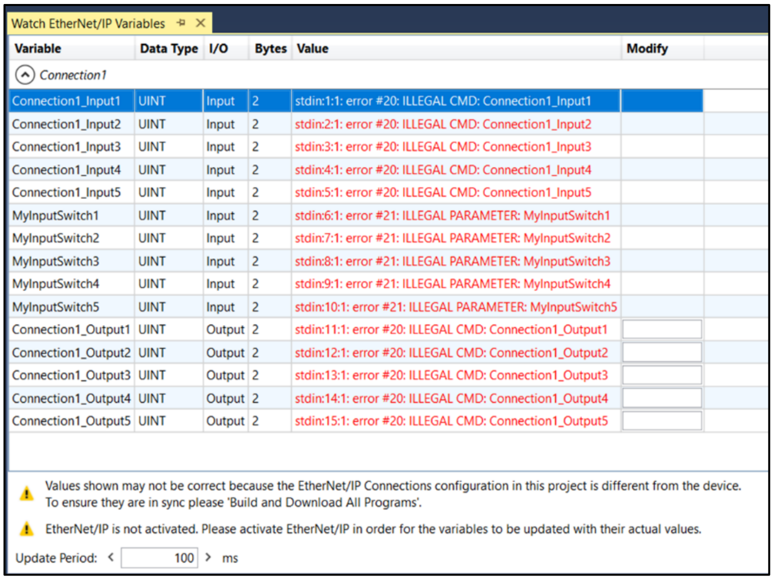
Watch EtherNET/IP Variables will automatically update the read and write variables on activating the EtherNet/IP.

The User can write to Output variables depending on their byte size. If the value is more than the byte size, it will be indicated with RED square.

Note

Watch EtherNet/IP variables window requires the project to be built and downloaded to Power PMAC.

If the Watch EtherNet/IP Variables menu is opened without a project built and downloaded, the following will be displayed…



Watch EtherNet/IP Variables

Variable	Data Type	I/O	Bytes	Value	Modify
Connection1					
Connection1_Input1	UINT	Input	2	stdin:1:1: error #20: ILLEGAL CMD: Connection1_Input1	
Connection1_Input2	UINT	Input	2	stdin:2:1: error #20: ILLEGAL CMD: Connection1_Input2	
Connection1_Input3	UINT	Input	2	stdin:3:1: error #20: ILLEGAL CMD: Connection1_Input3	
Connection1_Input4	UINT	Input	2	stdin:4:1: error #20: ILLEGAL CMD: Connection1_Input4	
Connection1_Input5	UINT	Input	2	stdin:5:1: error #20: ILLEGAL CMD: Connection1_Input5	
MyInputSwitch1	UINT	Input	2	stdin:6:1: error #21: ILLEGAL PARAMETER: MyInputSwitch1	
MyInputSwitch2	UINT	Input	2	stdin:7:1: error #21: ILLEGAL PARAMETER: MyInputSwitch2	
MyInputSwitch3	UINT	Input	2	stdin:8:1: error #21: ILLEGAL PARAMETER: MyInputSwitch3	
MyInputSwitch4	UINT	Input	2	stdin:9:1: error #21: ILLEGAL PARAMETER: MyInputSwitch4	
MyInputSwitch5	UINT	Input	2	stdin:10:1: error #21: ILLEGAL PARAMETER: MyInputSwitch5	
Connection1_Output1	UINT	Output	2	stdin:11:1: error #20: ILLEGAL CMD: Connection1_Output1	
Connection1_Output2	UINT	Output	2	stdin:12:1: error #20: ILLEGAL CMD: Connection1_Output2	
Connection1_Output3	UINT	Output	2	stdin:13:1: error #20: ILLEGAL CMD: Connection1_Output3	
Connection1_Output4	UINT	Output	2	stdin:14:1: error #20: ILLEGAL CMD: Connection1_Output4	
Connection1_Output5	UINT	Output	2	stdin:15:1: error #20: ILLEGAL CMD: Connection1_Output5	

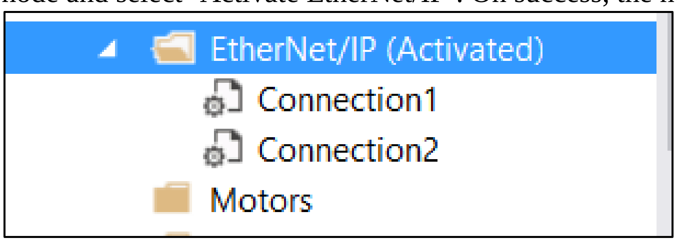
Values shown may not be correct because the EtherNet/IP Connections configuration in this project is different from the device. To ensure they are in sync please 'Build and Download All Programs'.

EtherNet/IP is not activated. Please activate EtherNet/IP in order for the variables to be updated with their actual values.

Update Period: 100 ms

Activate/Deactivate EtherNet/IP

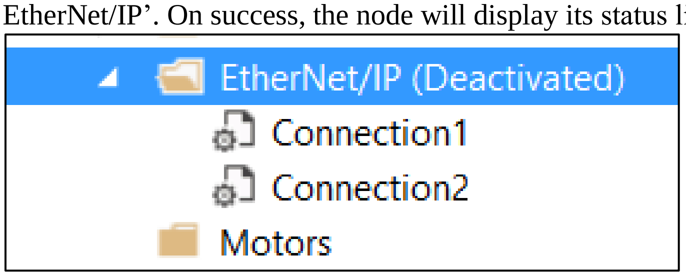
This menu is for activating and deactivating EtherNet/IP. When necessary EtherNet/IP setup is completed, the User can build and download a project. To test EtherNet/IP, right click on the EtherNet/IP node and select 'Activate EtherNet/IP'. On success, the node will display its status like this…



Activate EtherNet/IP sends Eip.Enabled = 1 command to Power PMAC. The User is required to enable the individual connection for testing using Eip[n].Enabled = 1 from the Terminal Window where n is the connection number(32 max)

Note

To deactivate, please right click on the EtherNet/IP node. Now the menu will display 'Deactivate EtherNet/IP'. On success, the node will display its status like this…



Deactivate EtherNet/IP sends Eip.Enabled = 0 command to Power PMAC. The User is required to disable the individual connection for testing using Eip[n].Enabled = 0 from the Terminal Window where n is the connection number(32 max)

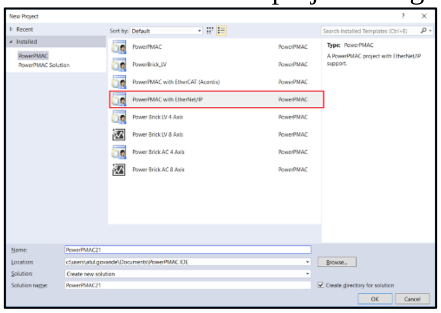
Note

EtherNet/IP Configuration Steps

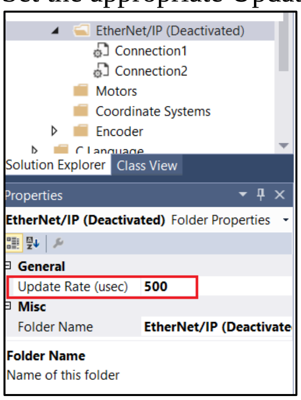
Here is a basic step to create an EtherNet/IP configuration.

Requirement: Power PMAC with factory installed 2.5.4.x FW that supports EIP as an Adpater (Slave) to a
NJ/NX as Scanner(Master) with Sysmac Studio

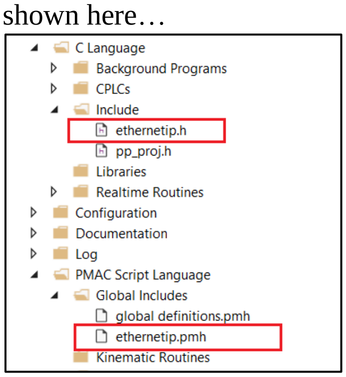
1. Create a Power PMAC project using Open New dialog like this…



Select the 'Power PMAC with EtherNet/IP' project type. If you open the normal Power PMAC project, then you will need to add an EtherNet/IP node.
Set the appropriate Update Rate in usec…



2. Add EtherNet/IP connection(s)
Right Click the EtherNet/IP node to add the connection(s)
3. Add variable(s) into input/output assembly in each connection and Accept variables. Remember: only one variable data type is allowed per connection.
On accept, make sure the ethernetip.pmh and ethernetip.h are created under the project node, as shown here…



4. Build and download the project

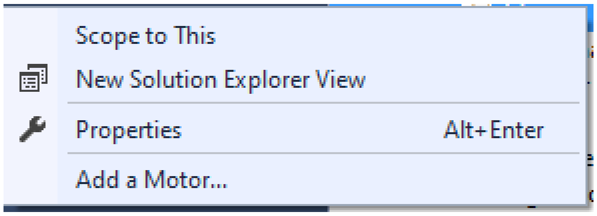
This is an important step in the EtherNet/IP configuration. Build and download creates the ethernetip.xml file. Download copies the file to the Power PMAC project configuration location. This file is important for EtherNet/IP data transfer. This file is not visible in the project as the file is maintain by the tool software. This file must not be altered by the User. The file looks like this…

```xml
<powerPMAC>
  <ethernetIP userMode="0" verbose="false" updateUSecs="500" enaOnStartup="true">
    <interface vendorId="47" name="eth0" macId="NA" ipAddress="NA" subnetMask="NA" gateway="NA" domainName="deltatau.com" hostName="PowerPMAC" />
    <connPath0 valid="1" enaOnStartup="true" verbose="0">
      <output0 assemNo="768" paramCount="0" size="0" />
      <explicit0 assemNo="770" paramCount="0" size="0" />
      <config0 assemNo="771" paramCount="0" size="0" />
      <input0 assemNo="769" paramCount="0" size="20" />
    </connPath0>
    <connPath1 valid="1" enaOnStartup="true" verbose="0">
      <output1 assemNo="772" paramCount="0" size="0" />
      <explicit1 assemNo="774" paramCount="0" size="0" />
      <config1 assemNo="775" paramCount="0" size="0" />
      <input1 assemNo="773" paramCount="0" size="0" />
    </connPath1>
```

5. It is recommend that after building and downloading, to save the project and issue the $$$ command. This will automatically enable EIP. Please refer the above ethernetip.xml file image, I particular the following attributes:
   enaOnStartup="true" This is for EIP level and command is Eip.enabled = 1
   enaOnStartup="true" This is for connection level and the command is Eip[0].Enabled = 1
   These are set to true, and this is the reason that after saving and issuing the $$$ command, the EtherNet/IP automatically gets activated.
   If the project is not saved, then the User will be required to activate the EtherNet/IP by right clicking the node and then individually enabling the connections from the Terminal Window. Please refer to the Activate EtherNet/IP section.
6. At this stage we expect the NJ/NX Scanner (master) is configured to communicate with an adapter using Sysmac Studio. This setup is out of scope for this manual. Please refer to Sysmac Studio documentation.

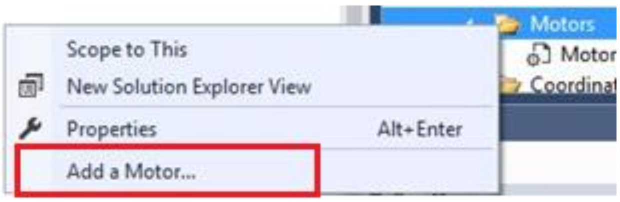
Motors – Context Menu

Right click on Motor node for available context menu.

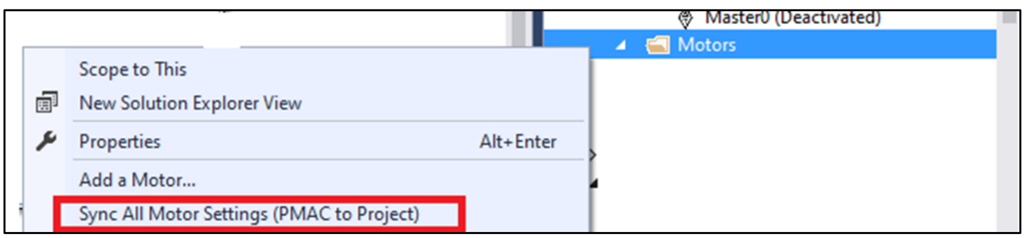


Add Motor

You can add a motor by right clicking on the motors node and select add motor



Motor Context menu is dynamic. When any motor is added to the project a new menu dynamically become visible, as shown below…

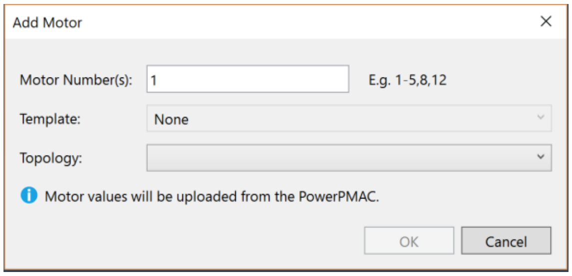


The Add motor dialog will open. The User can select a single or multiple motors to add, up to Sys.MaxMotors. If a motor already exists in the project this motor number will not be added but other selected Motors will.

The User will also be able to select a previously saved Template to use for the Motor configuration.

Note

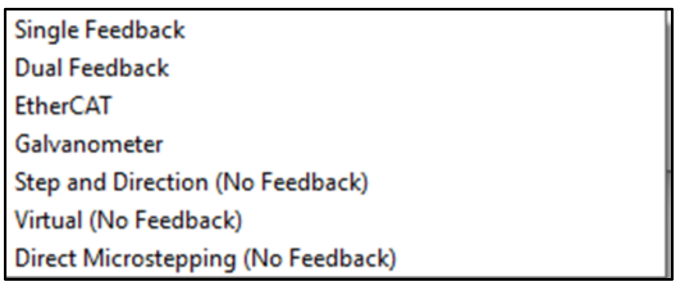
IDE V4.2 onwards Motors added to the project will be displayed in a natural order



In the IDE the motor configuration is in the form of a Topology view.
Currently there are six types of Motor configuration supported thorough Topology diagrams.

- Single feedback
- Dual Feedback
- EtherCAT
- Galvanometer
- Step & Direction (No Feedback)
- Virtual (No Feedback)
- Direct Microstepping (no Feedback)

The Topology dropdown is blank by default as the User needs to select the Topology type.

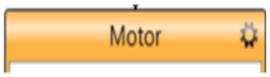


Note

When a Motor is added to a project the motor structure elements are saved to a file. Any motor structure element changes within the project domain will be automatically updated and maintained within the file. When the build is performed the motor file will be used to generate the systemsetup.cfg file. No backup is needed for the motor parameters as long as the changes are being made in the project system.

The following section will describe different Topology available for Add Motor menu.

Topology Color code

	This block is unavailable for setting either the previous block is not completed or this block is not needed for the current type of topology.
	This block has completed and settings are Accept.
	This block is ready for setup as the previous condition is met .
	When hoovering the mouse indicates this block can be selected to set

Common Motor Topology navigation guidelines
The Topology is a guide through the various different blocks. Once a block is accepted the next Block will be made available to edit.

Click Database icon  to open part manager where user can Add/Modify/delete Amplifier database.
Click Save  icon to save the Amplifier setting

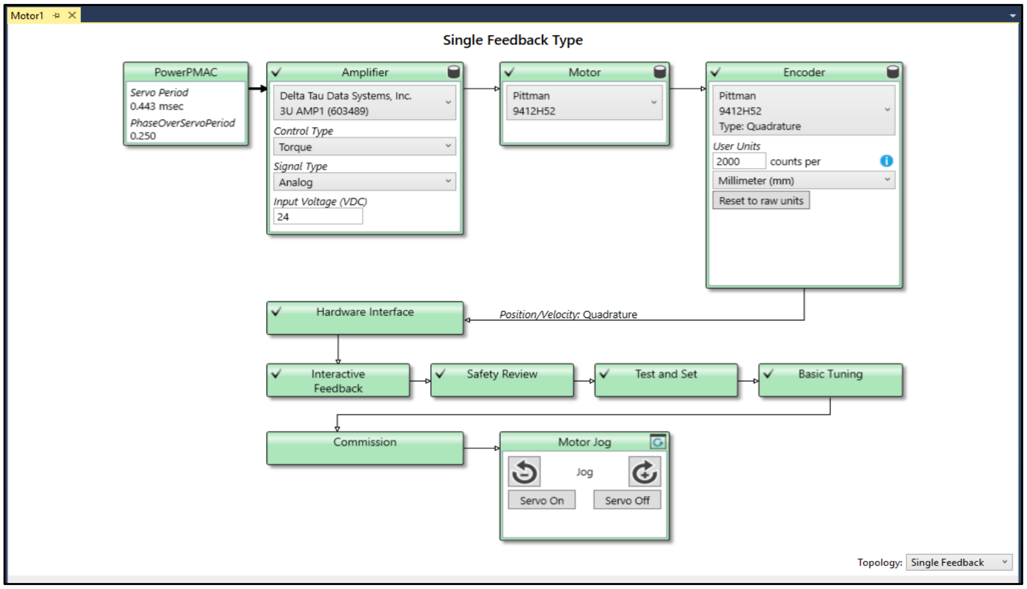
The tick indicates that a view has been opened and that the data has been Accepted.

Note

The difference in the single and dual feedback is that in dual feedback the user can set the second encoder and in hardware interface block can set the pEnc2 address differently .

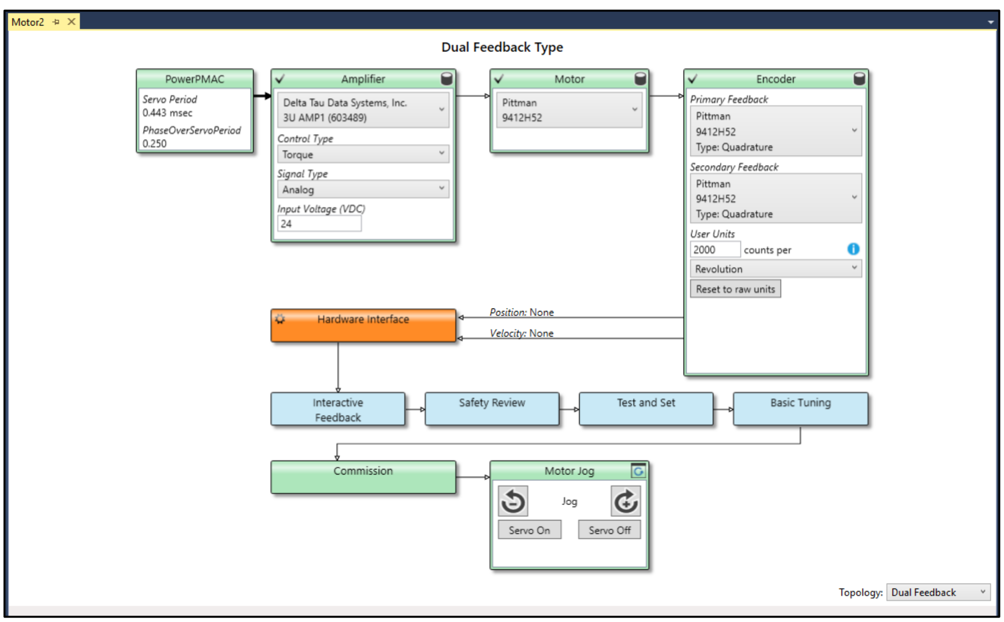
Topology- Single Feedback

The Single Feedback Topology is for a Single feedback solution, for position only.
If a Single Feedback Topology is selected, a Single Feedback Topology view will be displayed and the selected motor will be added under the Motor node in the Solution Explorer.



Topology- Dual Feedback

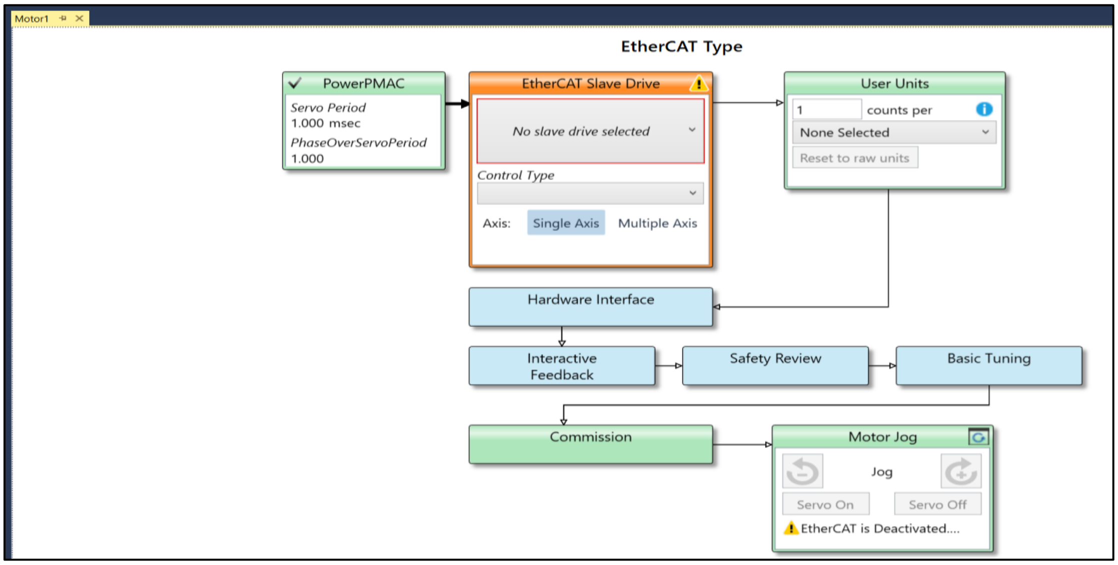
The Dual Feedback Topology is for a Dual feedback solution; one for position and one for velocity.
If a Dual Feedback Topology is selected a Dual Feedback Topology view will be displayed and the selected motor will be added under the Motor node in the Solution Explorer.



Topology- EtherCAT

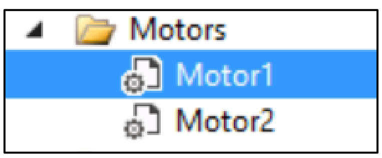
The EtherCAT Topology is for setting up an EtherCAT motor using an EtherCAT slave amplifier.

If an EtherCAT Topology is selected an EtherCAT Topology view will be displayed and the selected motor will be added under the Motor node in the Solution Explorer.

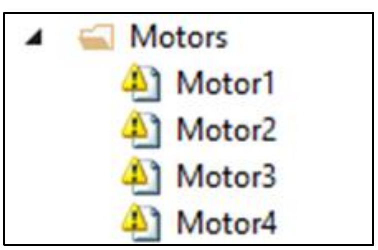


Topology dropdown allows the User to change the type of Feedback type. The User cannot go from Single Feedback or dual feedback to EtherCAT and vice-versa.

Once the motor is added it will show up under the Motors node as shown below:

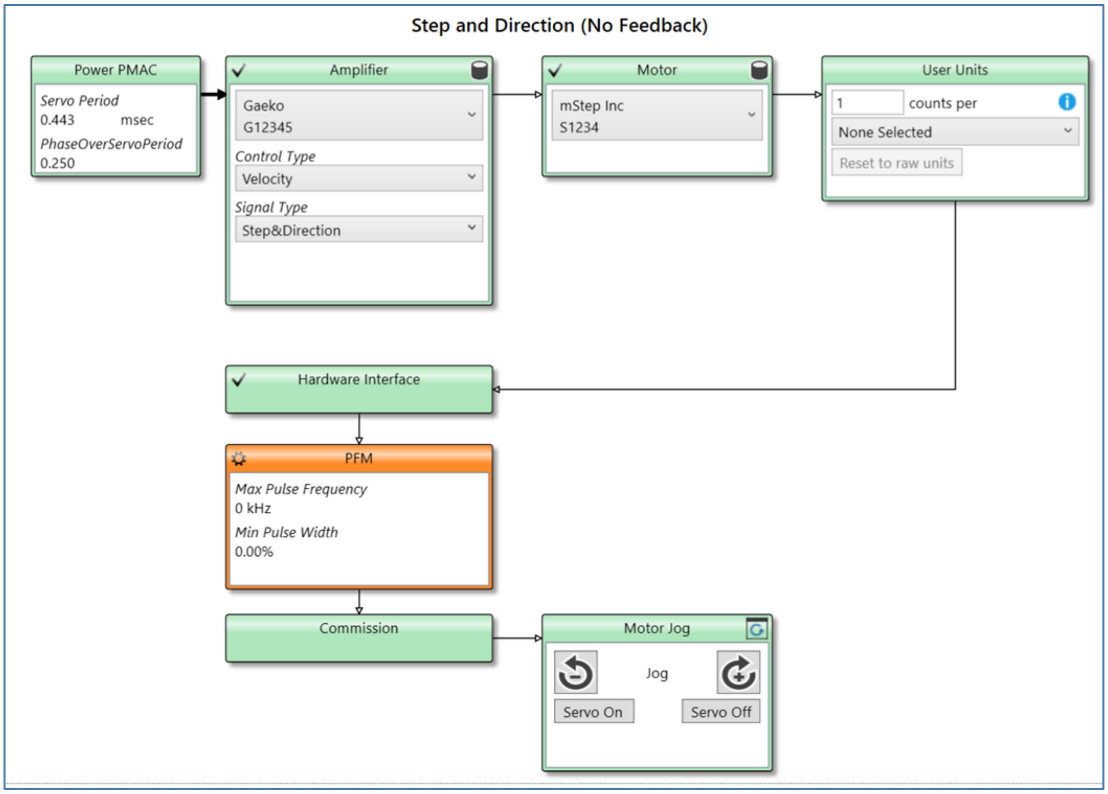


When the previously saved project is open that has motor and on opening if the Motor folder shows the motors like this….
The Yellow warning sign indicates that the project cannot find the associated motor file. In this case either user has to locate the file if it is accidently got deleted. Worst case user will require to add the motor again as the settings are lost.



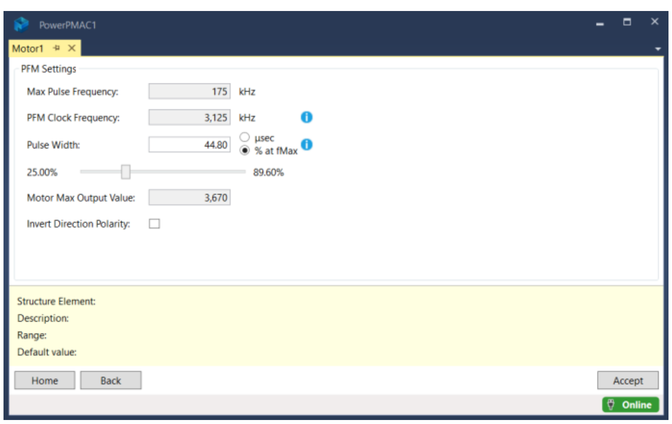
Topology- Step & direction (No Feedback)

The Step & direction (No Feedback) is setting the PFM mode and there is no feedback.
If a Step & direction (No Feedback) Topology is selected a No Feedback Topology view will be displayed and the selected motor will be added under the Motor node in the Solution Explorer

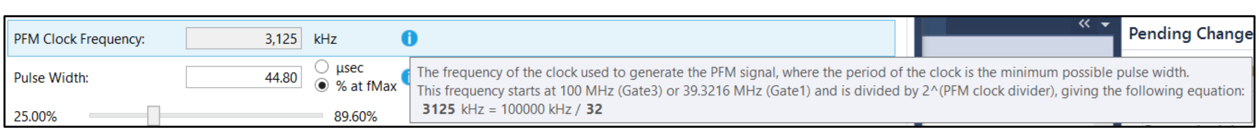


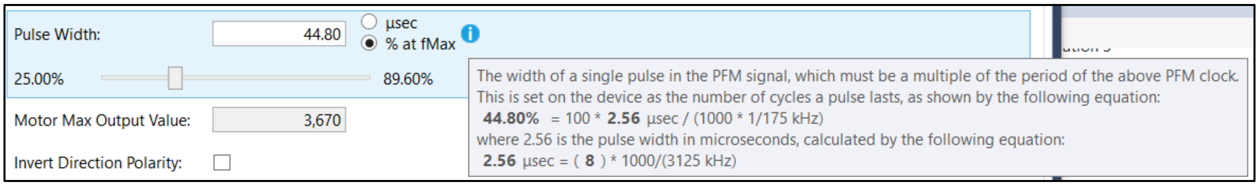
The PFM block allows the User to set up a Motor (Stepper) with no feedback. After following the topology workflow, when the PFM block is clicked, the User will see the dialog shown below.

This page will be prepopulated based on the Max frequency entry from the Amplifier page.



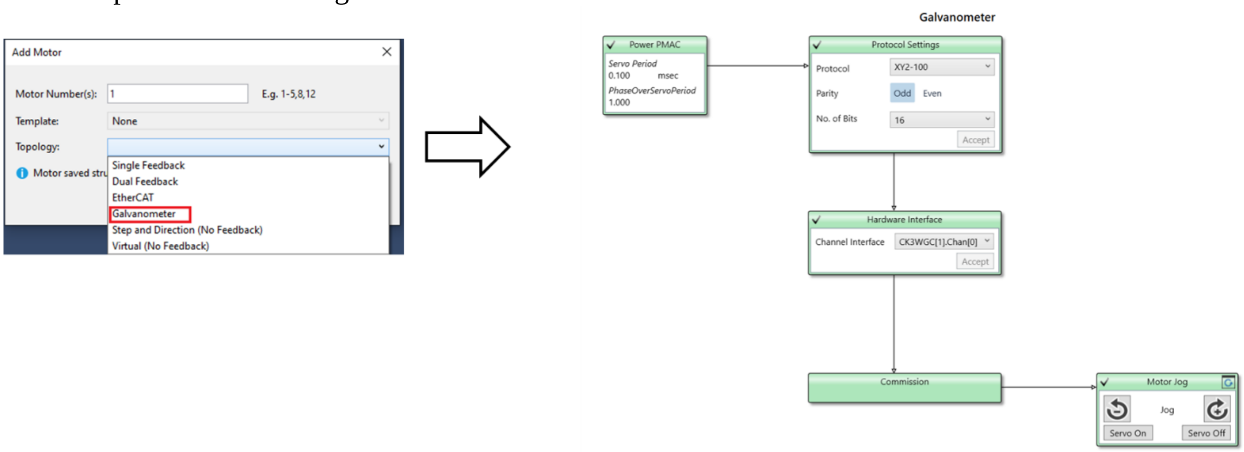
The following screen shots explain important properties and their settings.





Topology- Galvanometer

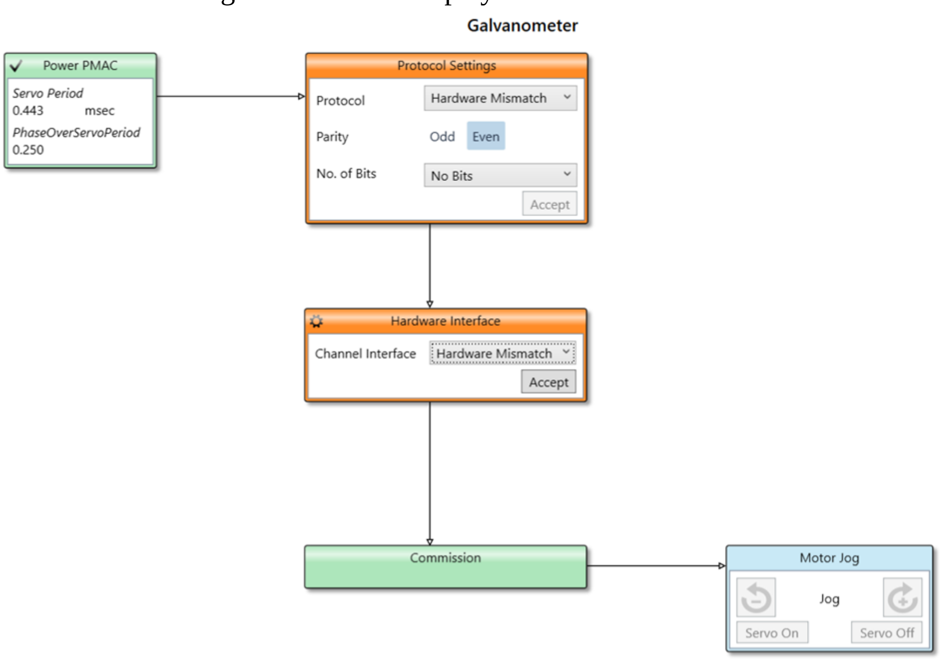
The Galvanometer Topology is for setting Galvo using CK3WGCxxx or Acc84 with either XY2-100 or SL2-100 protocol. On adding motor it will look like this..



User needs to Accept Protocol setting by selecting protocol from dropdown. Depending on card option protocol will be added to the list. User will also needs to enter number of bits. This all information is available with the amplifier that is used to control the galvo. Last select the type of parity. Default is Even parity.

Hardware interface page will display available channels based on the hardware detected. Accept the connected channel and Galvo is ready to go!

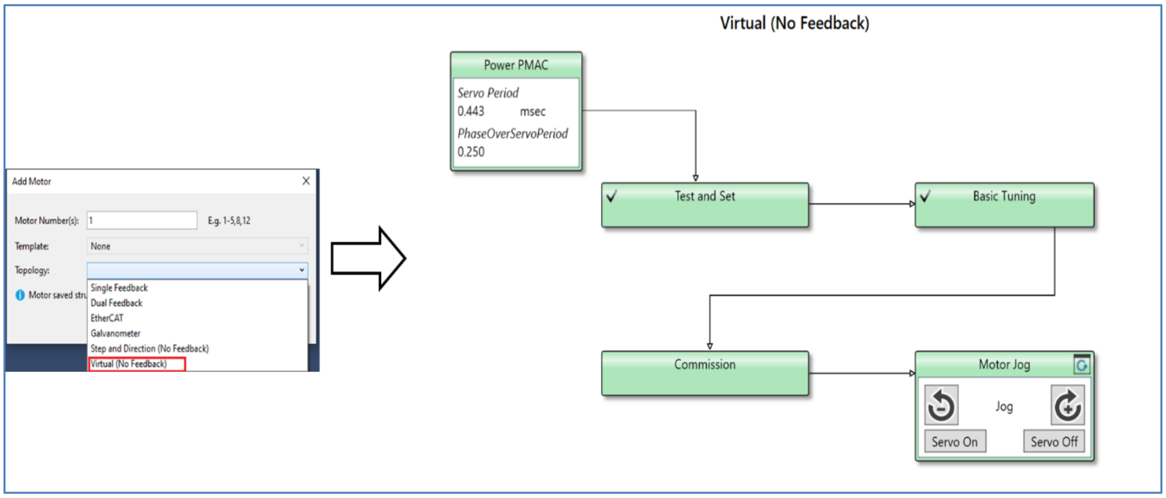
Power PMAC message window will display all the values that are downloaded to Power PMAC.



If Hardware mismatch is displayed (Above image) under Protocol settings and Hardware interface this means the detected hardware does not support Galvanometer Topology.

Topology- Virtual (No Feedback)

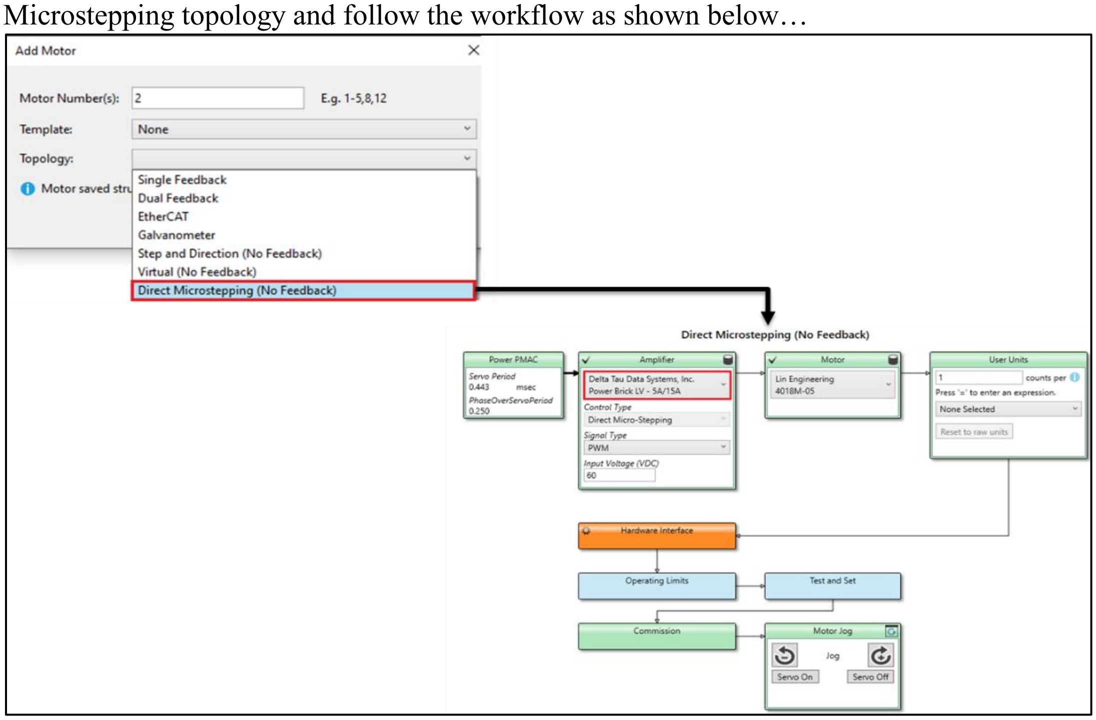
The virtual Motor topology is setting virtual motor. From Add Motor menu select Vitual (No Feedback) topology. It will show like this..



Adding virtual motor is simple as shown above once motor is added you are ready to Jog the motor in any direction. All the Topology block are Green meaning completed and settings are Accepted and downloaded to Power PMAC. Power PMAC Messages window show you what is downloaded to Power PMAC.

Topology-Direct Microstepping (No Feedback)

The Direct Microstepping topology is mainly used with Power Brick LV. From Add motor select Direct Microstepping topology and follow the workflow as shown below…

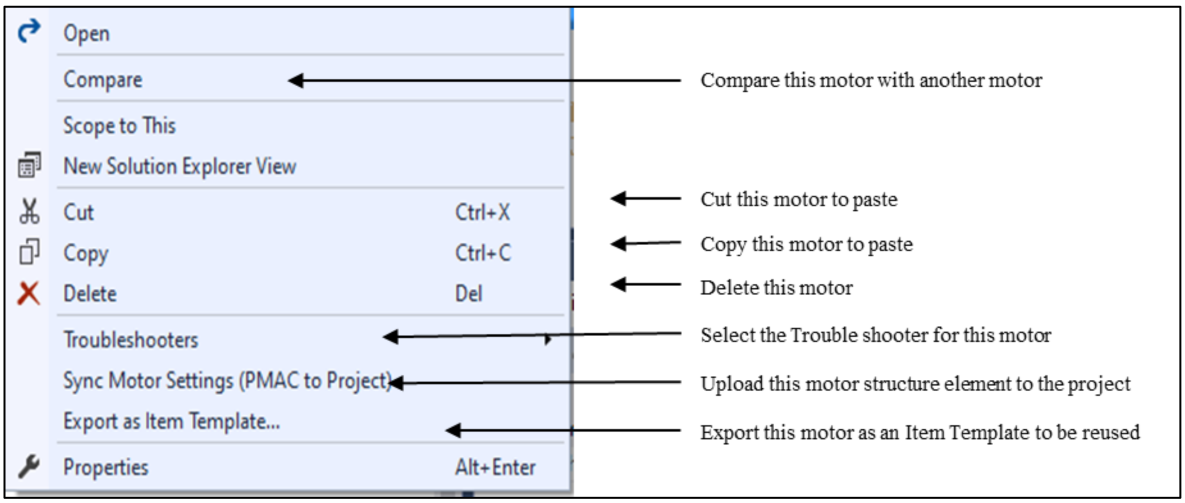


Sync All Motor Settings (PMAC to Project)

On selecting this option it will update the configuration for all the motor that are added under the motor Node. This command is useful to synchronize Motor structure element between Power PMAC and motors that are present in the project under Motor node.

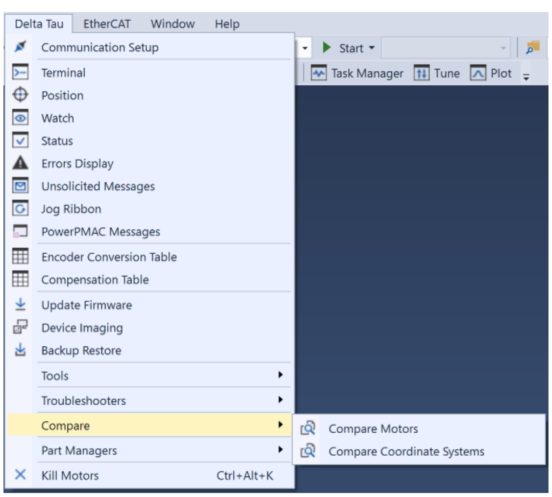
Motor – Context menu

This menu is available when any type of motor is added and displayed under Motors node.
Right-clicking on a motor node will open up a context menu containing various useful operations as shown below

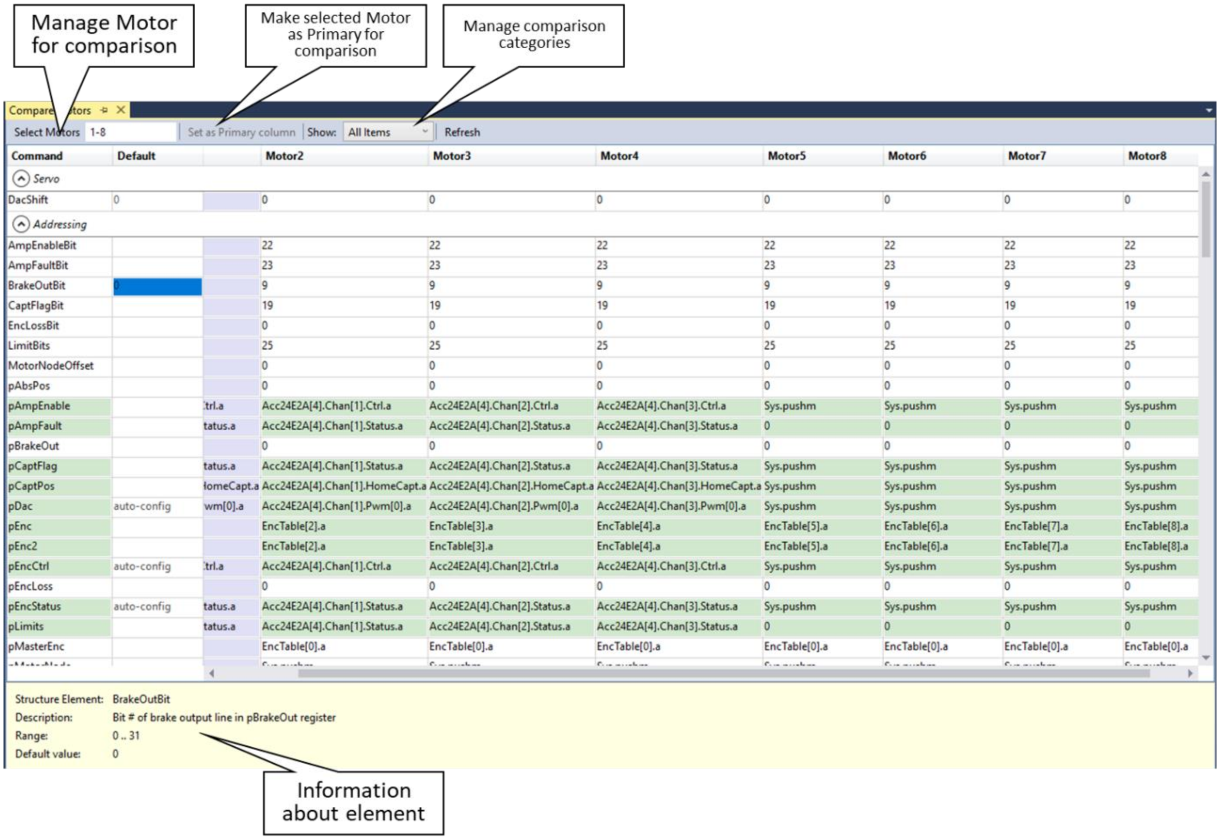


Compare

The compare feature is available for motors or coordinate systems. It allows the comparison of motor structure elements or coordinate system elements. The structure elements are categorized. A maximum of nine motors or nine coordinate systems can be compared at a time. The Compare motor function is available from the Delta Tau menu or by right clicking on the Motor in the Solution Explorer.
The following dialog shows the Compare feature being accessed from the Delta Tau menu.



The default view shows all the Motor structure elements. These can be hidden by selecting the arrow to the left of the name.



Copy

Right click on Motor to Copy motor settings. All the settings except addresses are copied for paste motor. .Copy motor not supported for virtual motor.

Note

Copy Motor function not available for Virtual(No Feedback) type motor topology

Paste

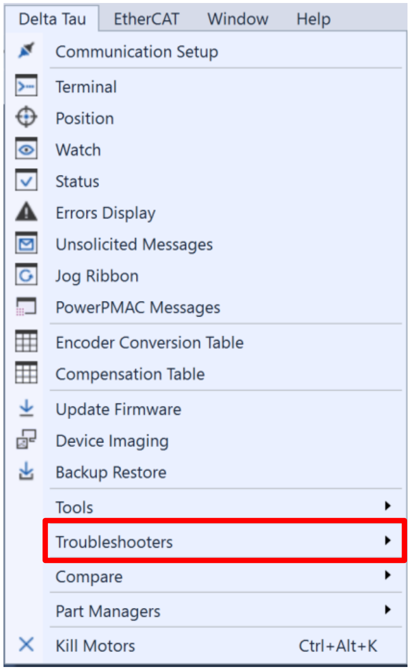
User can paste the motor by right clicking on Motors folder. This is dynamic menu if the Motor is copied only then this option will be available.

Only Amplifier, Motor, Encoder blocks are copied user will still require to go to Hardware interface and click Accept and then continue following the topology blocks

Troubleshooters

Troubleshooters are available which can generate reports and help in identifying or analyzing the Power PMAC structure elements. The menu is accesible from the Delta Tau Menu or by right clicking on the Motor in the Solution Explorer.

The following dialog shows the Compare feature being accessed from the Delta Tau menu.

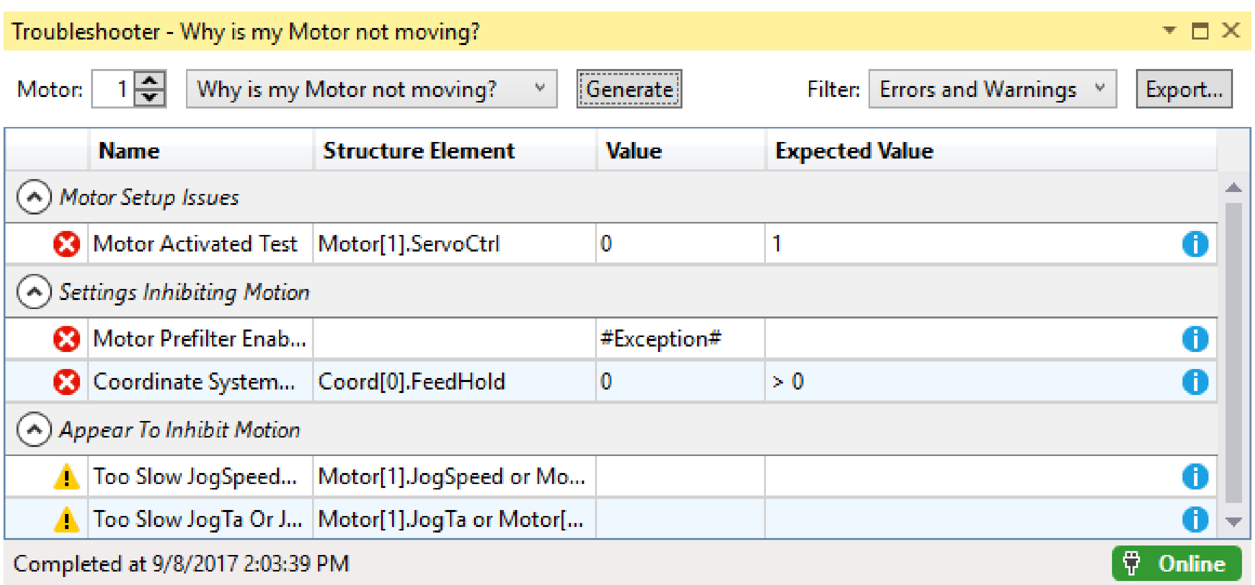


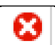
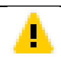
The available Troubleshooters are

1. Motor Report
2. Why is my Motor not moving
3. Why is my motor moving slowly

Layout

The dialog below shows the Troubleshooter for “Why is my Motor not moving”. The default location of this dialog is the Editor window. The dialog can be moved as required by dragging it to another docking point.

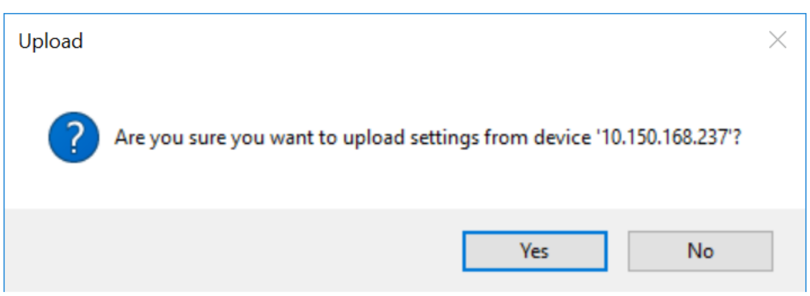


Symbols	Function
Motor: 1	Allows to change the motor number
Why is my Motor not moving?	This Combo box allows the selection of the troubleshooter type. The available Troubleshooters are: Motor Report Motor Report Why is my Motor not moving? Why is my Motor moving slowly?
Filter	This Combo Box allows the choice of what to display in the report. The possible choices are: All Errors and Warnings Errors Warnings All The default is set to Errors and Warnings.
Export...	To export the report in a .csv format.
	Indicates a Test has failed.There is an error in the setting of the setup element.
	Indicates a warning. Further analysis is needed for that particular setup element.
	Indicates that the Test has passed
	More detail information is available for the error or warning only.
Completed at 9/8/2017 2:40:58 PM	Status bar showing test execution progress.
Online	Indicates if the Power PMAC is either Online and connected or Offline and disconnected.

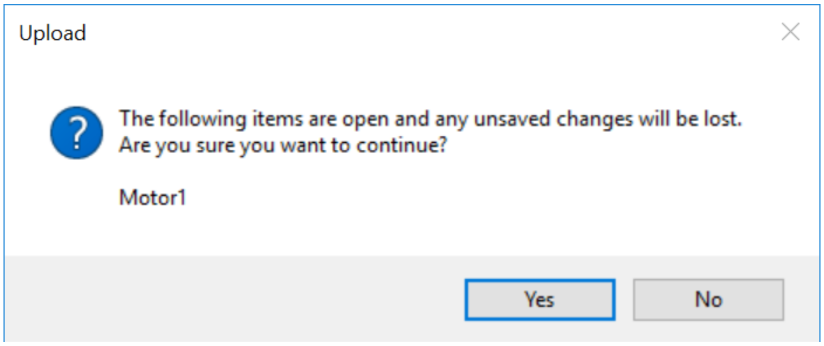
Sync Motor Settings (PMAC to Project)Upload

Upload motor gives the ability to upload the currently saved motor structure elements from the Power PMAC to the project.

On selecting this option, a confirmation dialog will be displayed as shown below:



On Clicking Yes, if the Motor View Editor is open in the IDE, a confirmation dialog will be displayed confirming that any unsaved data will be lost by performing the upload.



On a successful upload the motor in the project will be synchronized with the Power PMAC motor structure elements.

Note

This option is useful if the Motor structure element has been changed outside of project domain such as in the Terminal window.

Export as Item Template

The Motor can be exported or imported as item templates. All the motor settings will be exported during this process.

The typical use of the Motor template is to setup a complete Motor, including Custom Amplifier and encoder, and then share this with another user.

This User can then Import the Motor, using Import item template option, and use it in their project saving the time of having to create the Motor from new.

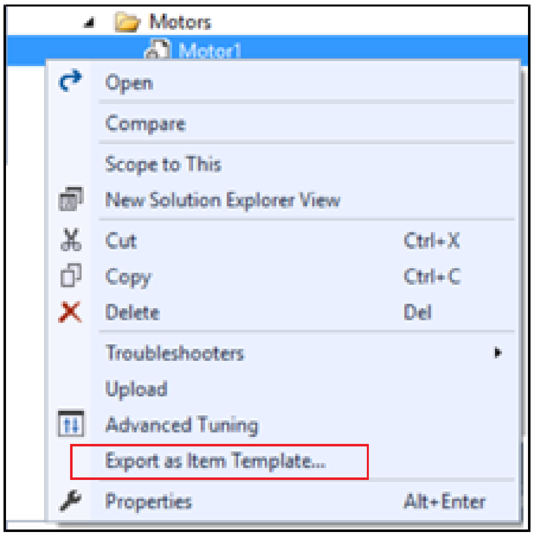
If the Power PMAC hardware is identical then user will not need to do complete motor setup for the imported motor.

Using this option the User can:

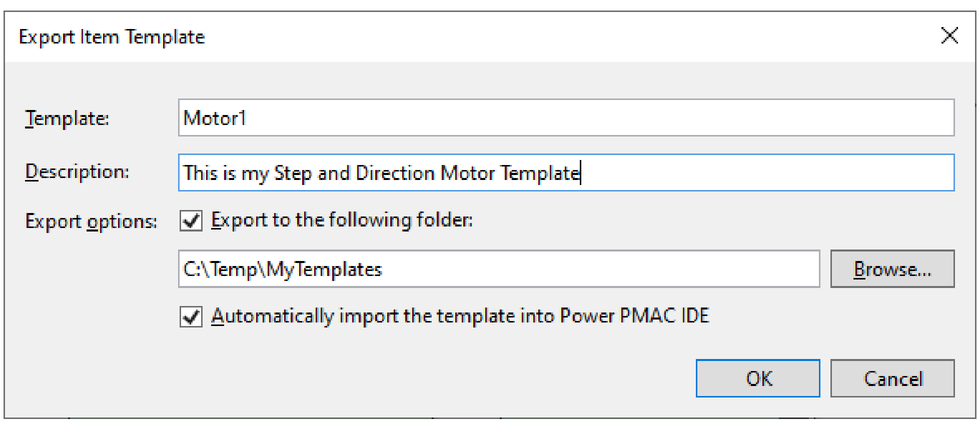
- Export a Motor in order to use it in another project
- Import a Motor to reuse in their own project
- Create a new Motor/ from an Imported Motor/ Item Template
- Choose whether or not to automatically Import an Item Template into Power PMAC IDE project at the point that it is exported
- Use a motor template based on a custom amplifier or motor definition that is not present in their system
- Is warned if they try to Export a motor template targeting multiple gate addresses
- Is warned if they try to create a motor from a template and their system does not have a suitable gate available so that it is clear that the Hardware Interface page will need to be updated

- Check they have the correct template by viewing the motor manufacturer and model number in the template
- Be sure that a motor that is created from a template will have the correct encoder information as this is saved on creation of the template
- Delete imported Custom Item templates

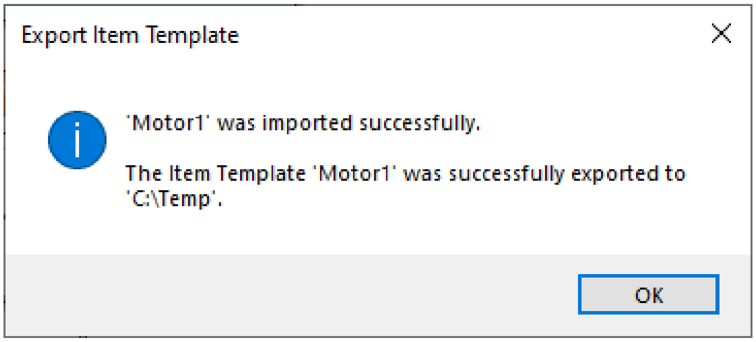
To export a Motor settings as a Template the user will need to right click on Motor node under Motors and choose the "Export as Item template" option as shown below:



On selecting the option, a new export item template dialog will be displayed, as shown below:

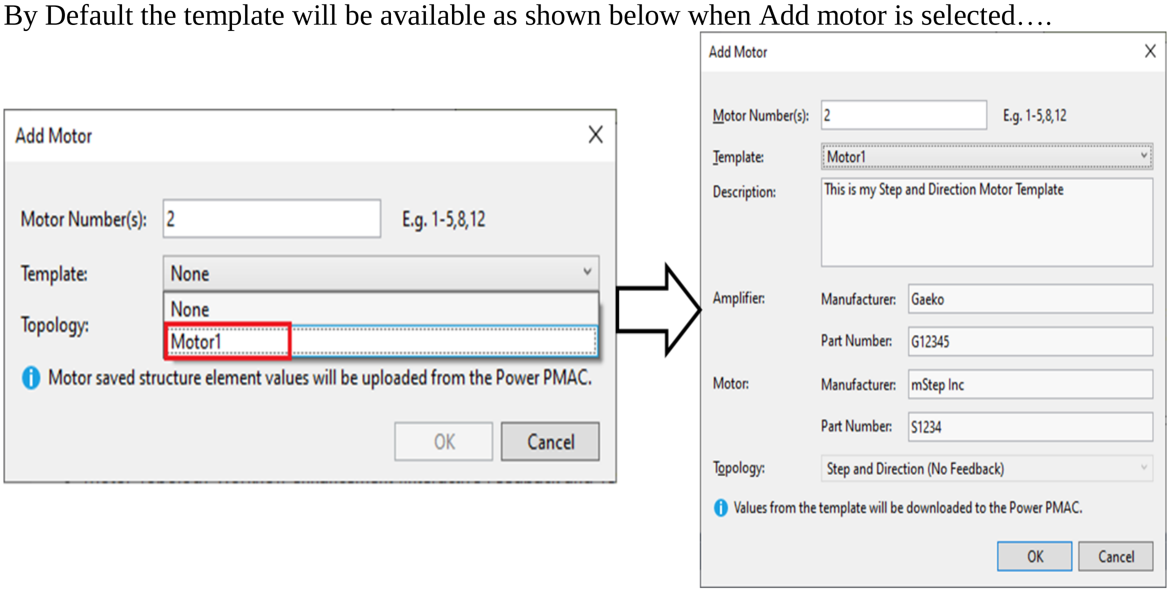


On selecting Ok the acknowledge message will be displayed and template will be exported and stored into the location defined in the dialog.



The User has ability to store the template to any folder by ticking the “Export to the following folder” checkbox.

By default, the template will be imported to be used in the current instance of the IDE. Un-ticking this check box will not import the template into the current instance of the IDE.
By Default the template will be available as shown below when Add motor is selected….



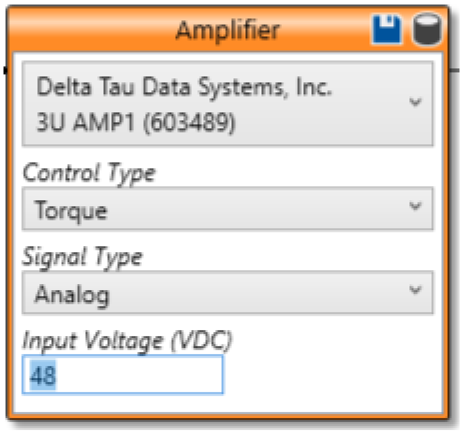
Topology Blocks

Note

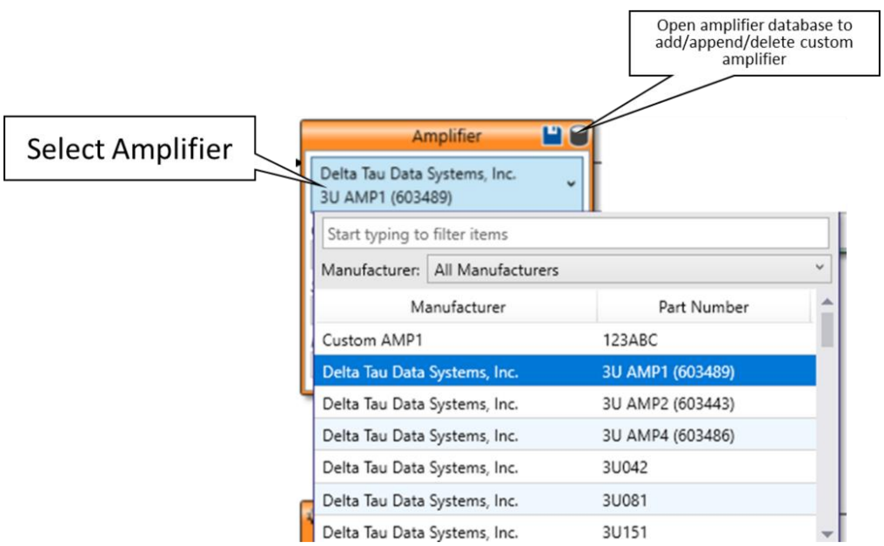
Please make sure that it is safe to setup a Motor using System Setup.
Following Topology Blocks sets Power PMAC structure element.
User Unit,
Hardware Interface Block,
Interactive Feedback Block,
Test and Set,
Basic Tuning,
commissioning block.

Amplifier block

The User can select the Amplifier from the topology block as shown below. As displayed in this screen the Amplifier can be selected from a list of Delta Tau Amplifiers or, if the Amplifier is not listed, can be added i.e. if a 3rd party amplifier is being used:

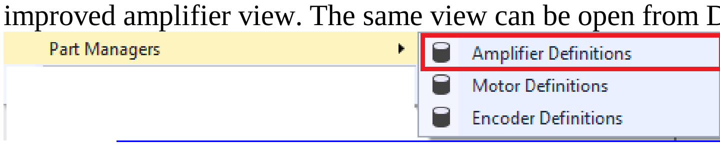


A standard filter is available to choose the amplifer. The User can choose the control and signal type and input voltage right on the topology block and press Save icon to set the amplifier for the motor. On success the block will turn Green with chek mark indicator.



From IDE V4.3 the amplifier database view is changed.

To add a new Amplifier entry into the database, click on the database  icon . This will open new and improved amplifier view. The same view can be open from Delta Tau menu under Part Managers.



Note

The User does not need to open a project and add Motor to add Amplifier/Motor/Encoder parts in the database.

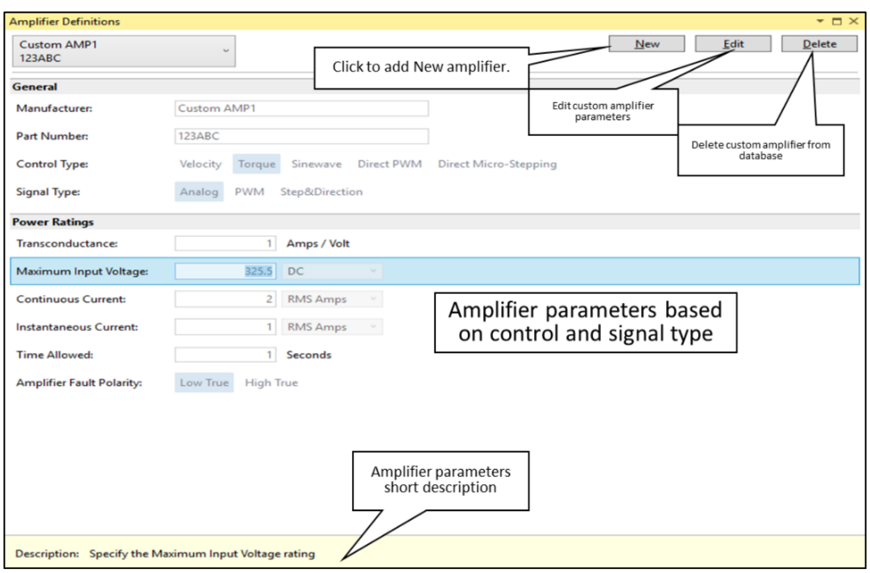
The amplifier part manger view looks like this when opened from Topology block.

The User cannot edit or delete if the amplifier is a Delta Tau Amplifier. To add a new amplifier press 'New' and enter the amplifier parameters from the amplifier manufacturer brochure.

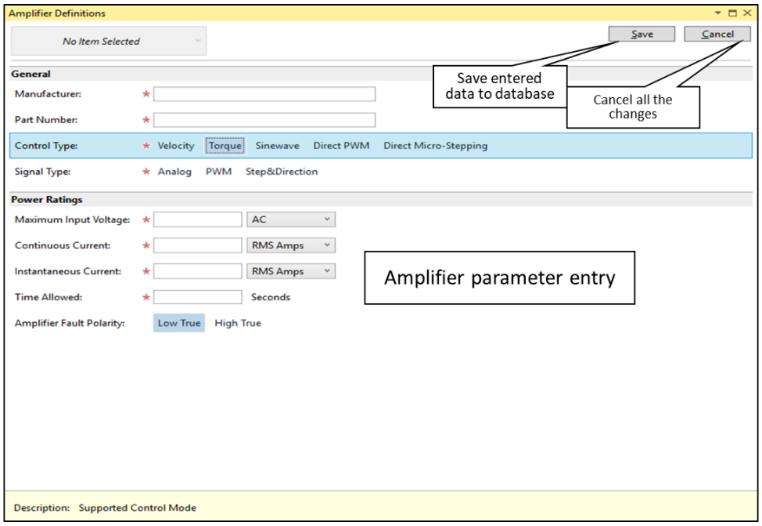
To edit a saved amplifier's parameters, select the amplifier from the drop down and then press 'Edit'.

To delete the amplifier from the database, select the amplifier from the drop down and then press 'Delete'.

The amplifier part manager view looks like this when open from Delta Tau Menu…



The add new amplifier view looks like this…



Settings parameters are dynamic based upon the control and signal type.
Once all Amplifier parameters are entered click Save.

Amplifier Parameters

The Amplifier parameters needed are described in detail below:

Amplifier Manufacturer

- Manufacturer: The name of the company which makes the Amplifier.
- Part Number: A unique part number to identify the Amplifier's model.

Supported Control Mode

- Velocity Control: Set this to True if the Amplifier interprets the control signal it receives from the Power PMAC as a velocity command e.g. this is common for Amplifiers which close their own position loop such as Amplifiers commonly used for spindles.
- Torque Control: Set this to True if the Amplifier interprets the control signal it receives from the Power PMAC as a torque command. In this mode the Power PMAC closes its own position and velocity loops. This is the recommended mode for most applications as it permits complete control over the current, position and velocity loop gains from within the Power PMAC.
- Sinewave Commutation: Set this to True if using two DAC lines per motor to command an amplifier which performs Sinusoidal Commutation.
- Direct PWM Control: Set this to True if using a Direct PWM amplifier which expects a PWM control signal.

Supported Signal Type

- Analog Command: Set this to True if the Amplifier expects to receive an analog voltage as its control signal.
- PWM Command: Set this to True if the Amplifier expects to receive a PWM signal as its control signal.
- Step and Direction Command: Set this to True if the Amplifier expects a Step and Direction (PFM) command as its control signal.

Power Ratings

- Maximum Input Voltage
    - Voltage (Volts): Specify the maximum bus voltage which can be applied to the Amplifier.
    - Type: Specify VAC if the number typed in the Voltage field is AC voltage or specify VDC if that number is DC voltage.
- Continuous Current
    - Continuous Current (Amps): Specify the continuous current rating for the Amplifier.
    - Unit: Specify whether this is Amps RMS (type Amp_RMS) or Peak Amps (type AMP_Peak).
- Instantaneous Current
    - Instantaneous Current (Amps): Specify the instantaneous current rating for the Amplifier.
    - Unit: Specify whether this is Amps RMS (type Amp_RMS) or Peak Amps (type AMP_Peak).
- Time Allowed (Seconds): Specify the maximum amount of time the Amplifier can tolerate its instantaneous current specification. Usually this is around 2.0 seconds, but it can vary between Amplifiers.
- Input Voltage (VDC): Specify the actual amount of voltage [VDC] to be applied to the Amplifier. This parameter is moved to topology block.
- Amplifier Fault Polarity: If the Amplifier expects a low-true logic signal for an Amplifier fault set this to LowTrue. If the Amplifier expects a high-true logic signal for an Amplifier fault set this to HighTrue.

Current Feedback Information

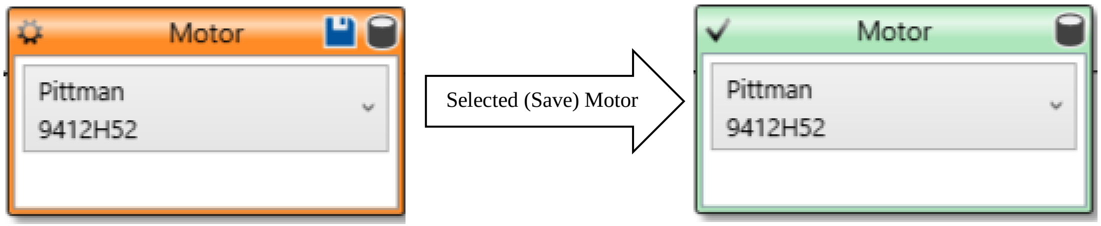
- Maximum ADC Current: This is the largest absolute magnitude of current [Amps] which the Amplifier’s current ADC sensors can read.
- ADC Header Bits: This is the number of bits used for the current ADC’s status.
- ADC Resolution (bits): This is the resolution [bits] of the Amplifier’s current ADCs.
- PWM Dead-Time (microseconds): This is the dead-time specified for the Amplifier.

Note

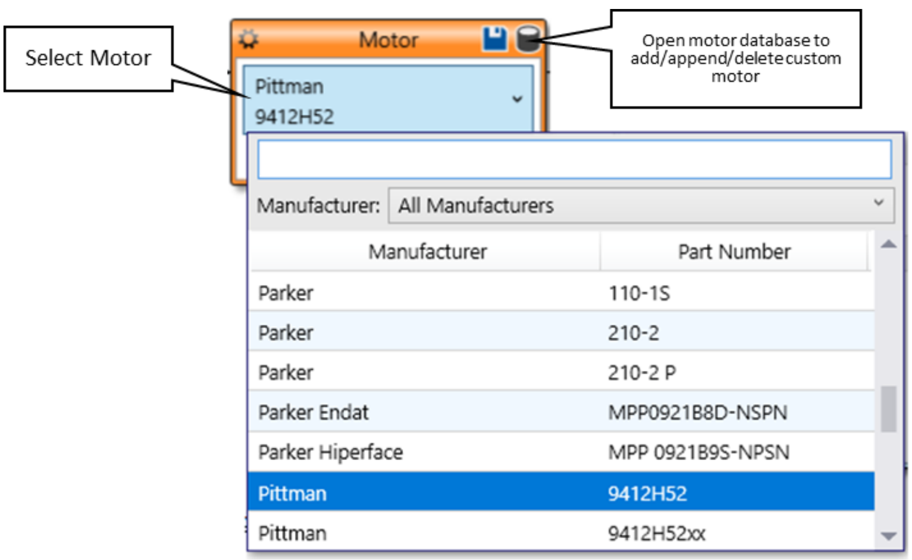
Verify all the contents of each of the fields specific to the Amplifier’s parameters before moving on as these will be used in subsequent setup calculations.

Motor Block

The User can select the Motor from the topology block as shown below. As displayed in this screen the Motor can be selected from a drop-down list. If the Motor is not listed, then it can be added.

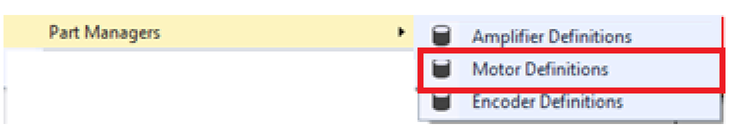


A standard filter is available to choose the motor. The User can choose the motor and click the Save icon to set the motor. On success the block will turn Green with chek mark indicator.



From IDE V4.3 the motor database view is changed.

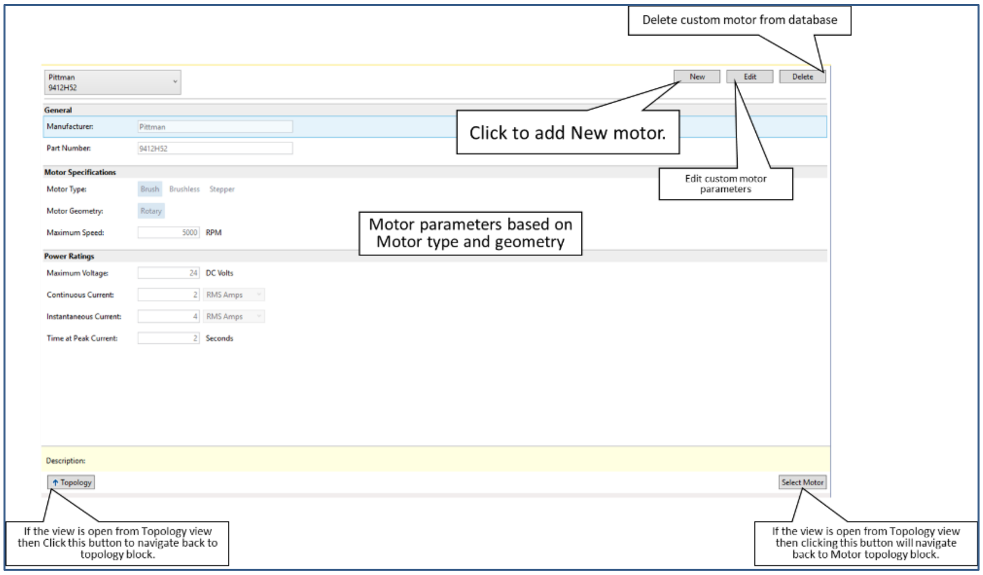
To add a new Motor entry into the database, click on the database icon . This will open new and improved motor view. The same view can be open from Delta Tau menu under Part Managers.



Note

The User does not need to open a project and add Motor to add Amplifier/Motor/Encoder parts in the database.

The motor part manger view looks like this if opened from Topology block.

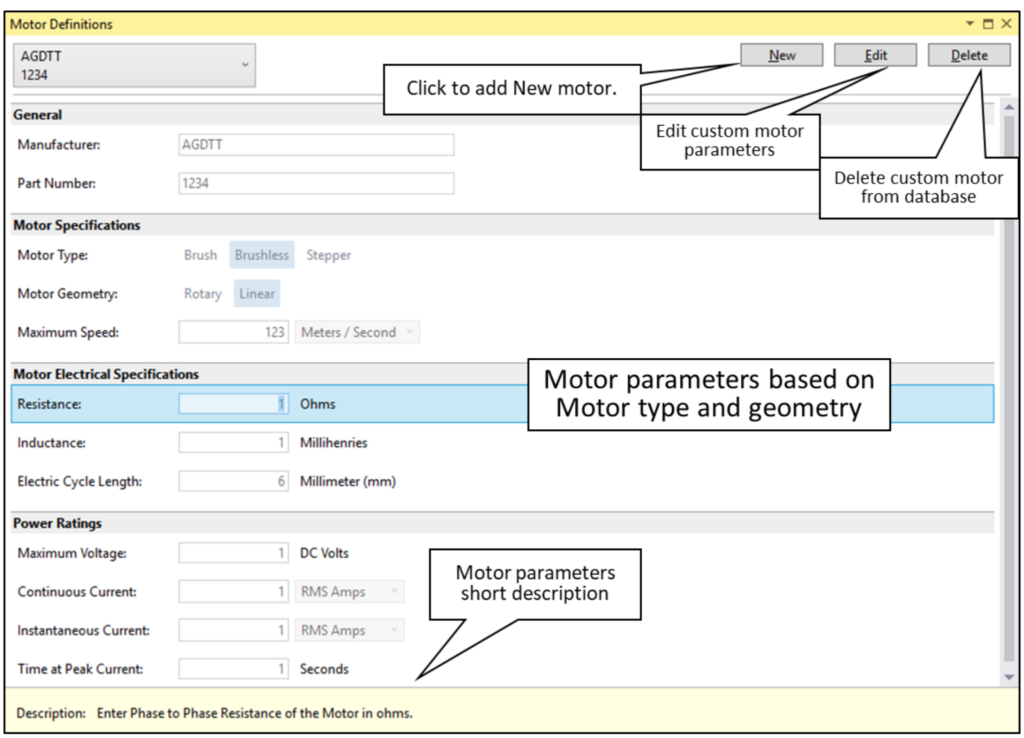


To add new motor press ‘New’ and enter the motor parameters from the motor manufacturer brochure.

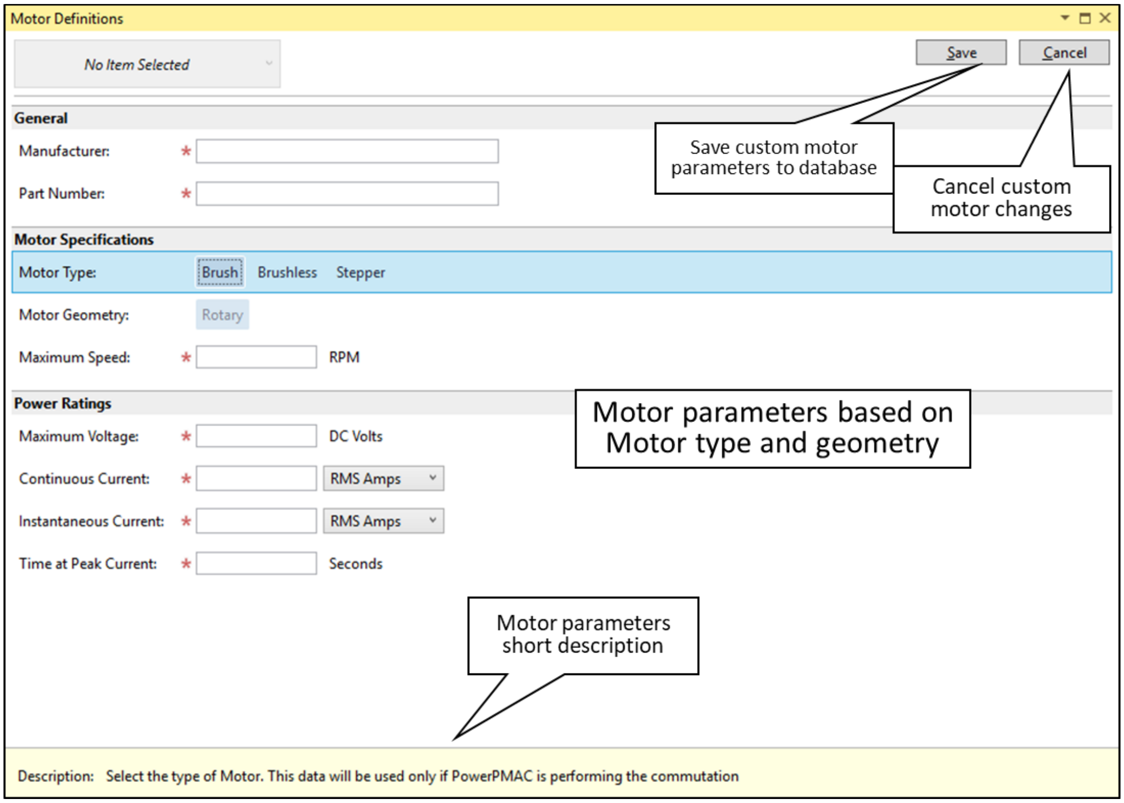
To edit the saved motor parameters, select the motor from the drop down and then press ‘Edit’.

To delete the motor from database, select the motor from the drop down and then press ‘Delete’.

The motor part manager view looks like this when open from Delta Tau Menu…



Add New Motor view looks like this…



Motor Parameters

The motor parameters which are needed are described in detail below:

Motor Manufacturer

- Name: The name of the motor's manufacturer.
- Part Number: The manufacturer's part number for this motor.

Motor Specifications

- Motor Type: The type of motor, whether it is Brush or Brushless.
- Nominal RPM: The rated continuous RPMs for this motor.
- Maximum RPM: The maximum possible RPM rating.
- Linear Motor: Set this to True if using a linear motor else set this to False.

Motor Electrical Specifications

- Inductance (mH): The phase-to-phase inductance of the motor in millihenries.
- Resistance (Ohms): The phase-to-phase resistance of the motor in Ohms.
- Number of Poles: The number of poles the motor has.
- Delta Winding: Set this to True if this motor has a Delta Winding or else set this to False.

Motor Built-In Feedback

- Absolute: Set this to True if this motor has an absolute feedback sensor or else set this to False.
- Feedback Type: Specify what kind of feedback this motor has or if there is no feedback set this to None.
- Resolution: Specify the resolution of the encoder in counts per revolution. For serial protocols use units of Least Significant Bits (LSB). For linear motors use the number of encoder counts per electrical cycle of the motor.
- Hall Sensor Available: Set to True if this motor has a Hall Sensor it can use for feedback.

Motor Power Rating Specifications

- Continuous Current
    - Continuous Current (Amps): The amount of current [Amps] which the motor can safely sustain for an indefinite period of time.
    - Current Unit: Select Amp_Peak if the continuous current limit is in units of Amps Peak otherwise select Amp_RMS.

- Instantaneous Current
    - Instantaneous Current (Amps): The amount of current [Amps] which the motor can sustain for only a finite period before being damaged. This time is specified in “Time Allowed” below.
    - Current Unit: Select Amp_Peak if the instantaneous current limit is in units of Amps Peak otherwise select Amp_RMS.

- Time Allowed (Seconds): The maximum amount of time during which the motor can sustain the amount of current specified by the Instantaneous Current limit.

Rating

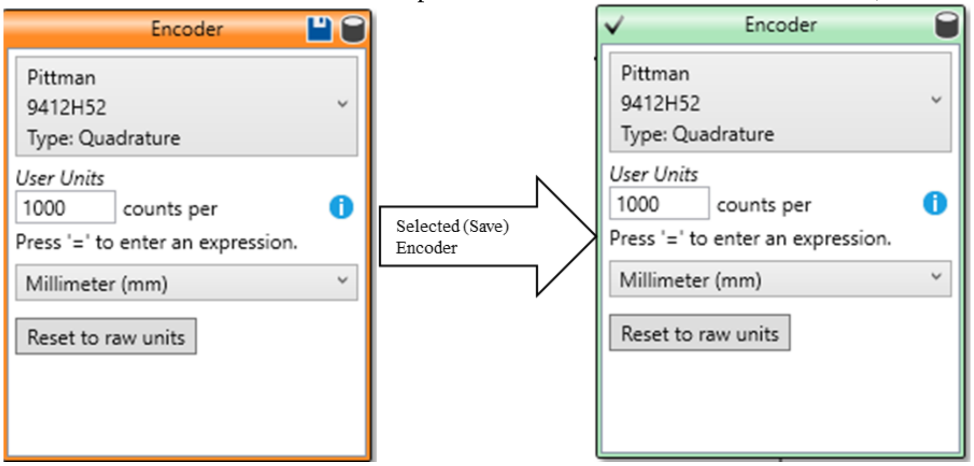
- Maximum Voltage (VDC): The maximum amount of DC voltage which can be supplied to the motor before damaging the motor.

Note

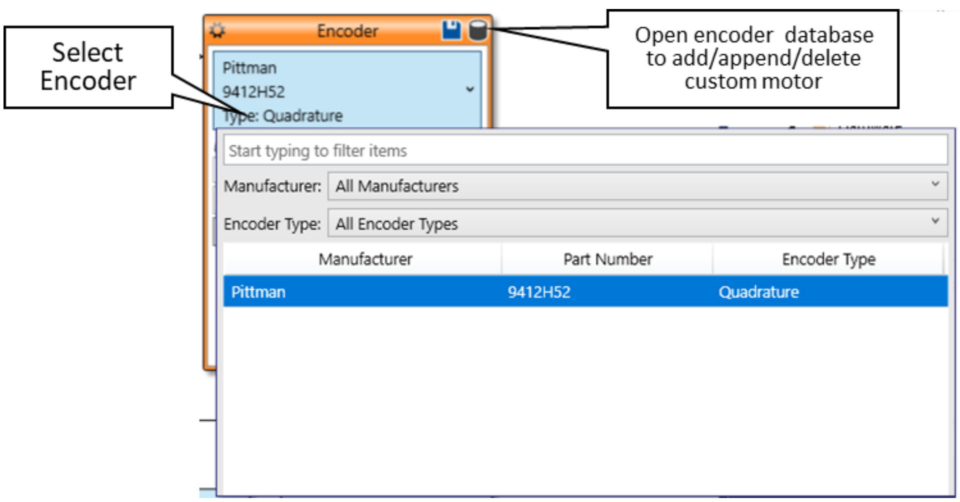
Verify all the contents of each of the fields specific to the Motor’s parameters before moving on as these will be used in subsequent setup calculations.

Encoder Block

The User can select the Encoder from the topology block as shown below. As displayed in this screen the Encoder can be selected from a drop-down list. If the Encoder is not listed, then it can be added.

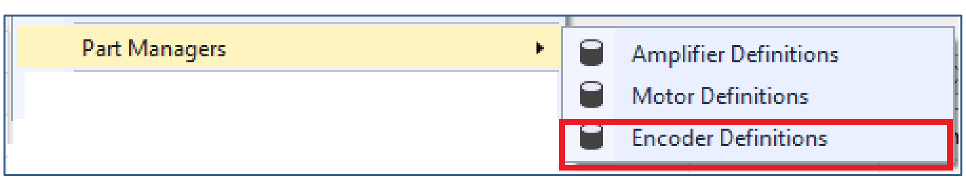


A standard filter is available to choose the encoder. The User can choose the encoder and press Save icon to set the encoder. On success the block will turn Green with check mark indicator.



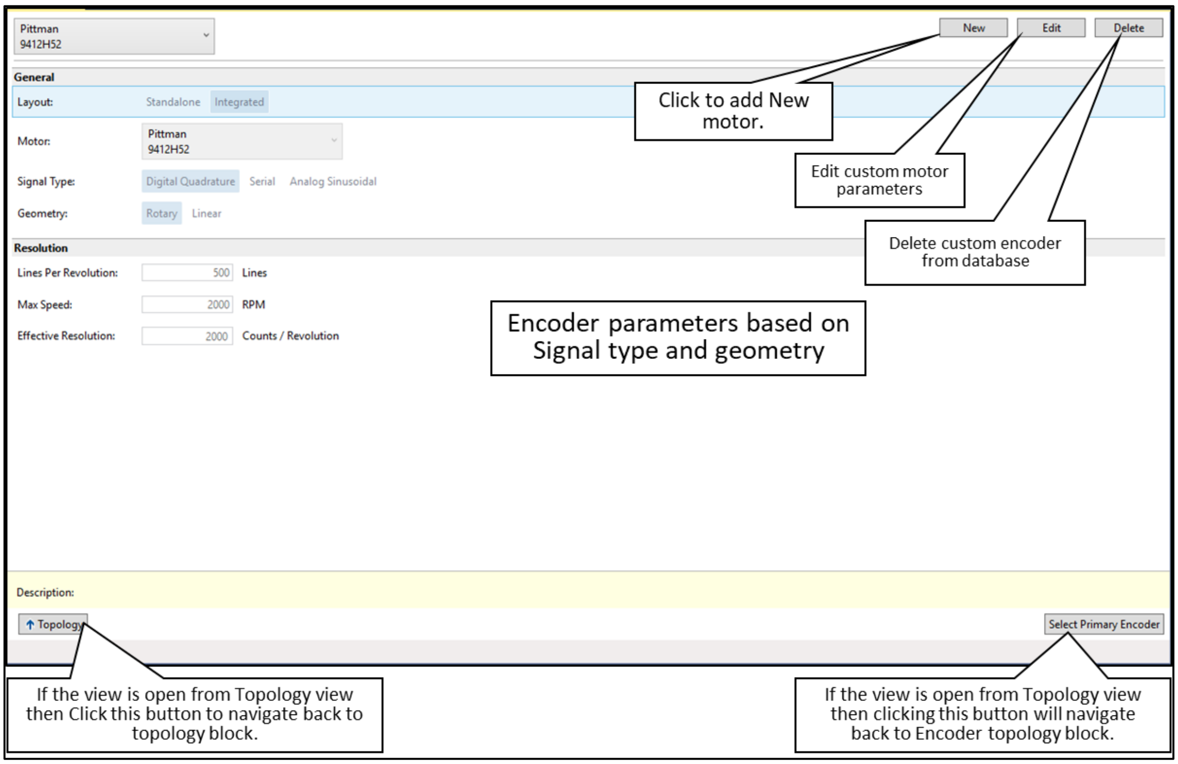
From IDE V4.3 the encoder database view is changed.

To add a new encoder entry into the database, click on the database  icon. This will open new and improved encoder view. The same view can be open from Delta Tau menu under Part Managers.



Note	1. Encoder drop down list is dependent upon the detected Power PMAC hardware. If the User cannot see the encoder that means it is not supported by the detected hardware. 2. User units are part of Encoder topology block 3. The User does not need to open a project and add Motor to add Amplifier/Motor/Encoder parts in the database.

The encoder part manger view looks like this if opened from Topology block.

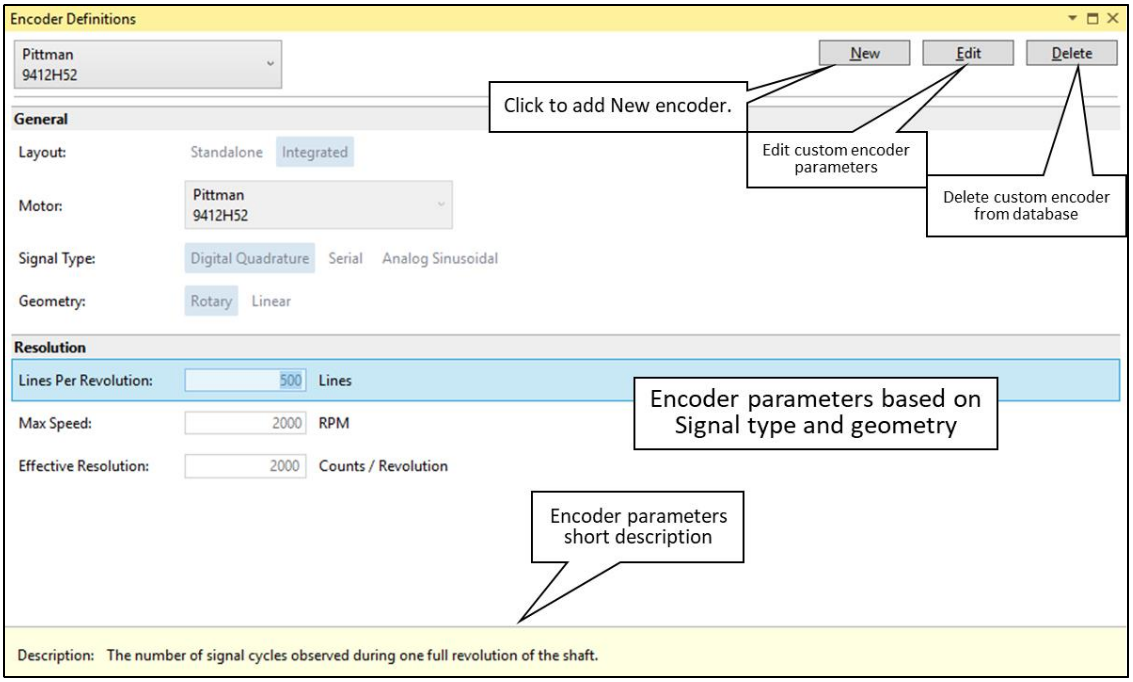


To add new encoder press ‘New’ and enter the encoder parameters from the encoder manufacturer brochure.

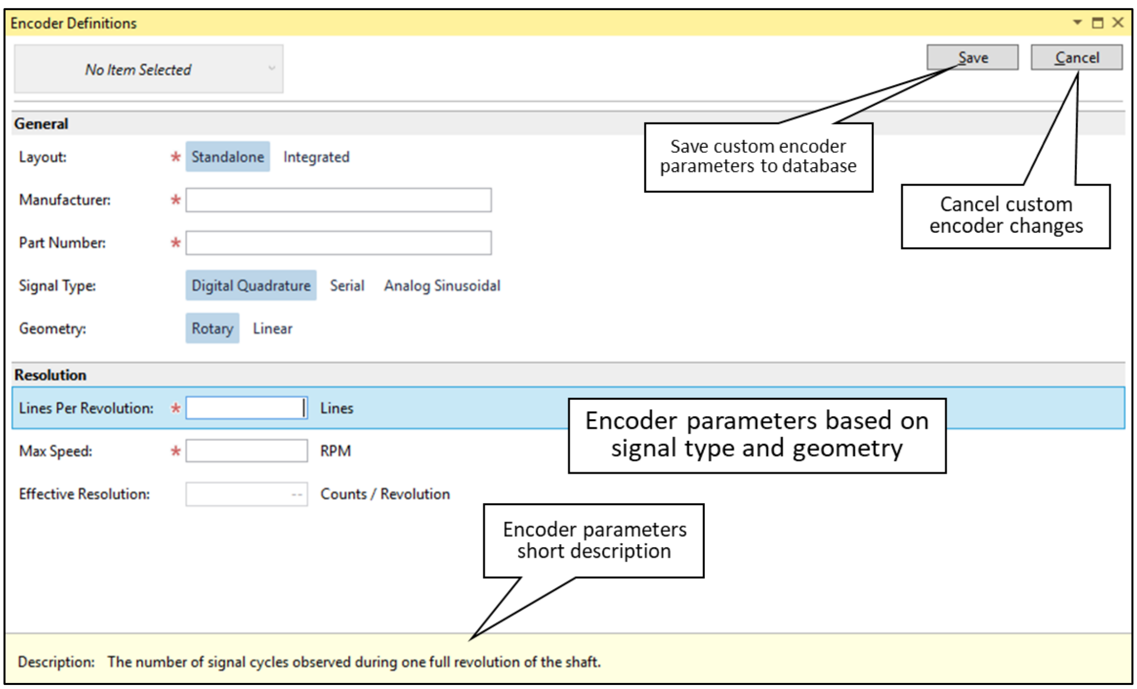
To edit the saved encoder parameters, select the encoder from the drop down and then press ‘Edit’.

To delete the encoder from database, select the motor from the drop down and then press ‘Delete’.

The encoder part manager view looks like this when open from Delta Tau Menu…



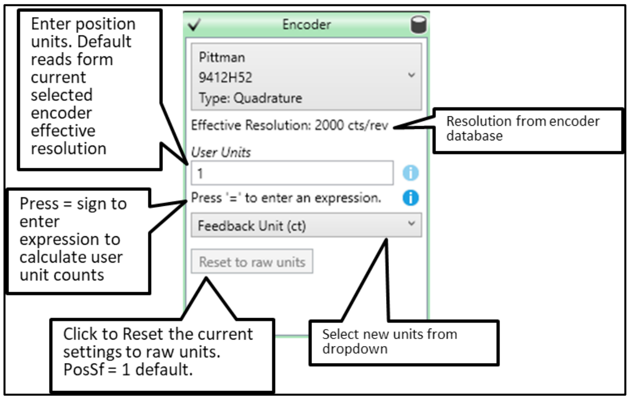
Add New Encoder view looks like this…



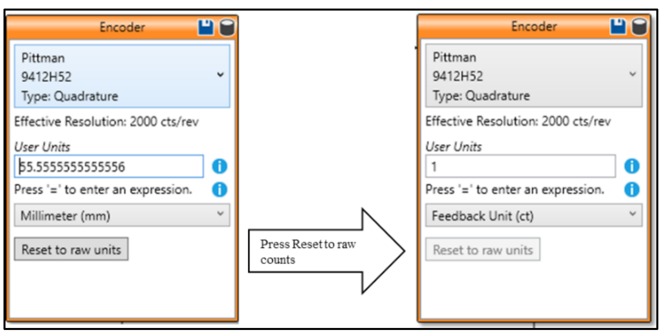
User Units Block

The User Units enables the setting of the Motor position units in terms of engineering units like mm, inch, meters, etc.

This block is used to make it easy to change the defined units for a motor even if the configuration for the motor has been completed.

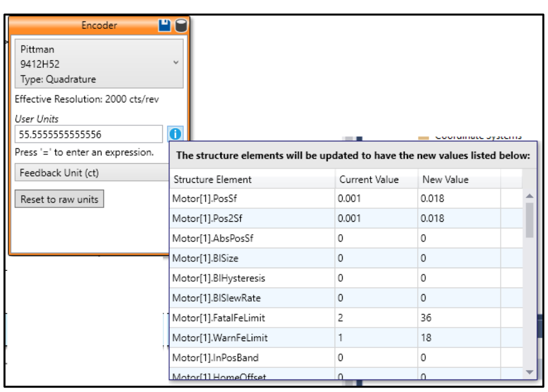


If an incorrect value is entered, then use $$$*** command to go back to default settings or use Reset to raw units. When Reset to raw units is clicked following are the changes in the Encoder Topology Block…



As soon as the units are changed the PosSf are recalculated and the User can view the newly calculated value by clicking on icon.

The view looks like this…



The dropdown list for User Units is represented by Motor[x].posunit as shown below.

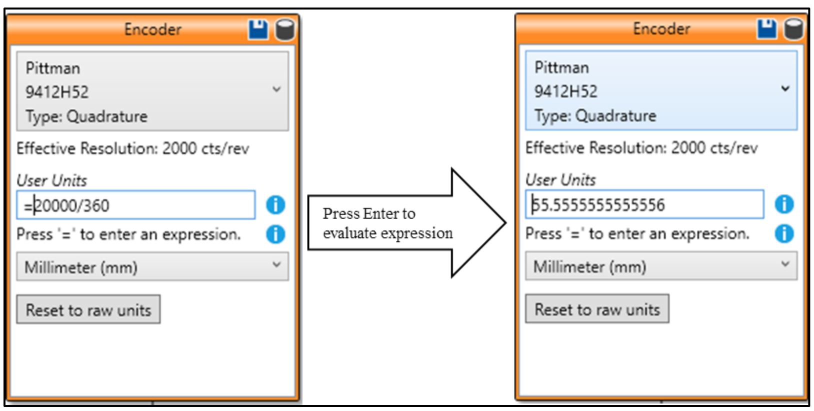
Motor[x].PosUnit	Selected Unit	Motor[x].PosUnit	Selected Unit
0	*None selected*	8	Mil (in/1000)
1	Feedback unit (ct)	9	Revolution
2	Meter (m)	10	Radian (rad)
3	Millimeter (mm)	11	Degree (deg)
4	Micrometer (µm)	12	Gradian (grad)
5	Nanometer (nm)	13	Arcminute (')
6	Picometer (pm)	14	Arcsecond ('')
7	Inch (in)	15	*Reserved*

Note

For example, when User Units are selected for motor 1 then the Coordinate System axis definition for that motor is simply #1->X. This will allow the user to command the motor in User Units. If the motor units for Motor 1 are in mm then #1J1 is 1 mm command and so on and so forth.

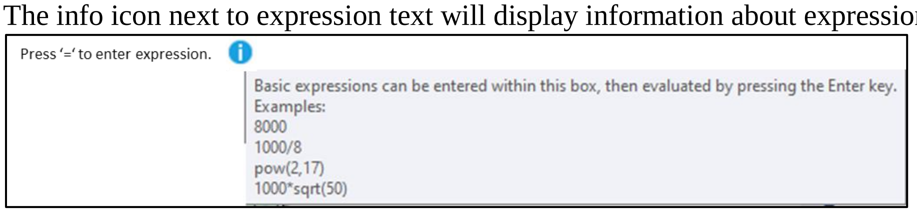
Calculating User units count by entering expression

Press = sign in the User units text block and start typing expression. Press enter to evaluate the expression and show the result in the User Units text box. As shown below…



The expression is stored in the project so next time when the project is opened and user opens the Motor topology and hoover the mouse on the User Units block the tool tip will show the expression.

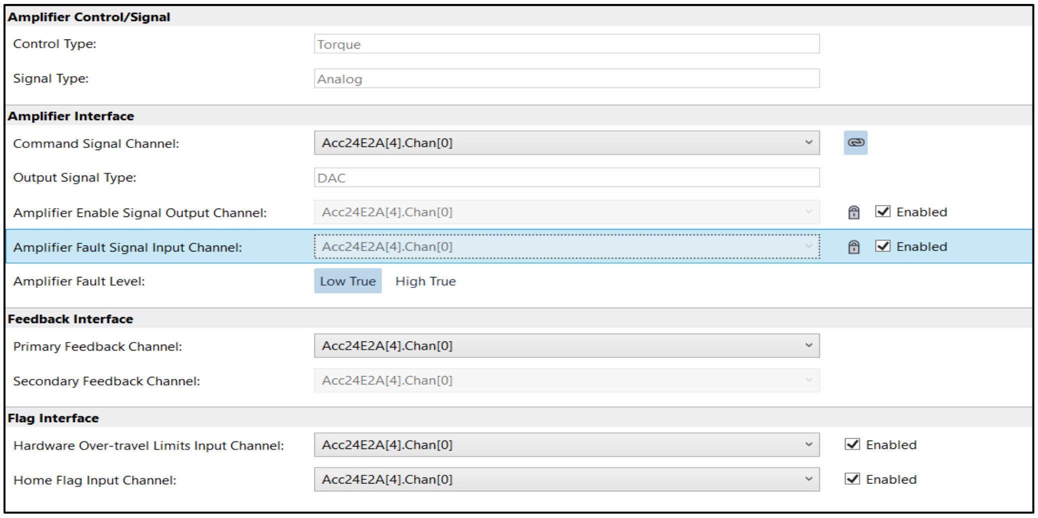
The info icon next to expression text will display information about expression as shown below.



Hardware Interface Block

Proceeding to the Hardware Interface step of the System Setup will show this screen:

The expression evaluator available in all topology types **except** Galvo and Virtual.



This part of the System Setup configures command control signals being produced from Axis Interfaces in the system and amplifier-related flags which these Axis Interfaces read from or send to the amplifier.

For UMAC, these Axis Interfaces will usually be ACC-24E2, ACC-24E2A, ACC-24E2S, or ACC-24E3. Each field is described in detail below:

Amplifier Control/Signal

- Control Type: Displays the type of control signal (Position, Velocity, Torque, Sinusoidal, Direct PWM, or Direct Micro stepping) that the amplifier connected to this Axis Interface's channel supports.
- Signal Type: Displays selected signal type (Analog, Direct PWM, or Step & Direction).

Note

On this page these two parameters are read only so, in order to make a change, the User must go back to Command and Feedback type page.

Amplifier Interface

- Amplifier Advanced Interface Mode: Setting this to True permits gives the ability to obtain the AmpEna and AmpFault bits from different locations than the address of the channel which produces the command signal. Setting this to False assumes that AmpEna and AmpFault come from the channel which produces the command signal (Command Signal Channel; see next field).
- Command Signal Channel: Specify the structure of the channel which sends the command control signal to the Amplifier.

- Amplifier Fault Level: Select Low True if the Amplifier expects a Low-True signal to indicate an amplifier fault. Select High True if the Amplifier expects a High-True signal to indicate an amplifier fault.
- Amplifier Enable Signal Output Channel: Specify the structure of the channel which produces this motor's Amplifier enable signal.
- Amplifier Fault Signal Input Channel: Specify the structure of the channel which produces this motor's Amplifier fault signal.

Feedback Interface

- Dual Feedback Interface Mode: If the motor has separate encoders for position and velocity feedback then select True.
- Primary Feedback Channel: Select the structure of the primary feedback channel; typically, this is the position feedback channel.
- Secondary Feedback Channel: Select the structure of the secondary feedback channel; typically, this is the velocity feedback channel. This property is grayed out for Single Feedback Motor Topology but available for edit for Dual Feedback Motor Topology.

Flag Interface

- Hardware Overtravel Limits Input Channel: Select the Axis Interface channel which reads the hardware overtravel limits.

Note

If "Hardware Mismatch" error message is displayed it is probably because the chosen control type or signal type is not compatible with the Amplifier chosen in the Amplifier Information section of the setup or with the motor type chosen in the Motor Information section.

- Home Flag Input Channel: Select home flag input channel from available list. Usually the default is not needed to change.

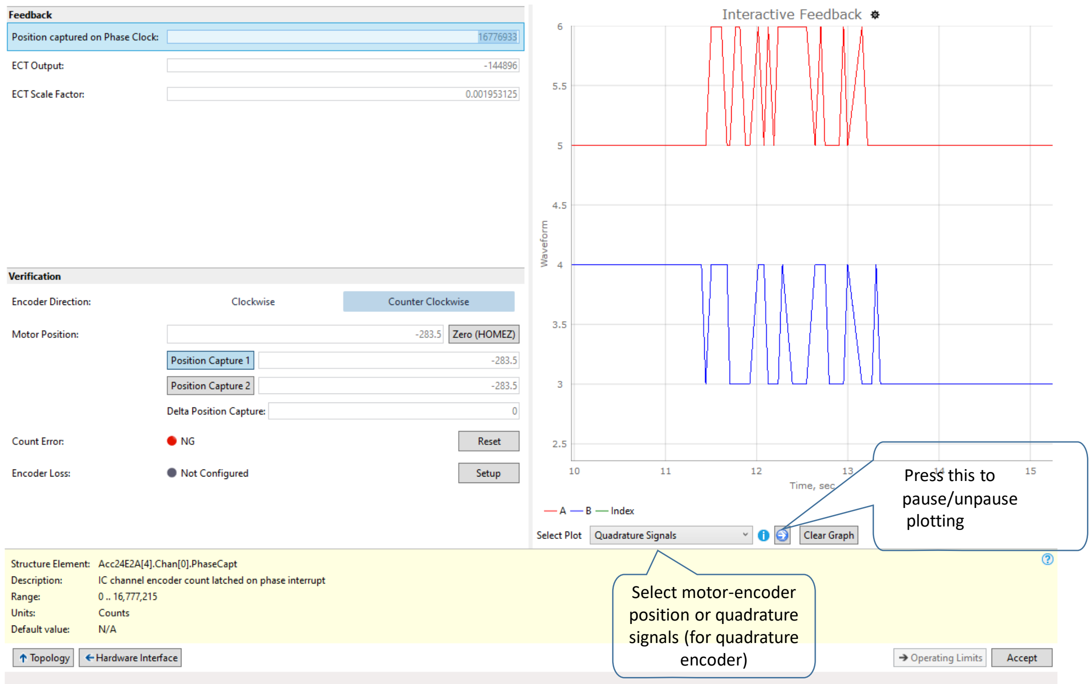
Interactive Feedback Block

The Interactive Feedback screen displays real-time plots of the feedback devices associated with the motor on the right side and fields containing feedback-related data. The purpose of this screen is to help determine whether the encoder feedback is working properly. The User can try to physically move the encoder by hand and observe whether the feedback can been seen to change on the screen.

Note

The contents of the Interactive Feedback screen change greatly depending on which kind of feedback is selected. The following example is for Quadrature Encoders which are very common encoder types. If using another encoder type, like an absolute serial encoder, this screen would configure the number of bits of feedback data, absolute power-on position and phasing, and other parameters relevant to that encoder type.

The screen below shows a Quadrature Encoder:



The left axis of the plot shows the units of the encoder output's waveform while the bottom axis shows time passing in units of seconds. "ServoCapt", indicated by the red curve, is **EncTable[x].PrevEnc**; the ECT output before being scaled by EncTable[x].ScaleFactor. "Motor Input", indicated by the blue curve, is **Motor[x].Pos**; that scaled output of the ECT entry.

Note

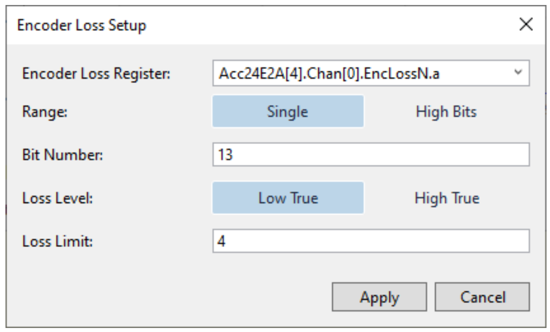
If the "Motor Input" curve does not change as the encoder spins, make sure that the Accept button has been clicked on the previous screen; the Hardware Interface screen.

The fields shown on this screen are described below:

- Position Captured on Phase Clock: This is the encoder's position captured at the Power PMAC's phase frequency.
- ECT Output: This is the encoder tabel entry previous input value.
- ECT Scale Factor: The ECT will read the encoder's address, perform the shifting specified in **EncTable[x].index1 and EncTable[x].index2**, and then multiply this value by the ECT Scale Factor, **EncTable[x].ScaleFactor**, before producing the final output of the ECT entry.
- Encoder Direction: Let the user set the positive direction of the motor to Counter Clockwise or Clockwise (for Rotary Encoders only).
- Motor Position:
    - Zero (HomeZ): It sets the Motor 1 position to 0
    - Position Capture 1: Captures the motor's position
    - Position Capture 2: Captures the motor's position

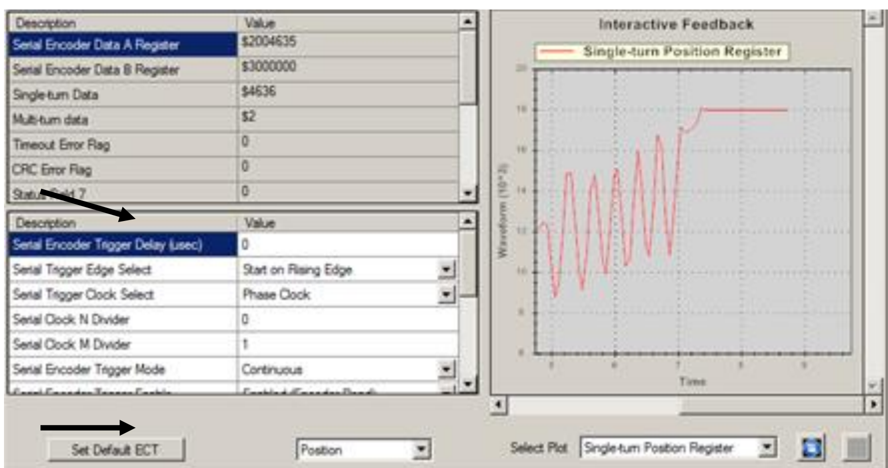
- Delta Position Capture: Displays the difference between Position Cpature 1 and Position Capture 2.

- Count Error: If there are any encoder count error, it will be displayed here. The user can reset this error by clicking on Reset button.
- Encoder Loss: if there are any encoder loss, this status bit will be set to show the loss error. To be able t detect the encoder loss, the user must use th efollowing configuration page by clicking on the Setup button. In te page the suer must specify encoder information and the max allowes loss limit to detect if the encoder count loss is greater that the loss limit.



The screen below shows a serial encoder, in this example from a Panasonic MSMD082S1S encoder:

This image is from the V3.x but works same way in Newer IDE with newer screen layout



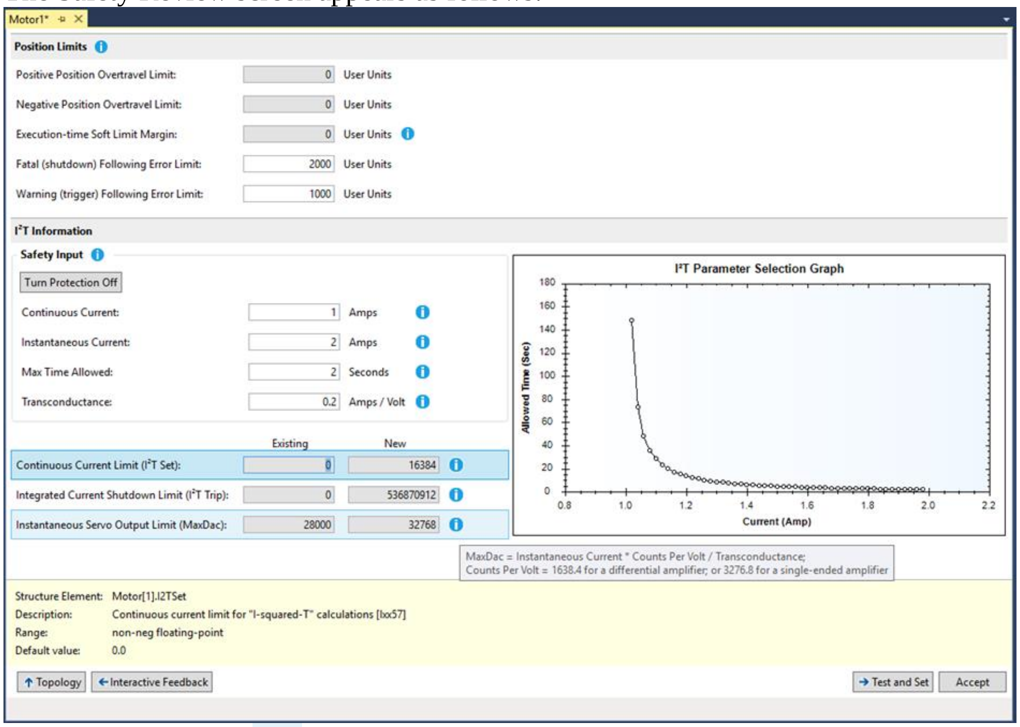
Note	Absolute phase position feedback in parallel/serial mode only supports binary data. If an encoder has Gray code mode, the conversion from binary to Gray code will take place in the hardware (DSPGate or FPGA) before the data is read by the CPU.

Safety Review

From V4.3 IDE the previous I^{2}T blocked is renamed as Safety Review. These settings in the Power PMAC limit current and voltage outputs in order to prevent damaging motors and amplifiers. The

Position limits section on this view is for the Servo Safety which is for configuring following error limits. Software position limits are read-only as without Home reference the software limits won't work correctly. The user can reset the Software Limits to default settings if they are different.

The Safety Review screen appears as follows:



The information icon  will display additional information about the parameter or the how the value is calculated as displayed in the above image.
Under the I2T Information section, the Safety Input section allows the user to edit the values which are prepopulated from the Amplifier and Motor database. Most users will never need to change from these prepopulated values. An information icon for each parameter will show additional information.

The next sections is for "Calculated Values", which contains three fields:

- Continuous Servo Output: This is the calculated continuous output limit from the servo loop in units of a 16-bit DAC. It is calculated based on the limits entered for Continuous Current, Instantaneous Current and Maximum Time Allowed, under the "Data Input" area of this view. When "Accept" is clicked this value will be written to **Motor[x].I2TSet**.
- Integrated Servo Output Limit: This is the maximum output from the servo loop's integrator based on the current limits entered. When "Accept" is clicked this value will be written to **Motor[x].I2TTrip**.
- Maximum Servo Output Limit: This is the maximum value which the servo loop can output. When "Accept" is clicked this value will be written to **Motor[x].MaxDac**.

Press the "Accept" button to accept the settings and move on to the next topology block.

Test and Set Block

Note

Please make sure that it is safe to Test and set the motor System Setup - Test and Set Topology block.
It is recommended external Emergency Stop switch connected that will kill the amplifier power in case of motor runaway or loss of communication.

This block performs a series of tests to ensure that the motor is working correctly. The tests which are run depend upon which kind of motor is being used.

The two selections available here are;
"Auto" - to run the predefined tests and configure the motor.
"Manual" - to manually specify parameters for each test and execute them sequentially.

In the manual screen each step in the testing process is listed with a Step Number, Description, Progress, and Result as shown below:

Step No.	Description	Progress	Result
1	Detect current sensor direction	100%	Pass
2	Measure current sensor bias value		
3	Voltage six step test	100%	Pass
4	Tune current loop	100%	Pass
5	Current six step test	100%	Pass
6	Open loop test		
7	Phase reference search		

"Progress" shows how far along the test has progressed.
"Result" will state whether the test passed ("Pass") or failed ("Fail"). The tests listed here depend on whether a Brush motor or a Brushless motor are being used.

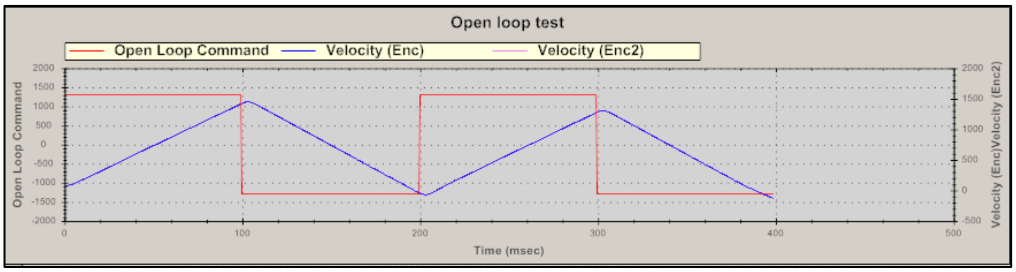
Brush Motors
If using a brush motor this window will run three tests:

Open Loop Test
This test issues an open loop command to the motor outputting the voltage to it without closing the servo loop. The purpose of this test is to ensure that a positive output command produces positive motion on the motor and that a negative output command produces negative motion. There are four parameters that can be adjusted in the Open Loop Test when using it in "Manual" mode:

Step No.	Parameters	Value
1*	MotorNumber	1
1	Magnitude (%)	20
1	Duration (msec)	100
1	Iterations	2

- “MotorNumber” selects which motor will execute the test.
- “Magnitude (%)” selects what percentage of the total output magnitude permitted by **Motor[x].MaxDac** to output to the motor for the test.
- “Duration (msec)” specifies how long to output voltage to the motor during the test.
- “Iterations” specifies how many times to output voltage to the motor. Each iteration consists of applying the magnitude of output specified in “Magnitude (%)” in the positive direction, and then once again in the negative direction.

A correct Open Loop Test should appear as follows where a positive output command produces positive encoder motion and a negative command produces negative encoder motion:



If the motor’s motion is the inverse of this i.e. a positive command produces negative motion and a negative command produces positive motion, then try changing the direction of the encoder decode structure. This structure is **Gate1[i].Chan[j].EncCtrl** for Gate1-Style Axis Interfaces and **Gate3[i].Chan[j].EncCtrl** for Gate3-Style Axis Interfaces. For Quadrature Encoders, to change the direction of the encoder decode using these structures change the structure’s value to 7 if it was 3 or to 3 if it was 7. Swapping the two leads of the motor can also be tried.

Measure DAC Bias Value

This test will output a zero-voltage command to see whether the motor moves. If it moves there is a bias on the DACs. It will then vary the DAC voltage until the motor stops moving in order to calculate this offset and then write it to **Motor[x].IaBias**.

There are two parameters that can be adjusted when executing this test manually:

Step No.	Parameters	Value
2*	MotorNumber	1
2	Iterations	2

- “MotorNumber” indicates which motor to perform the test.
- “Iterations” indicates how many times the window should try varying the output to the motor in order to determine the DAC bias.

Brushless Motors

If using a brushless motor this window will run eight tests:

Detect Current Sensor Direction

This test determines the directional sense of the current sensors being used to measure the currents in the motor's phases, as specified by **Motor[x].PhaseOffset**.

The only parameter to specify when executing this test manually is the motor number (MotorNumber):

Step No.	Parameters	Value
1*	MotorNumber	3

Note

Setting the **Motor[x].PhaseOffset** parameter correctly is extremely important because failing to do so can cause the current loop to be a positive feedback loop thus potentially causing damage to the motor or amplifier.

Measure Current Sensor Bias Value

This test measures any offset present on the ADCs which read the values of the current flowing through the motor's phases A and B. It does this by commanding a zero output and observing the current flowing through the phases for a brief period. The bias values are then stored in **Motor[x].IaBias** for phase A and **Motor[x].IbBias** for phase B.

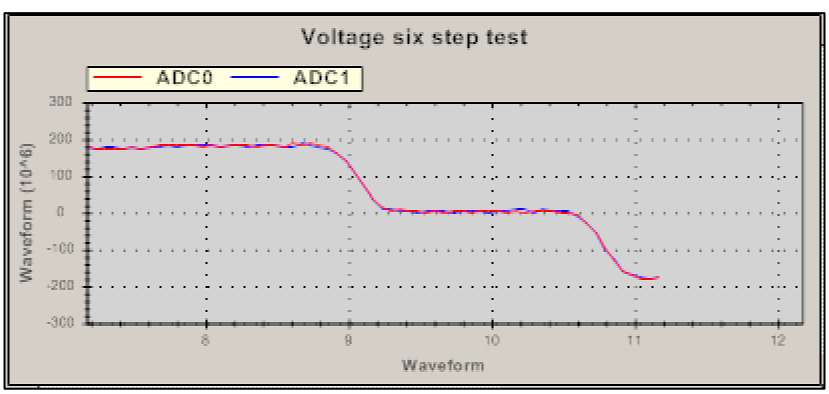
The only parameter to specify when executing this test manually is the motor number (MotorNumber below):

Step No.	Parameters	Value
2*	MotorNumber	3

Voltage Six Step Text

This test applies voltage across the motor's phases in order to commutate it one revolution. The test measures how many counts per electrical cycle in order to set **Motor[x].PwmSf**, **Motor[x].PhaseOffset**, and **Motor[x].PhasePosSf.**

During the test a plot will be displayed showing the ADC results for the current values on phases A (red) and B (blue) on the vertical axis moving with Time (horizontal axis):



Three parameters can be adjusted in this test:

Step No.	Parameters	Value
3*	MotorNumber	3
3	Magnitude (bits)	9421
3	Commutation Size (pPhaseEnc LSB)	2000

- "MotorNumber" indicates which motor to perform the test.
- "Magnitude" is the voltage to be applied to the motor in units of 16-bit DAC bits.
- "Commutation Size" is an input from the user; it specifies how many counts per commutation cycle. It is in units of the LSB of the register to which this motor's Motor[x].pPhaseEnc structure points.

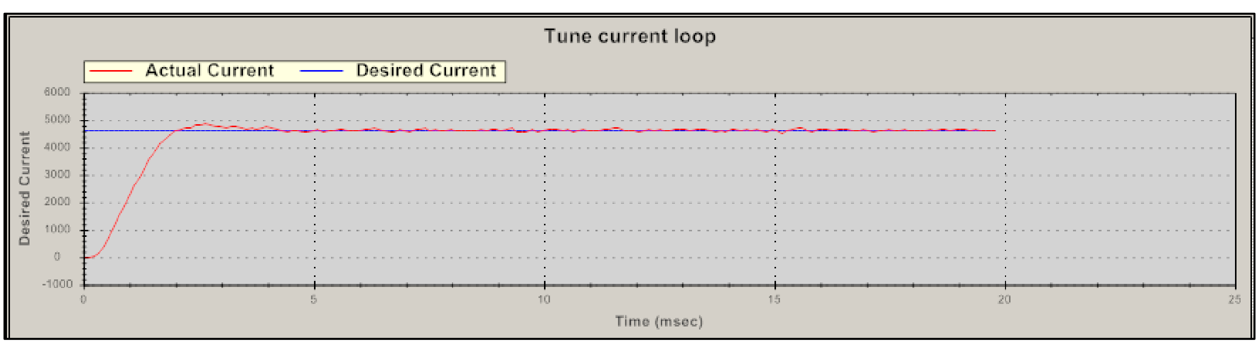
Tune Current Loop
This test will command current to the motor's phases and then calculate gains for the motor's current loop. The current loops gains are stored in the following structures: **Motor[x].IiGain**, **Motor[x].IpfGain**, and **Motor[x].IpbGain**.

The parameters available when executing the test manually are shown below:

Step No.	Parameters	Value
4*	MotorNumber	3
4	Magnitude (bits)	9421
4	Duration (msec)	20
4	Desired Bandwidth (Hz)	0

- "MotorNumber" indicates which motor to perform the test.
- "Magnitude" is the current to put through the motor's phases in units of 16-bit DAC bits.
- "Duration" is how long to apply current to the phases [msec].
- "Desired Bandwidth" is the amount of bandwidth, which was specified, for the current loop to have [Hz] was specified.

After the test tunes the current loop it will plot the current loop's response, which should look more or less like the image below, where the actual current (red) rises to the desired current (blue):



The desired current is on the left axis in units of 16-bit DAC bits and time is on the horizontal axis in units of milliseconds.

Often the automatic tuning is adequate but if interactive fine-tuning is required please refer to the the section labeled "Tuning the Servo Loop in the IDE" in the Power PMAC User's Manual.

Current Six Step Test

This test applies voltage across the motor's phases in order to commutate it one revolution. The test measures how many counts per electrical cycle the motor has in order to set **Motor[x].PhasePosSf.**

The parameters available when executing the test manually are shown below:

Step No.	Parameters	Value
5*	MotorNumber	3
5	Magnitude (bits)	3084.6438
5	Commutation Size (pPhaseEnc LSB)	2000

- "MotorNumber" indicates which motor to perform the test.
- "Magnitude" is the current to be applied to the motor in units of 16-bit DAC bits.
- "Commutation Size" is an input from the user; it specifies how many counts per commutation cycle. It is in units of the LSB of the register to which this motor's **Motor[x].pPhaseEnc** structure points.

Open Loop Test

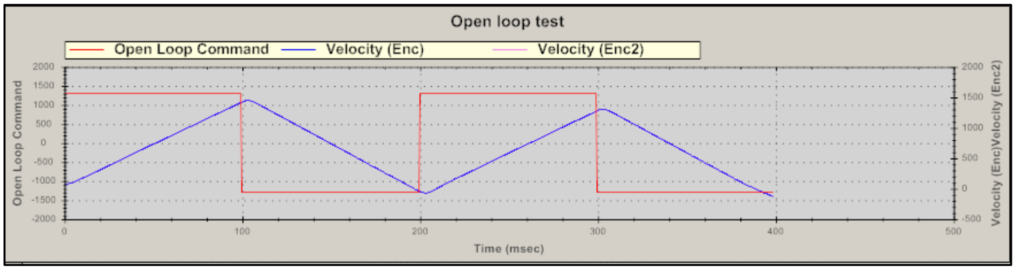
This test issues an open loop command to the motor outputting voltage to it without closing the servo loop. The purpose of this test is to ensure that a positive output command produces positive motion on the motor and that a negative output command produces negative motion.

There are four parameters that can be adjusted in the Open Loop Test when using "Manual" mode:

Step No.	Parameters	Value
1*	MotorNumber	1
1	Magnitude (%)	20
1	Duration (msec)	100
1	Iterations	2

- "MotorNumber" indicates which motor to perform the test.
- "Magnitude (%)" selects what percentage of the total output magnitude permitted by **Motor[x].MaxDac** to output to the motor for the test.
- "Duration (msec)" specifies how long to output voltage to the motor during the test.
- "Iterations" specifies how many times to output voltage to the motor. Each iteration consists of applying the magnitude of output specified in "Magnitude (%)" in the positive direction, and then once again in the negative direction.

A correct Open Loop Test should appear as follows where a positive output command produces positive encoder motion and a negative command produces negative encoder motion:



If the motor's motion is the inverse of this i.e. a positive command produces negative motion and a negative command produces positive motion, try changing the direction of the encoder decode structure and rephasing the motor (for commutated motors). This structure is **Gate1[i].Chan[j].EncCtrl** for Gate1-Style Axis Interfaces and **Gate3[i].Chan[j].EncCtrl** for Gate3-Style Axis Interfaces. For Quadrature Encoders, to change the direction of the encoder decode using these structures change the structure's value to 7 if it was 3, or to 3 if it was 7. Swapping the two leads of the motor can also be tried.

If the Open Loop Test's response is not inverted from the picture above, but is rather erratic, try rephasing the motor or retuning the current loop (see the Tuning section of this manual for more details on tuning).

Phase Reference Search
This test establishes a phase reference for the motor, i.e. it tries to align the rotor with a phase in order to maximize the motor's torque output.

There are four parameters that can be adjusted for this test:

Step No.	Parameters	Value
7*	MotorNumber	3
7	Phasing Method	1
7	Magnitude (bits)	0
7	Phase search time (msec)	0

- "MotorNumber" indicates which motor to perform the test.
- "Phasing Method" determines which automatic phasing routine to use:
    - Set to 1 to use Stepper Method
    - Set to 2 to use the Two-Guess Method
- "Magnitude" is the current to apply to the motor when phasing [16-bit DAC bits].
- "Phase Search Time" is how long to apply current to the motor before setting the phase position to 0

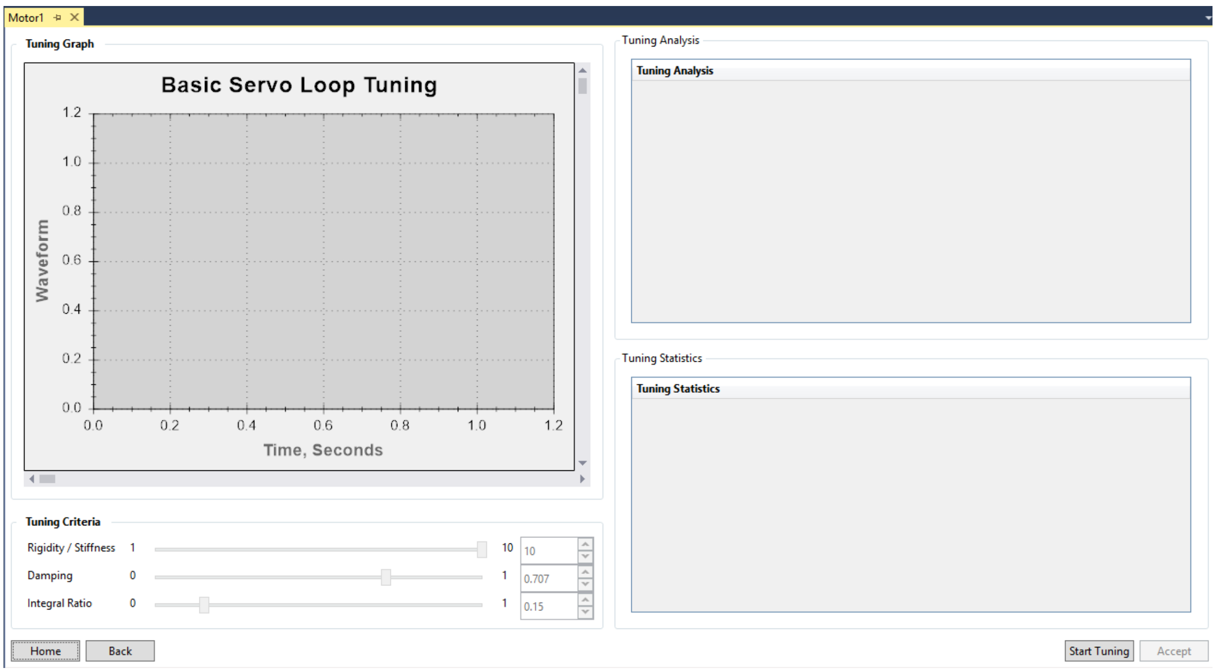
Basic Tuning Block

The major difference between IDE V2.x or V3.x and V4.x is the Servo loop tuning from Test and Set is removed and replaced by the Basic Tuning block. The concept of the basic tuning is that for new and basic users the tuning algorithm should achieve the performance needed therefore not requiring the use of the Advance tuning.

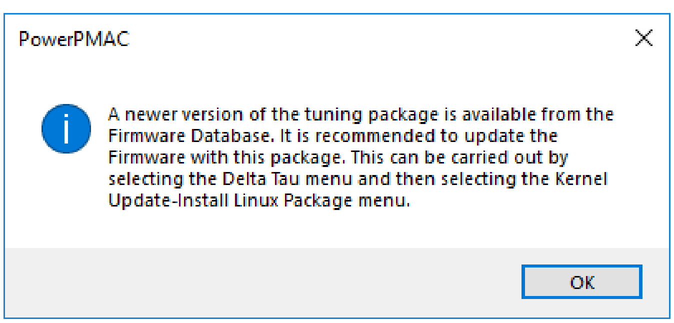
This is a simple one button tune function. Once the Basic Tuning is complete the bandwidth can be changed, and the test can be re-run to recalculate the gains. This can be used to optimize the tuning by utilizing our intelligent tuning algorithm.

Advance tuning is for the expert User who possess the correct knowledge of controls theory.

When Basic Tuning is selected the screen below will be displayed.

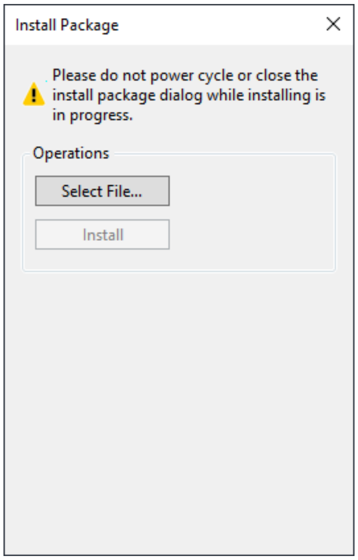


If the User is using the FW 2.5.1.7 without a new Tuning package, then a warning will be shown as below…



It is not mandatory to upgrade the tuning package, but the User does not then they will not get the benefit of improvements in the tuning and setup algorithms.

If the User wants to upgrade the tuning package, they can download this from the Delta Tau Firmware location and use Install package dialog from the from the Delta Tau menu and select File - Install Linux Package like this…

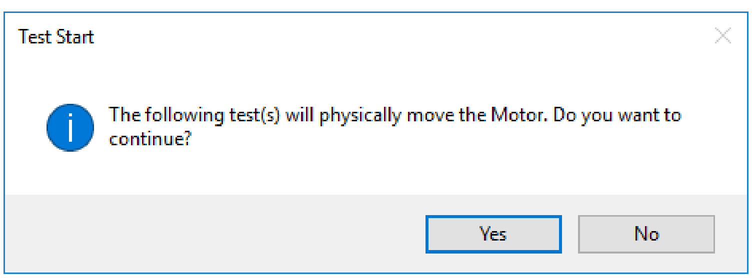


Once the package is updated then the User can use the Basic tuning block to tune the Torque or velocity mode and on success proceed to Commissioning and Motor Jog Block to test the motor.

Note

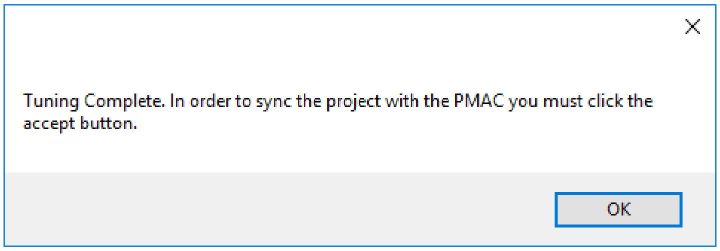
The User only needs to install the Tuning package once. For any following set up’s the Warning message will not be displayed.

Press “Start Tuning” and a safety warning will be displayed before the tuning starts.

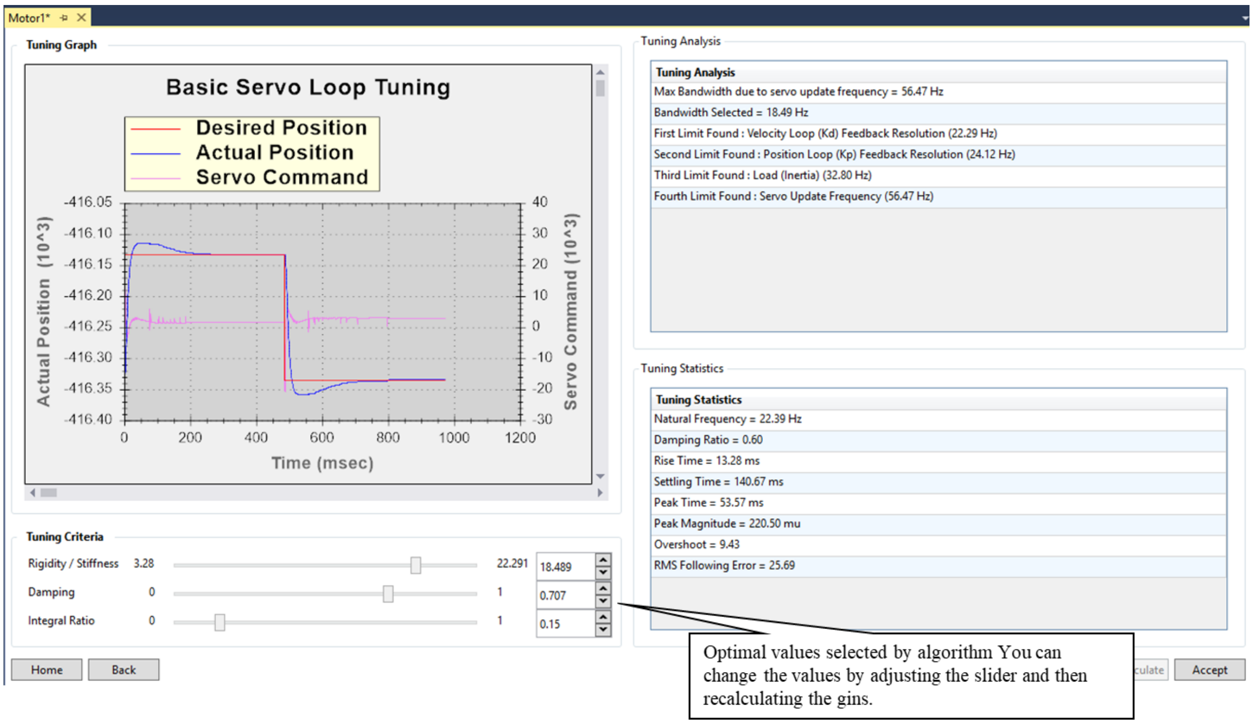


Press Yes to start the Tuning.

On completion synchronize the results on the Power PMAC and project by pressing Accept.



On successful Tuning the screen will be displayed as shown below:



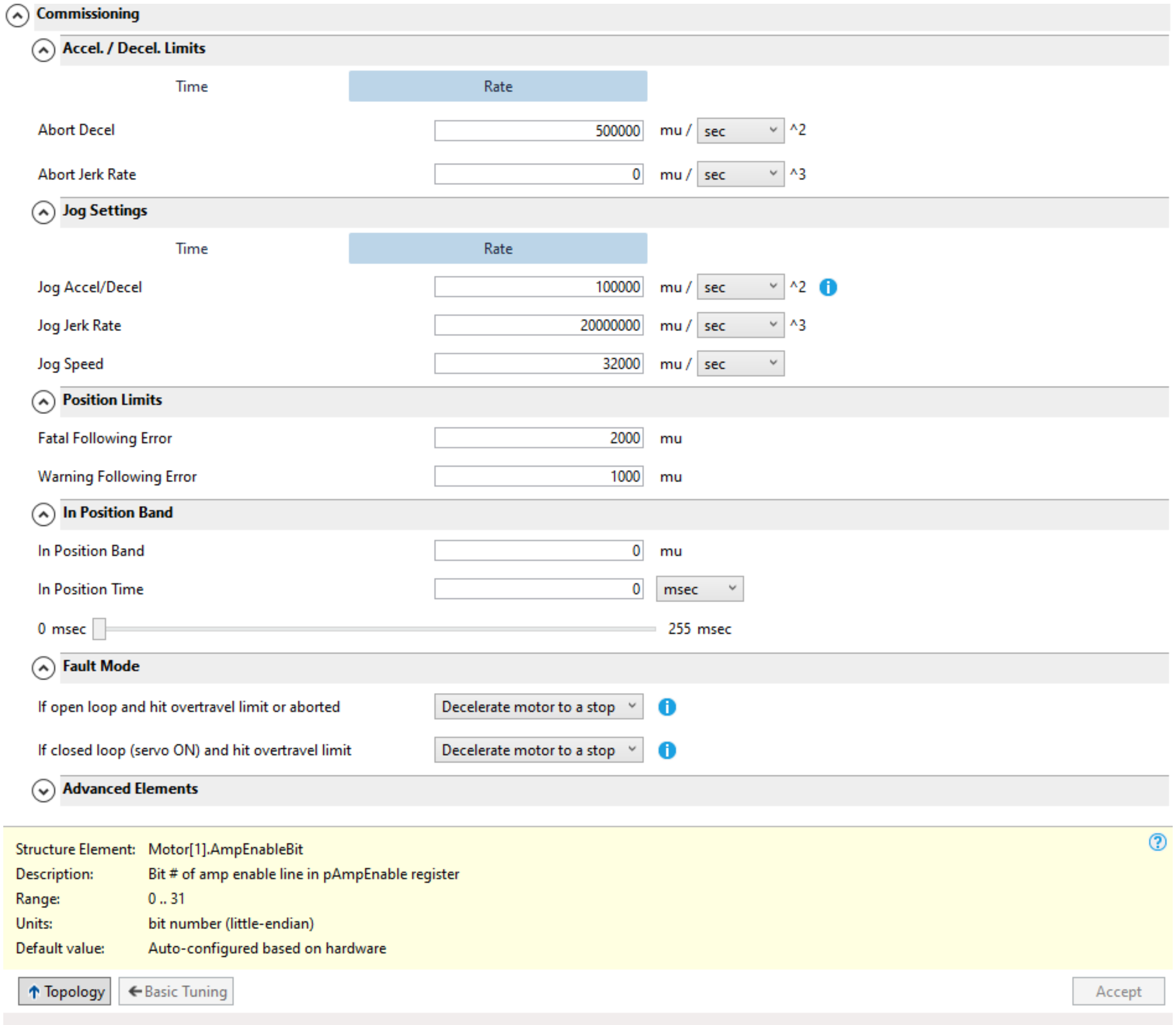
On Re-calculate it will make another move to accept the changes.

Commissioning Block

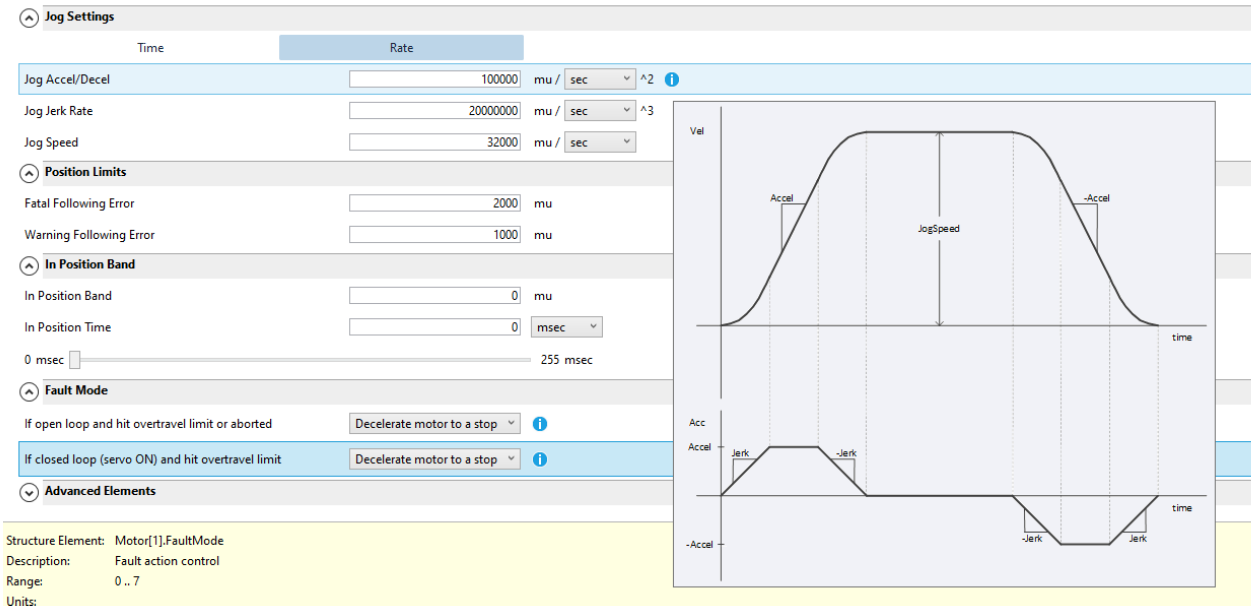
Please make sure that it is safe to commission motor parameters.
Commissioning blocks sets Power PMAC Motor structure elements.

Note

Like Coordinate System, the Commissioning block is a collection categorized Motor elements that are commonly used.
In the commissioning page the user can set accel/decel limits, jog settings, position limits, in position band, fault mode and advanced settings. The User Units, to the right of the data entry, will display whatever has been set in the User Units window. For example, if, in the User Units block on the topology, inch is set then all the User Unit fields will show inch. Each row of this page is selectable and a brief description of the selected will be displayed at the bottom of the page.



This icon in front of parameter means there is additional information available. On clicking the Icon, a graphical image will be displayed to give a better understanding of the parameter.

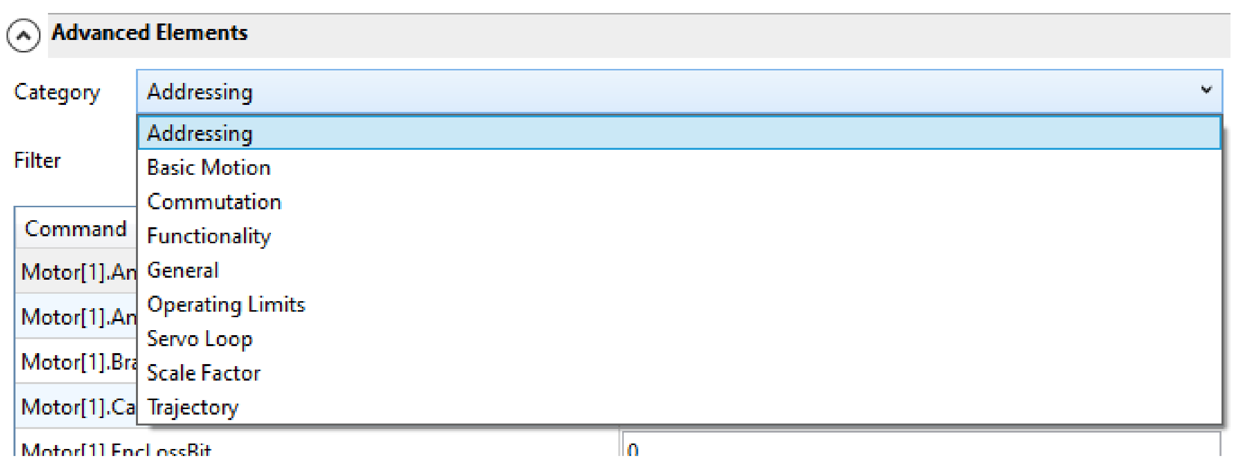


The Advance view can be expanded to view the entire motor structure elements.

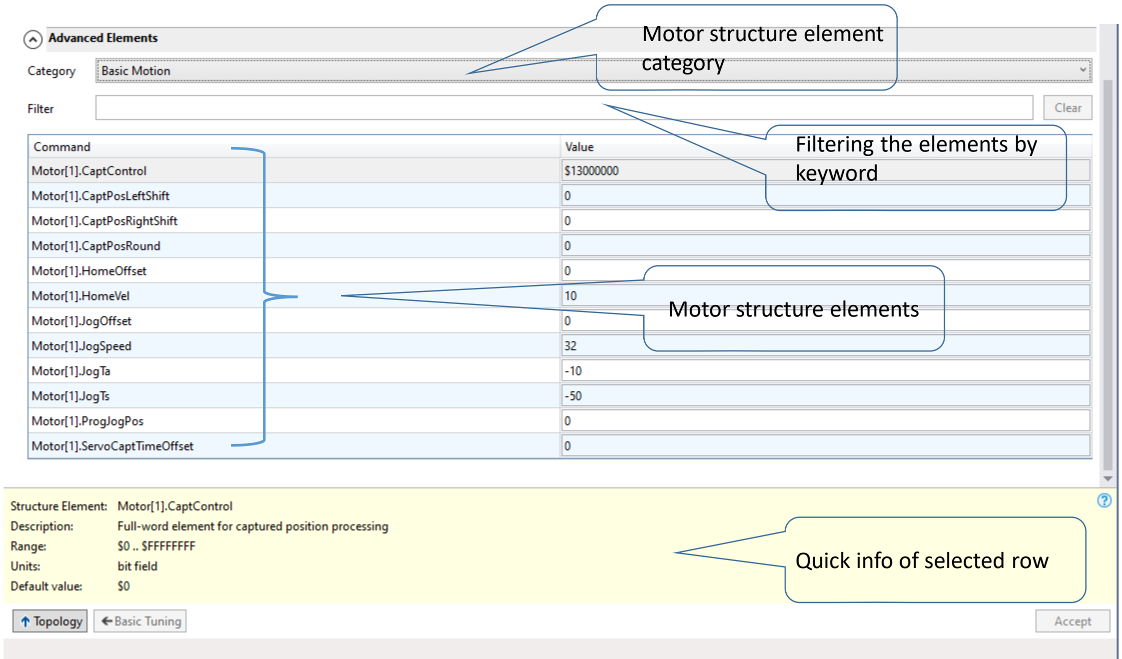
Note

The Advanced System Elements View can be accessed from the Commissioning View. This shows the categorised motor setup elements. This view is for Users who do not want to use the topology approach and have the expertise to setup the Power PMAC. As with the topology approach all the values are written into their relevant files and used, when built, to generate the systemsetup.cfg.

The advanced view will allow the user to select from one of the motor element categories.

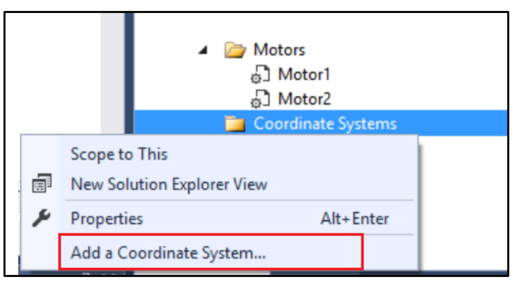


By selecting a category the table will be populated by the motor structure elements specific to that category. Also the populated list can be filtered by typing any keyword in the filter section. The following is the advanced setting view.

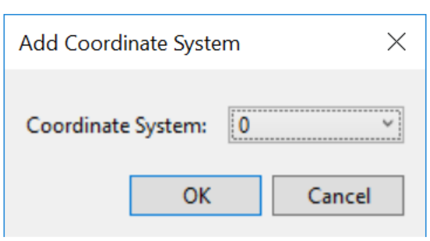


Coordinate Systems-Context menu

A Coordinate System can be added to the project by right clicking the Coordinate Systems node in the Solution Explorer, as shown below:



When “Add a Coordinate System” is selected a dialog box will be displayed and the number of the Coordinate System can be selected, as shown below:

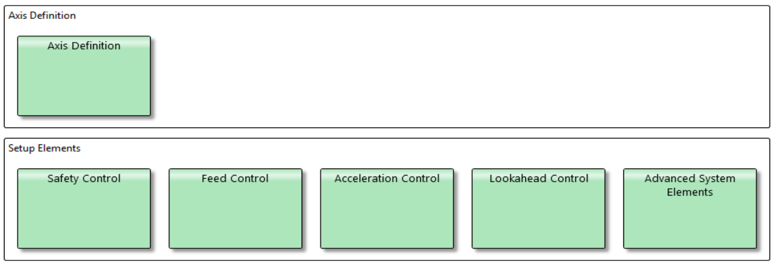


The dropdown box will display default coordinate system number of 0. The range available is from 0 to Sys.MaxCoords.

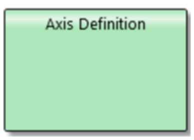
Note

When the Motor is added to a project the Coordinate System structure elements are saved to a file. Any Coordinate System structure element changes within the project domain will be automatically updated and maintianed within the file. When the build is performed the Coordinate System file will be used to generate the systemsetup.cfg file. No backup is needed for the Coordinate System parameters as long as the changes are being made in the project system.

On clicking the OK the Coordinate System view will be displayed, as shown below.



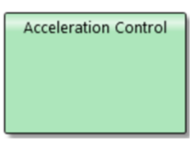
These settings are for coordinate system elements.



You can create axis definition that will be used in the motion script. The dropdown list only show motors that are available and not assigned to any other coordinate system.
Example : #1->1000X OR #1->X

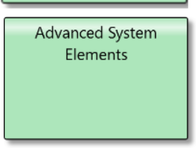
You can set safety limits or action in case of Abort is issued in the middle of the motion program.

You can set motion program feed control settings. These setting are..
Feedrate time units, Feedrate slew rate etc.



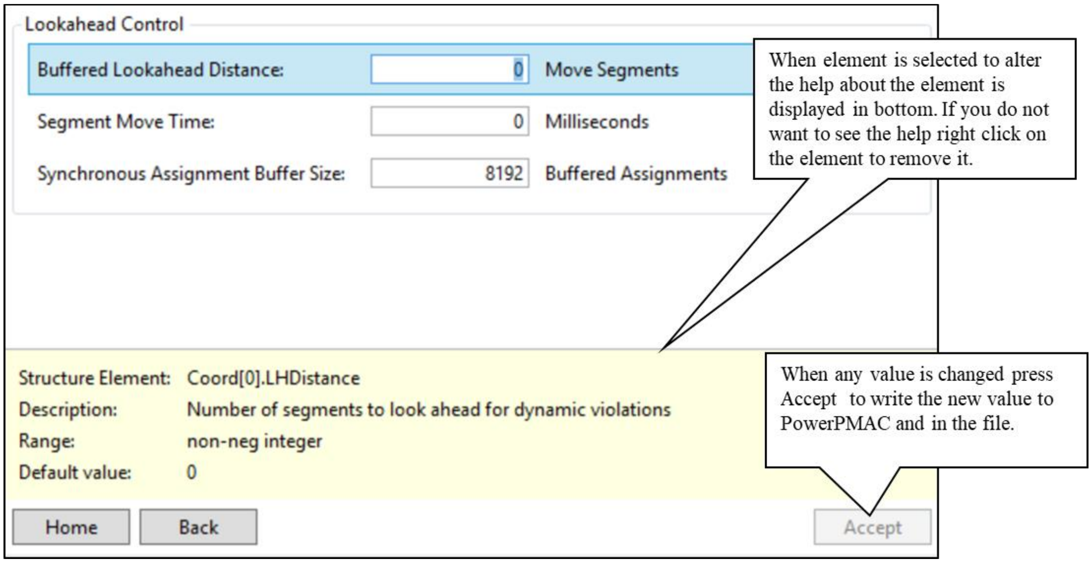
You can set acceleration limits in blending moves. These settings are…
Acceleration time for blended moves, deceleration time for blended move etc.

You can set Lookahead mode related settings.
The settings are..
Lookahead distance, segmentation move time, etc.



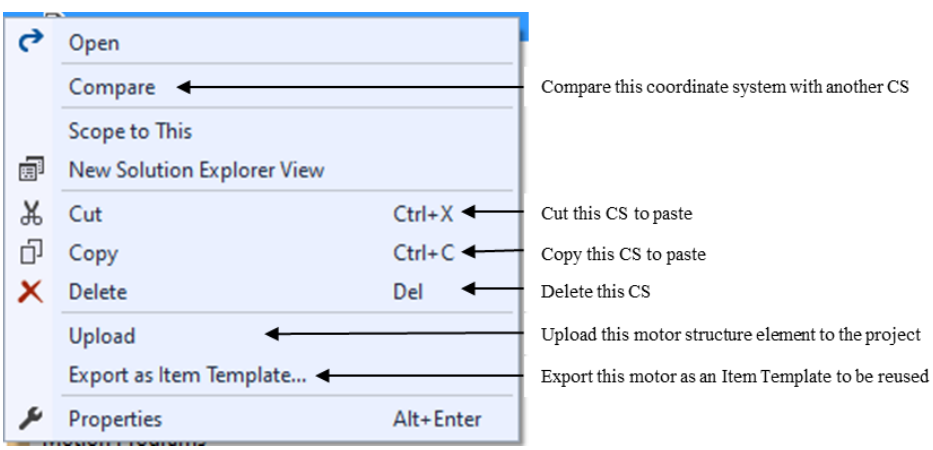
User can access the Advanced System Elements. These are for Expert PMAC Users

On clicking any of the Coordinate System blocks a common view will be opened in the editor. The layout and navigation are the same across the Coordinate System and Motor commissioning blocks.
For example:



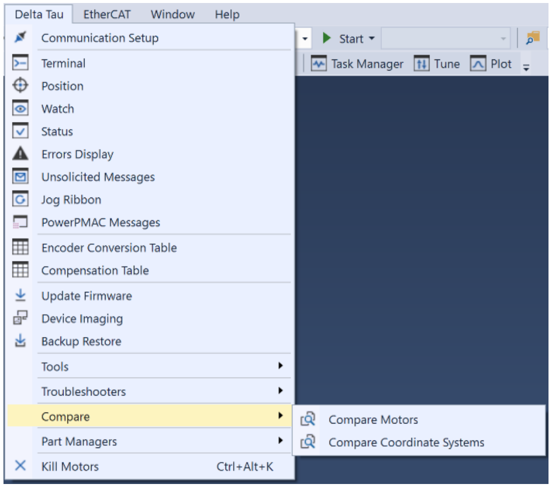
CoordinateSystem-Context menu

This menu is available when any type of motor is added and displayed under Motors node. Right-clicking on a motor node will open up a context menu containing various useful operations as shown below … For convinience a word CS = Coordinate System will be .

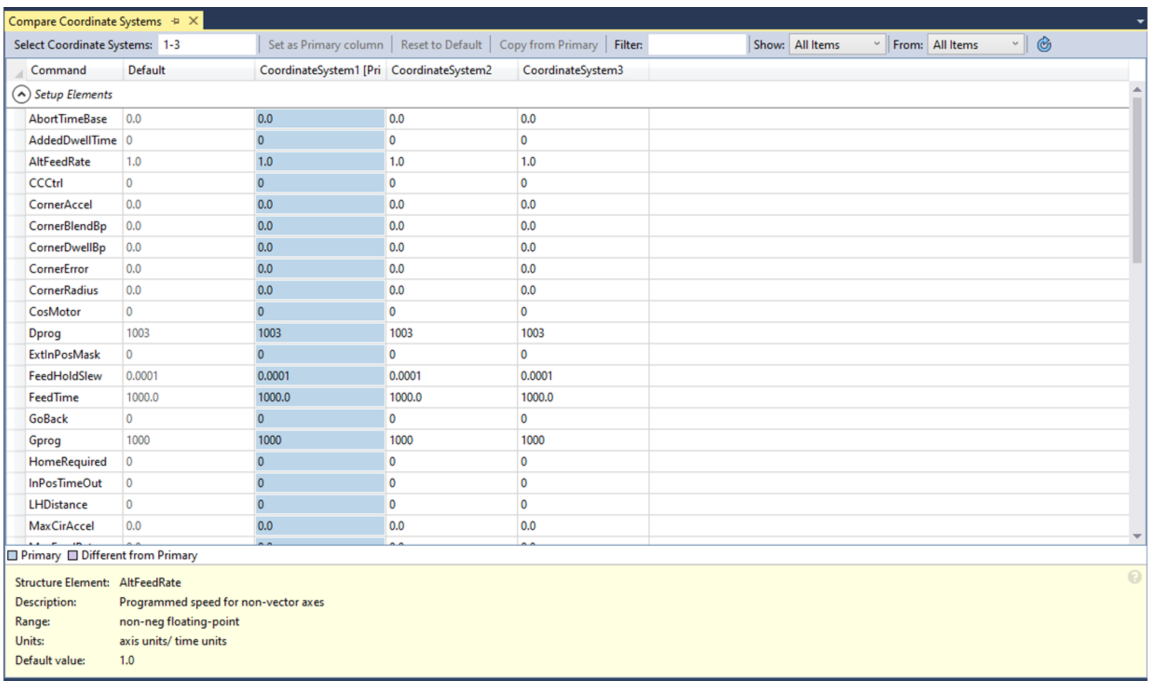


Compare

The compare feature is available for coordinate systems. It allows the comparison of coordinate system elements. The structure elements are categorized. A maximum of nine coordinate systems can be compared at a time. The Compare function is available from the Delta Tau menu or by right clicking on the CoordinateSystems in the Solution Explorer.
The following dialog shows the Compare feature being accessed from the Delta Tau menu.



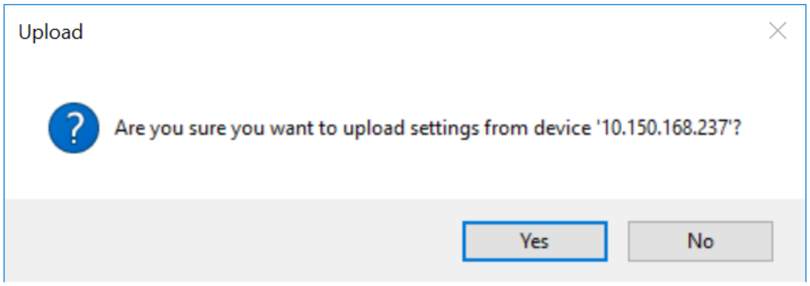
The default view shows all the coordinate structure elements.



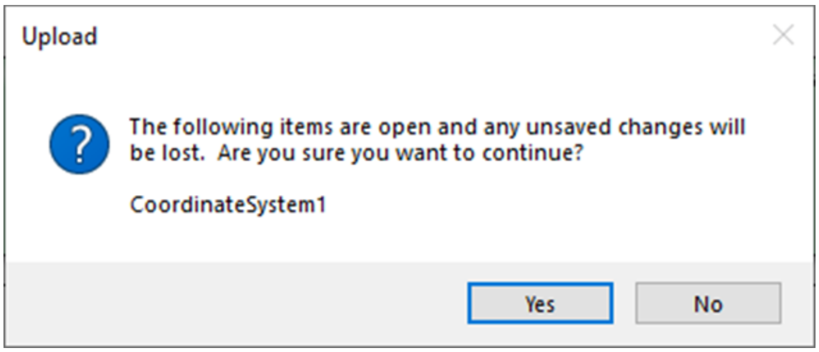
Upload

Upload coordinate system gives the ability to upload the currently saved CS structure elements from the Power PMAC to the project.

On selecting this option, a confirmation dialog will be displayed as shown below:



On Clicking Yes, if the CS View Editor is open in the IDE, a confirmation dialog will be displayed confirming that any unsaved data will be lost by performing the upload.



On a successful upload the CS in the project will be synchronized with the Power PMAC motor structure elements.

Note

This option is useful if the CS structure element has been changed outside of project domain such as in the Terminal window.

Export as Item Template

The CS can be exported or imported as item templates. All the CS settings will be exported during this process.

The typical use of the CS template is to setup a complete CS, and then share this with another user.

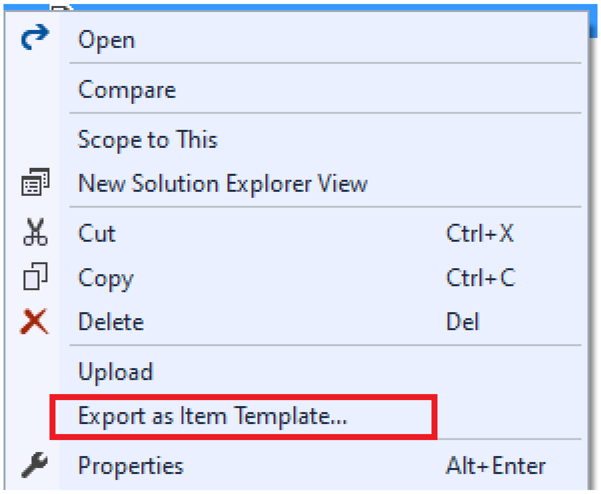
This User can then Import the CS, using Import item template option, and use it in their project saving the time of having to create the CS settings from new.

Using this option the User can:

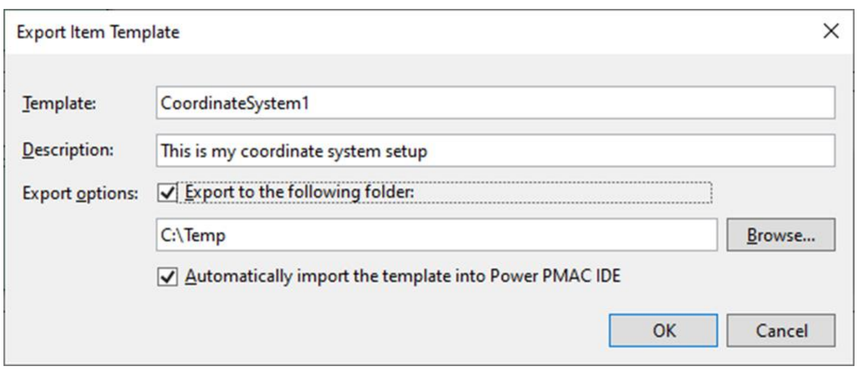
- Export a CS in order to use it in another project
- Import a CS to reuse in their own project
- Create a new CS/ from an Imported CS/ Item Template
- Choose whether or not to automatically Import an Item Template into Power PMAC IDE project at the point that it is exported

- Delete imported Custom Item templates

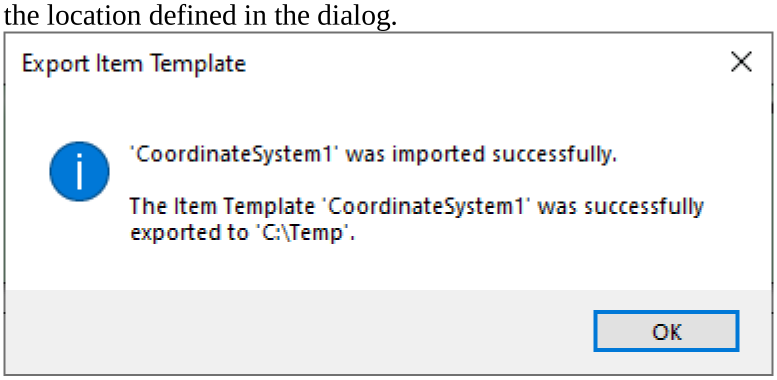
To export a CS settings as a Template the user will need to right click on CoordinateSystem node under Coordinate System and choose the “Export as Item template” option as shown below:



On selecting the option, a new export item template dialog will be displayed, as shown below:

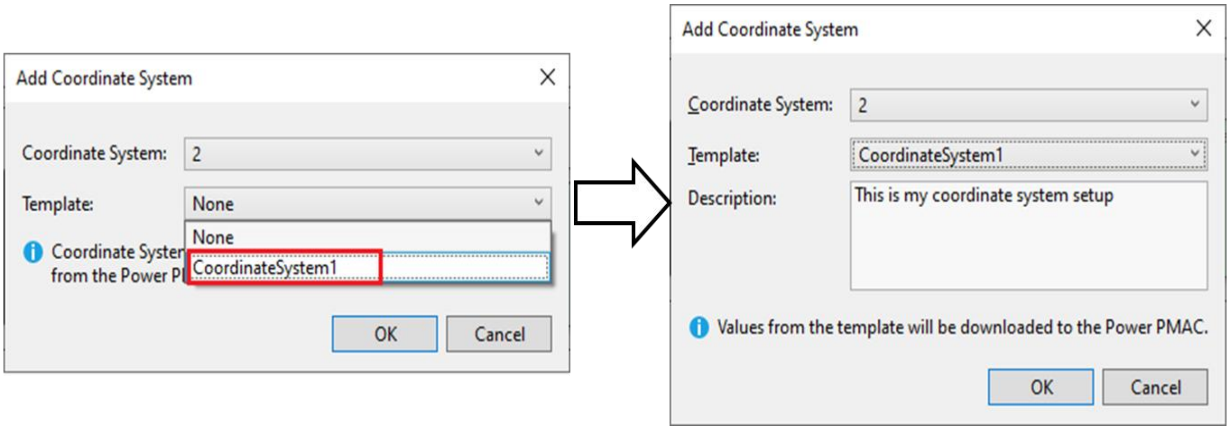


On selecting Ok the acknowledge message will be displayed and template will be exported and stored into the location defined in the dialog.



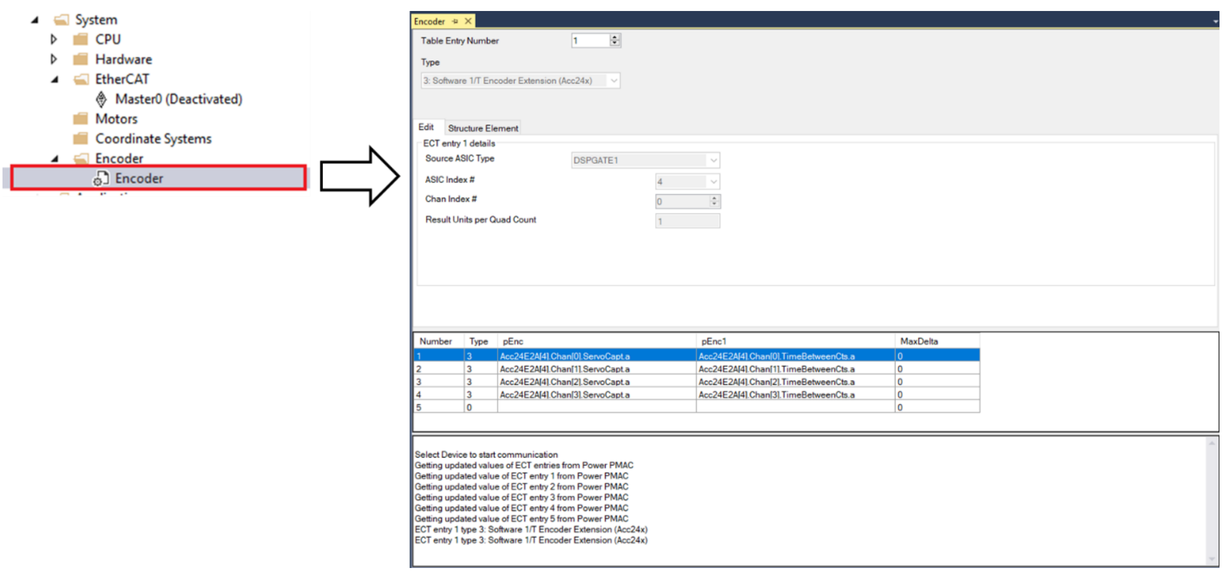
The User has ability to store the template to any folder by ticking the “Export to the following folder” checkbox.

By default, the template will be imported to be used in the current instance of the IDE. Un-ticking this check box will not import the template into the current instance of the IDE. In this case use Template Manager from File Menu to import Coordinate system template.
By Default the template will be available as shown below when Add Coordinate System is selected….



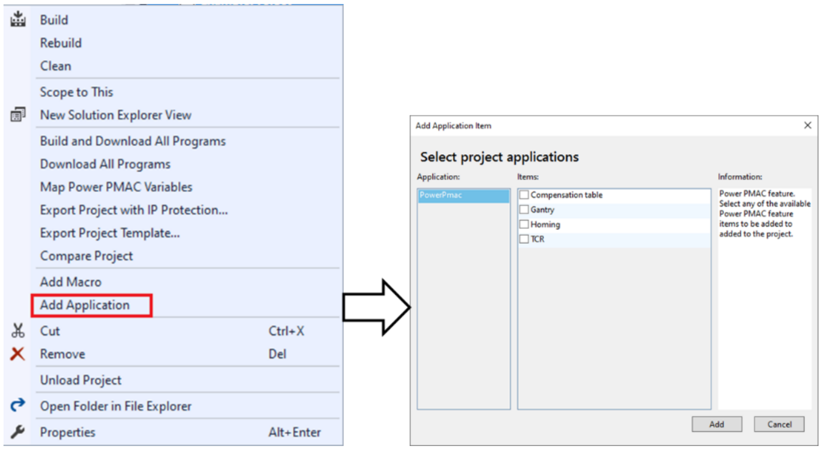
Encoder

The Encoder tables' settings are stored in the Encoder file. The Hardware Interface block on the Motor Topology writes to the Encoder file on Accepting the data. These settings are then used in creating the systemsetup.cfg on build. On double clicking the Encoder viewer. This is read-only and will allow user to verify encoder table setting for configured motor. The viewer will look like this…



Application

This is the new folder added in the Power PMAC project. The application folder can be added to existing project by right clicking the project and selecting Add Application context menu. The work flow is shown below. The Add Application is dynamic. As soon as the folder is added to the Project the Add Application menu will disappear from context menu. If user only adds one application then the context menu will dynamically change to Add Application Item to add other applications from the list.



Another way to get Application folder in the project is using Project Wizard to create project. It is explained under File –NEW-Project/Project Wizard

There are currently four Power PMAC common application are supported

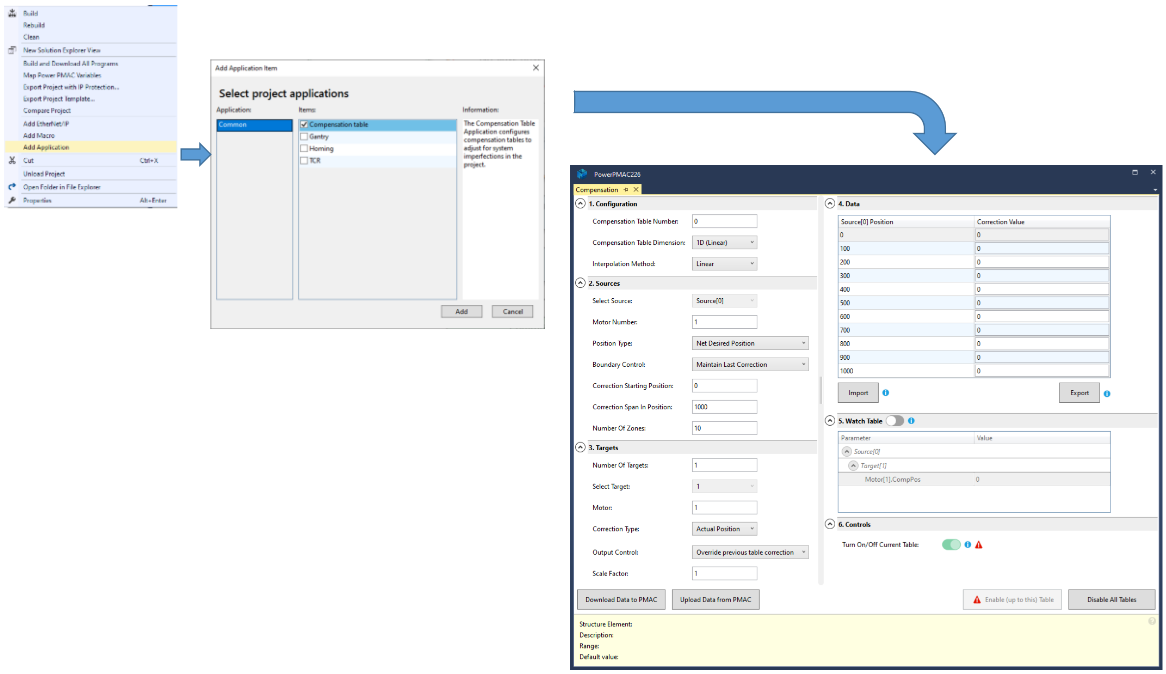
1. Compensation Table
2. Gantry
3. Homing
4. TCR (Requires CK3WGCxxxx hardware)

Application Expectation:
Motor is fully configured using System Setup (Recommended) and can freely jog

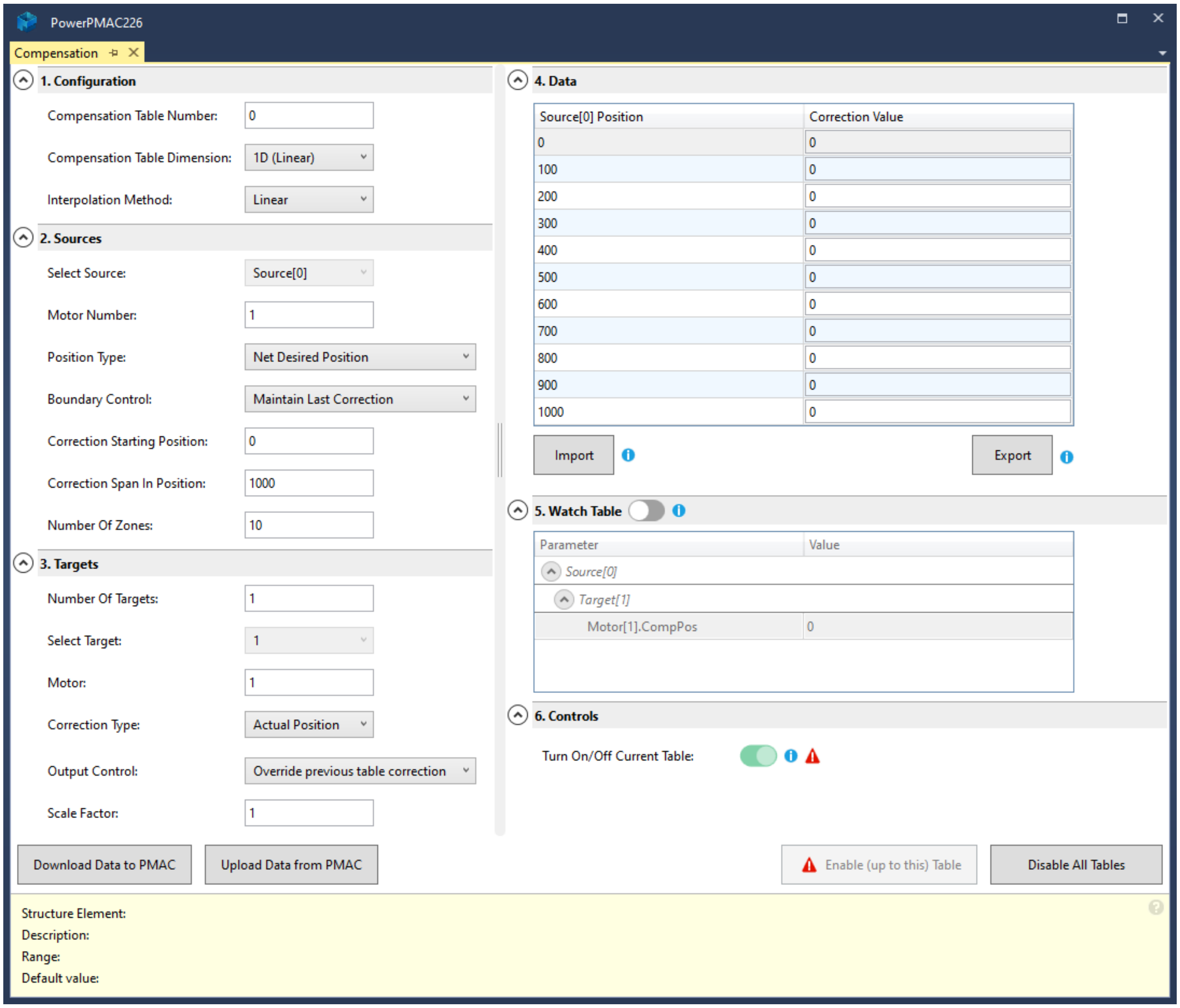
Note

Compensation Table

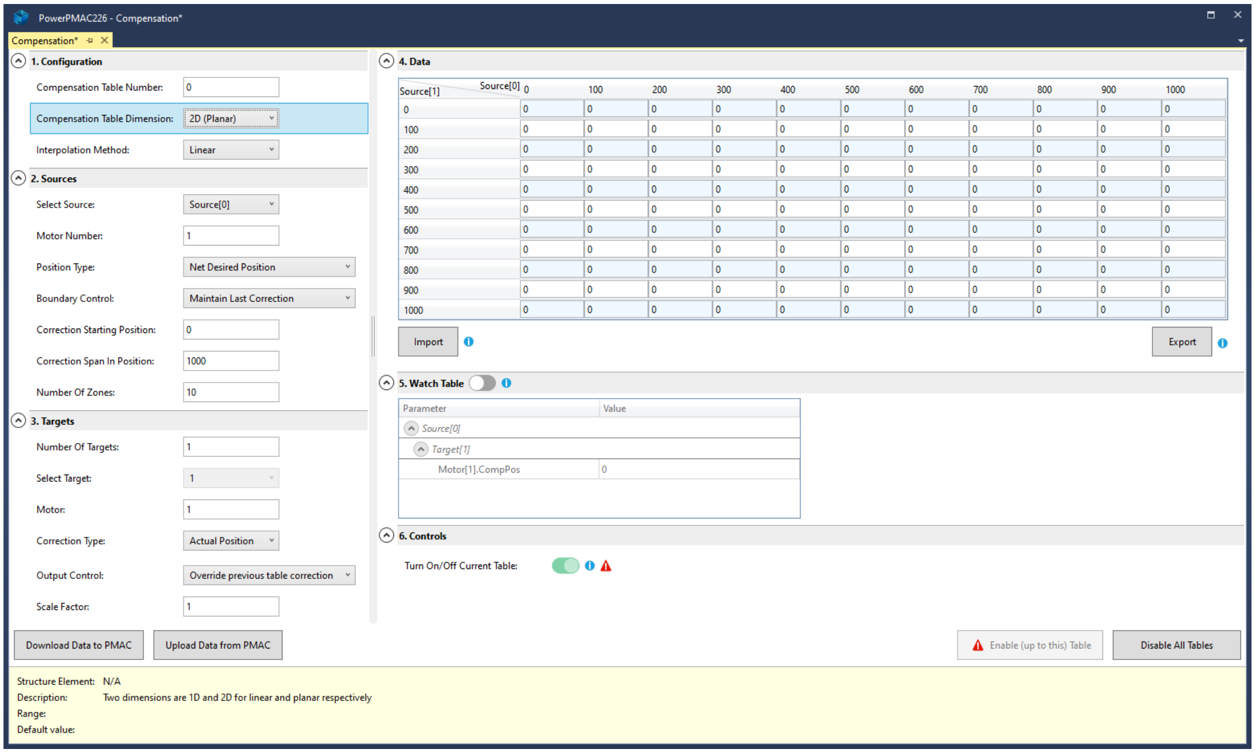
To add this App use the Add Application or use Project wizard. Typical workflow shown below. This workflow shows that the compensation table application added using Add Application item context menu.



As shown above the compensation added under Application Node, marked with Red square.
As it is part of the project it is integrated with project so all the setup parameters are stored with the project.
The Power PMAC Comp Table Setup window is used for setting up Compensation Tables in Power PMAC (i.e. the members of the **CompTable[x]** structure). Power PMAC Compensation Tables can be configured for 1D, 2D. The main window for a 1D Compensation Table appears as follows:


PowerPMAC226
Compensation
1. Configuration
Compensation Table Number: 0
Compensation Table Dimension: 1D (Linear)
Interpolation Method: Linear
2. Sources
Select Source: Source[0]
Motor Number: 1
Position Type: Net Desired Position
Boundary Control: Maintain Last Correction
Correction Starting Position: 0
Correction Span In Position: 1000
Number Of Zones: 10
3. Targets
Number Of Targets: 1
Select Target: 1
Motor: 1
Correction Type: Actual Position
Output Control: Override previous table correction
Scale Factor: 1
4. Data
Source[0] Position
Correction Value
0
100
200
300
400
500
600
700
800
900
1000
Import
Export
5. Watch Table
Parameter
Value
Source[0]
Target[1]
Motor[1].CompPos
6. Controls
Turn On/Off Current Table:
Download Data to PMAC
Upload Data from PMAC
Enable (up to this) Table
Disable All Tables
Structure Element:
Description:
Range:
Default value:

A 2D Compensation Table appears as follows:



The Comp Table Setup grid contains three sections:

A. Configurable items in the property grid are categorized into six sections:
   1. The “Configuration” section includes the following items:
      a) Comp Table Number: runs from 0 to 255.
      b) Comp Table Dimension: allows the user to select the dimension of the Compensation Table from three choices: 1D-Linear; 2D-Planar.
      c) Interpolation method: Linear or cubic.
      With linear (first-order) interpolation, the correction in the dimension of the source is calculated as a linear fit between the points on either side of the present position. It can have sudden changes in slope as it passes a point in the table, which may result in noticeably rough motion.

      With cubic (third-order) interpolation, the correction in the dimension of the source is calculated as a cubic fit using two points on either side of the present position. The slope of the correction is always continuous, yielding smooth motion. This interpolation takes about twice the calculation time of first-order interpolation.

   2. The “Sources” section includes the following items:
      a) Select Source: Depends upon the dimension of the Compensation Table: 1 for Linear, 2 for Planar. User has the option to select the motor number for each source motor. For 1D, only Source[0] is displayed; 2D, Source[0] and Source[1];

b) Motor Number: Specify the motor number for each target.
c) Position Type: The position type can be "Net Desired Position" or "Uncorrected actual position"
d) Boundary Control: there are three possible choices for the boundary control.
Roll Over: Rolls over at table boundary
Maintain Last Correction: Maintains last correction past the table boundary
Mirror: Mirrors the table correction at the boundary
e) Correction Starting Position: sets the correction starting position of the compensation table.
f) Correction Span In Position: Sets the total span of the compensation table.
g) Number Of Zones: Sets the total number of zones in the compensation table.

3. "Targets" section includes the following items:
    a) "Number of Targets": Power PMAC Compensation Tables support up to 8 target addresses.
    b) "Select Target": Select target is populated based on the number of targets and one target can be selected.
    c) "Motor": Specify the target motor number for each target.
    d) Correction Type: There are 6 different types of corrections. Actual Position, Actual Velocity, Backlash, Desired Position and Torque.
    e) Output control: Supports two options. Override previous table correction and Add To Previous Table Correction.
    f) "Scale factor": the target scale factor.
    g) "First Data Point" is the table's starting point in motor units in the given dimension.
    h) Total Span specifies the length of the compensation table in motor units.
    i) "Number of Zones" is equal to the number of sections between the First Data Point and the Last Data Point, which can be computed as (First Data Point + Total Span).

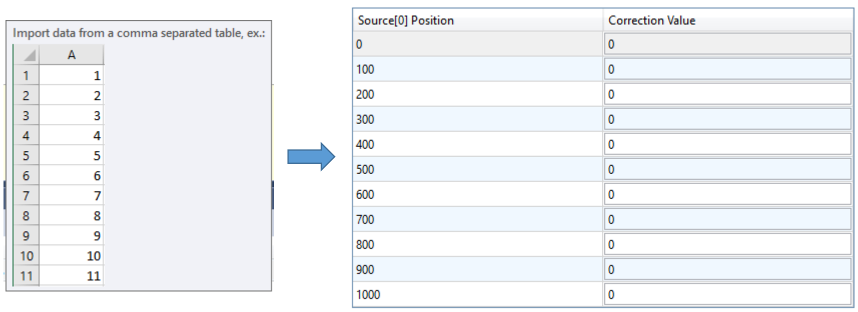
4. Data Values for each Dimension:

Based on the Data Properties grid items, enter data values at equally spaced points between first and the last point.
For 1D Tables, a 1D list of points is generated. Each correction value entered in the list corresponds to the **CompTable[*n*].Data[*i*]** structure, where ***n*** specifies the Compensation Table number and ***i*** specifies the 1st dimension point index.
For 2D Tables, a 2D Data Grid is generated. Each correction value entered in the 2D grid corresponds to the **CompTable[*n*].Data[*j*][*i*]**, where ***n*** specifies the Compensation Table number, ***j*** specifies the 2nd dimension point index and ***i*** specifies the 1st dimension point index.

a) "Import": The import allows the user to import a comma separated table (.csv) file. Import button prompts the user to open a data file corresponding to a previously configured Compensation Table. If the dimension in the property grid and data values (given by a comma separated file) match, then the values are appropriately added in the data grid.

Following is typical csv file for 1D and 2D…

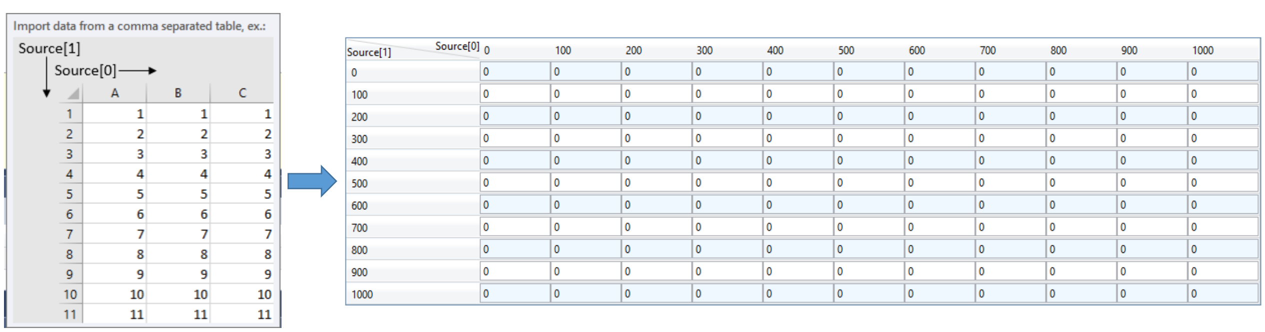
1D csv file



	A
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11

Source[0] Position	Correction Value
0	0
100	0
200	0
300	0
400	0
500	0
600	0
700	0
800	0
900	0
1000	0

2D csv file



	A	B	C
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10
11	11	11	11

Source[1] \ Source[0]	0	100	200	300	400	500	600	700	800	900	1000
0	0	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0	0	0	0
300	0	0	0	0	0	0	0	0	0	0	0
400	0	0	0	0	0	0	0	0	0	0	0
500	0	0	0	0	0	0	0	0	0	0	0
600	0	0	0	0	0	0	0	0	0	0	0
700	0	0	0	0	0	0	0	0	0	0	0
800	0	0	0	0	0	0	0	0	0	0	0
900	0	0	0	0	0	0	0	0	0	0	0
1000	0	0	0	0	0	0	0	0	0	0	0

b) “Export”: The export allows the user to export the compensation table into a comma separated (.csv) file.

5. Watch Table:

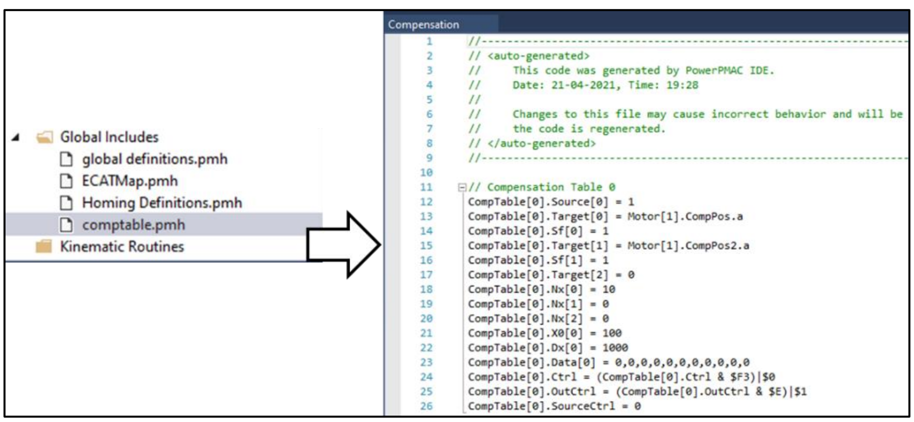
   This section enables the user to monitor some source and target parameters. To enable the watch table the toggle button can be toggled on or off. The data should be downloaded to PMAC and the table must be enabled before watching the parameters.

6. Controls:

   This section allows the user to turn the current table on or off. The source motor must have completed a homing move before a table can be enabled.

7. At the bottom of the screen, four buttons are provided for the user’s convenience to achieve the following tasks:

   a) **Download Data to PMAC** allows the user to download complete Compensation Table configurations and data values to Power PMAC. This should be done after the Table is crafted using this tool. On Download it also creates compatable.pmh file under Global Includes. It stores the table so on build and download it will get loaded to Power PMAC after reset. It shows like this…



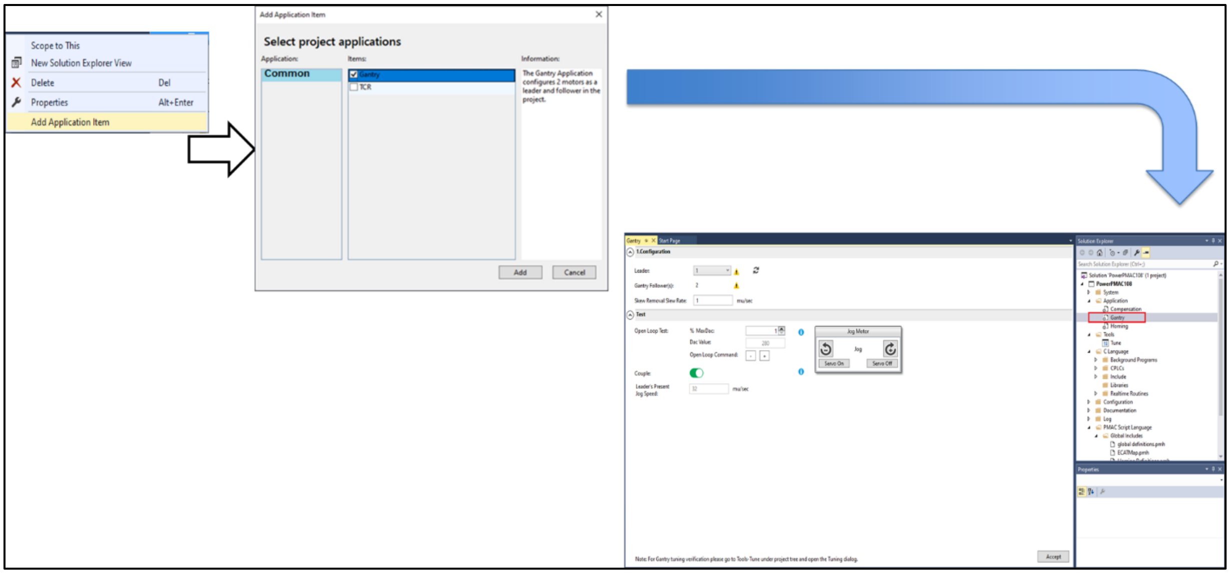
   b) **Upload Date from PMAC** button uploads Table number ***n*** corresponding to the table selected in the CompTableNumber field and its data values from Power PMAC and displays the complete configuration on the screen.

   c) Enable (up to this) Table button enables all the tables up to the current table.

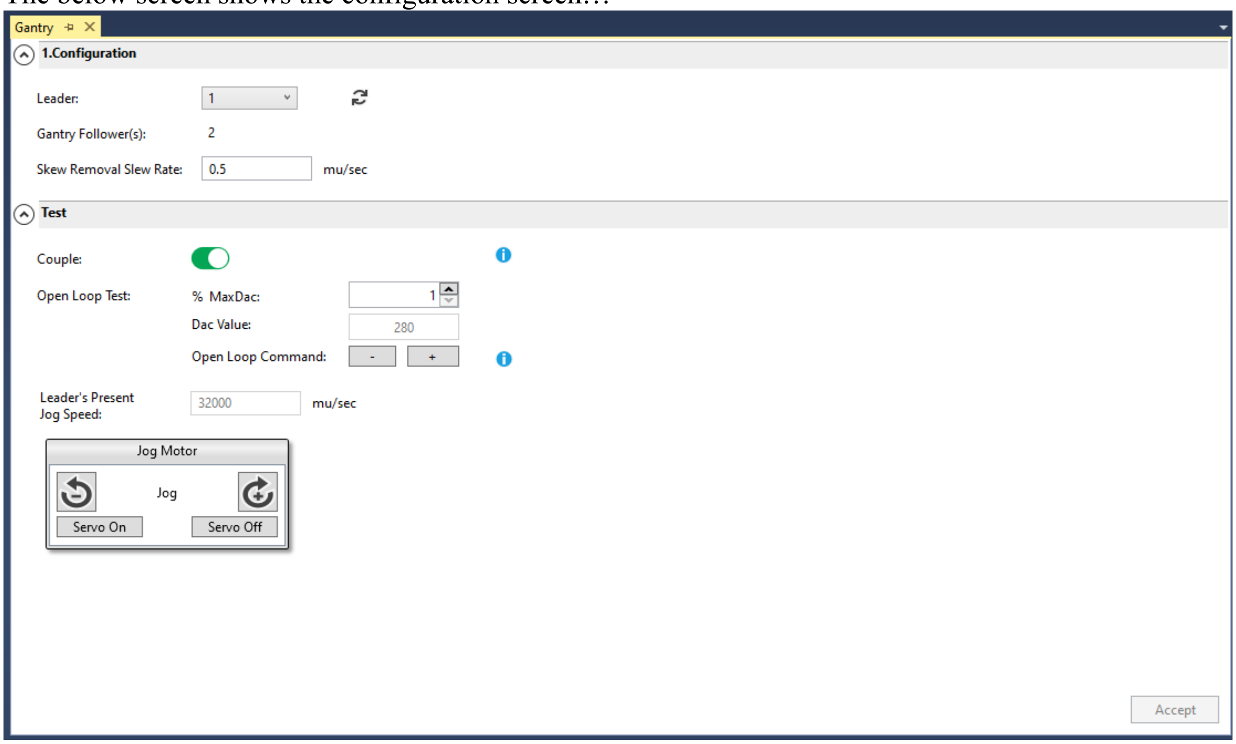
   d) Disable All Tables button, disables all the compensation tables.

Gantry

To add this App use the Add Application or use Project wizard. Typical workflow shown below. This workflow shows that the gantry application added using Add Application item context menu.



As shown above the Gantry added under Application Node, marked with Red square.
As it is part of the project it is integrated with project so all the setup parameters are stored with the project.
The below screen shows the configuration screen…



Gantry configuration involves two steps Configuration and Test .

1. Configuration

The Leader DropDown automatically filled if the motor exists in the project OR motors are active (Motor[n].servoctrl = 1). The follower is always next sequential number from Leader motor number. Current setup supports one leader and one follower.

Hoovering the mouse will give following warning…

This motor was set up outside of the system setup environment. Motor Structure elements will not be saved automatically. It is the user's responsibility to update/save those in their pmh files.
A disadvantage is the Gantry motor setup elements will not be written to the file and user will require to maintain the settings on their own.
It is our recommendation to use motors that are that are part of the project and configured using system setup to get all the project integration benefits.

Press this to refresh and update drop down.

Enables the motors to be coupled together as a Gantry system, or disable this feature to run as individual motors.

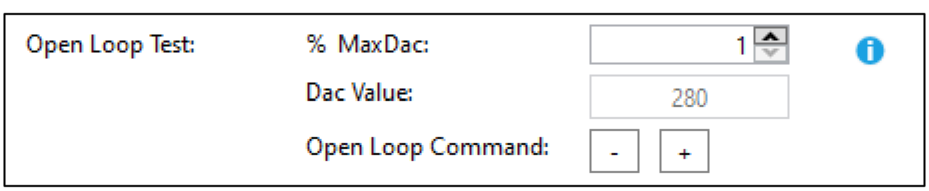
Like across the IDE the info icon will provide additional information about the parameter or control.

Please enter the skew removal rate. It is important to enter the non-negative floating skew rate for proper functioning of gantry.
On completing the configuration press Accept to setup gantry configuration for Leader and Follower. On success the output will be written to Power PMAC message window as well as respective Motor file. On build and Download these settings will be part of systemsetup.cfg file. A sample settings are shown below..

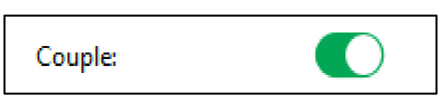
```
Motor[1].Ctrl=Sys.GantryXCtrl
Motor[1].ExtraMotors=1
Motor[2].Ctrl=Sys.GantryXCtrl
Motor[2].ServoCtrl=8
Motor[2].CmdMotor=1
Motor[2].GantrySlewRate=0.00088548422
```

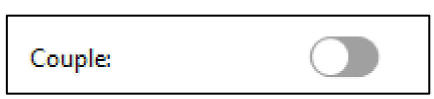
2. Test

The one important step before testing gantry is making sure the direction of the leader and follower is same. To do so the open loop control is provided. We recommend to use very low % MaxDac . This % can be selected in increments of 0.1.



Enter appropriate %MaxDac in the numerical box. The DAC output will be displayed in the Dac Value RO box.
Using + and – button test the gantry motor direction. If the directions are same then you can use Jog Box to verify Gantry motion.





Couple will enable gantry functionality indicated by Green switch. Couple OFF will decouple the gantry configuration. Default is Off because this is Gantry setup and user is setting up.

Note

Please choose appropriate and safe Open loop value by choosing safe %MaxDac value. If the leader and follower motion direction are not same, choosing high MaxDac value may damage the machine.

Couple is ON

```
Motor[1].Ctrl=Sys.GantryXCtrl
Motor[1].ExtraMotors=1
Motor[2].Ctrl=Sys.GantryXCtrl
Motor[2].ServoCtrl=8
Motor[2].CmdMotor=1
Motor[2].GantrySlewRate=0.00044274211
```

Couple is OFF

```
Motor[1].Ctrl=Sys.ServoCtrl
Motor[1].ExtraMotors=0
Motor[2].Ctrl=Sys.ServoCtrl
Motor[2].ServoCtrl=1
Motor[2].CmdMotor=0
Motor[2].GantrySlewRate=0
```

Typical Gantry Setup Steps

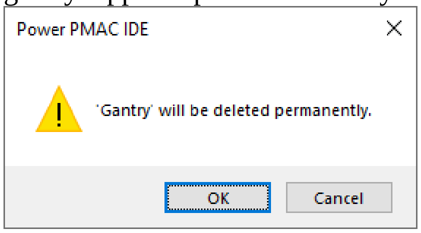
1. Open Project using Wizard and make sure to select Gantry application
2. Setup minimum two motors using system setup
3. Select Gantry and in the leader box select leader. Follower will be automatically added
4. This completes the gantry configuration
5. Go to test section. Default is decouple, in this make sure the direction of motion is same for both motors using small open loop move. Increments are 0.1%. Once direction confirmed click Couple to couple leader and follower
6. Using Jog control Servo On and try to Jog the gantry axis.
7.

Note

Please choose appropriate and safe Open loop value by choosing safe %MaxDac value. If the leader and follower motion direction are not same, choosing high MaxDac value may damage the machine.

Removing Gantry

To remove the Motors from gantry mode, simply right click on the gantry and select Delete or select gantry App and press Delete Key. On delete it will ask you to confirm the selection as shown below…

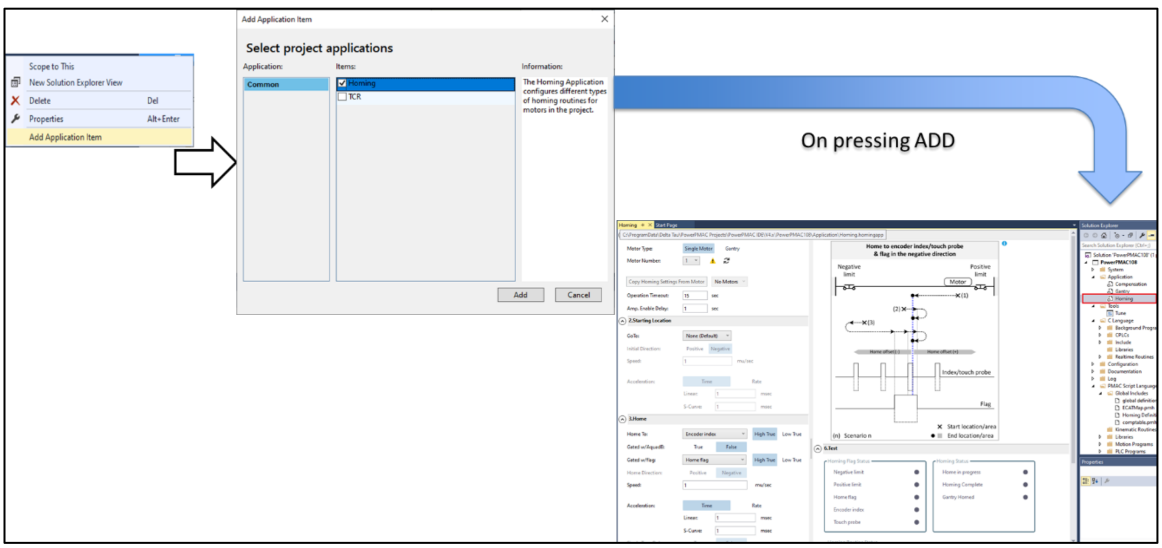


On selecting OK it will remove all the motor that are set in gantry mode and it will also update msetup file.

Once Delete there is no UNDO! User will need to reconfigure gantry.

Homing

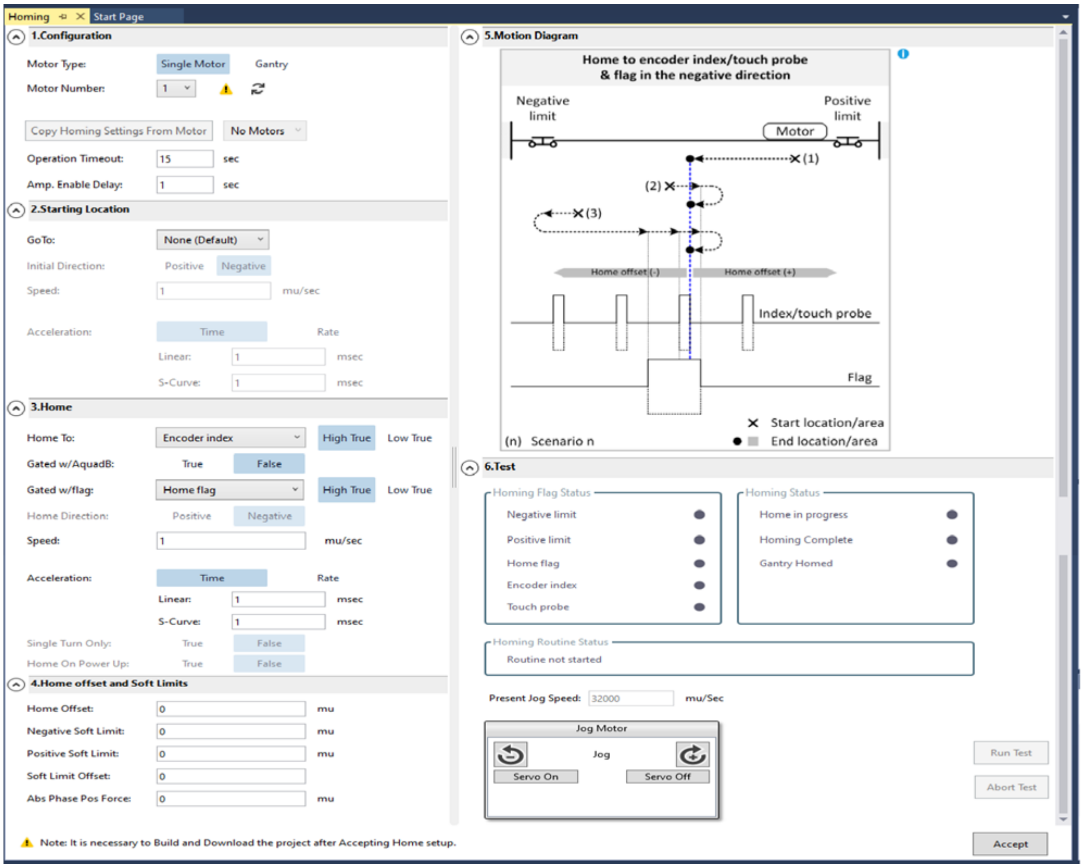
To add this App use the Add Application or use Project wizard. Typical workflow shown below. This workflow shows that the Homing application added using Add Application item context menu.



As shown above the Homing added under Application Node, marked with Red square.

As it is part of the project it is integrated with project so all the setup parameters are stored with the project.

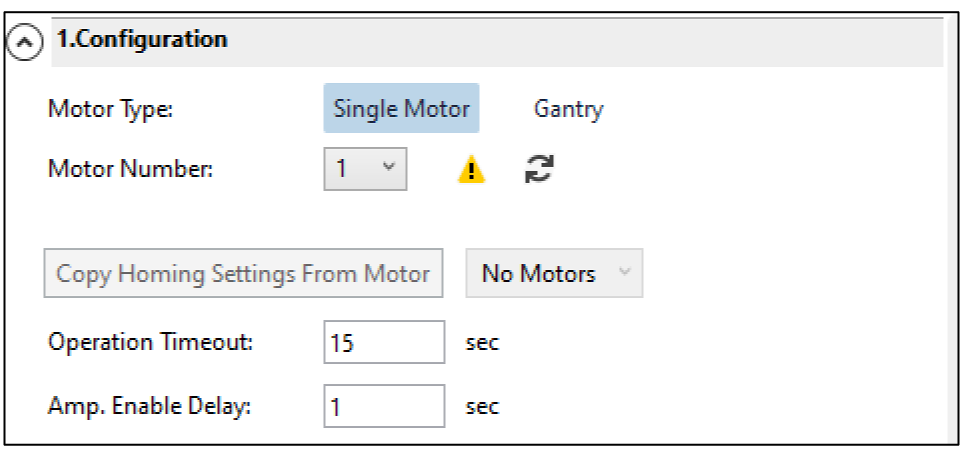
The below screen shows the configuration screen…



Homing configuration requires total 6 steps. Next section will explain these section.

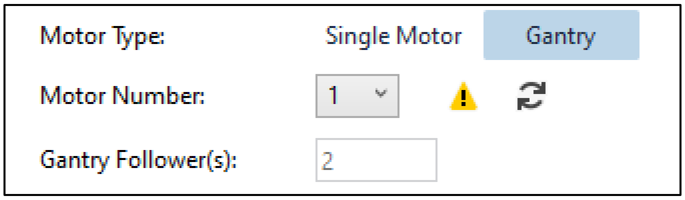
1. Configuration

Here is the configuration section. Select appropriate type for setting homing.

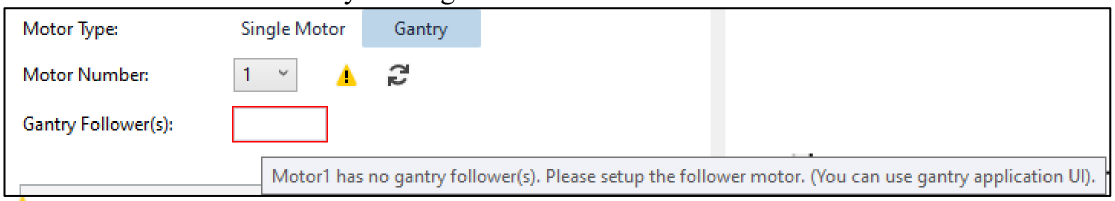


It can be Single Motor or Gantry motor. Default is Single motor. If Gantry is selected we will automatically check for follower motor and fill the box else error will be displayed when you will hoover the mouse on the Follower box that is with Red border.

follower found and Gantry homing selected



No follower found and Gantry homing selected…



Hoovering the mouse will give following warning…
This motor was set up outside of the system setup environment. Motor Structure elements will not be saved automatically. It is the user's responsibility to update/save those in their pmh files.
Warning sign indicates that the selected motor is not present in the Project tree under System-Motors. A disadvantage is Homing motor setup elements will not be written to the file and user will require to maintain the settings on their own.
It is our recommendation to use motors that are that are part of the project and configured using system setup to get all the project integration benefits.

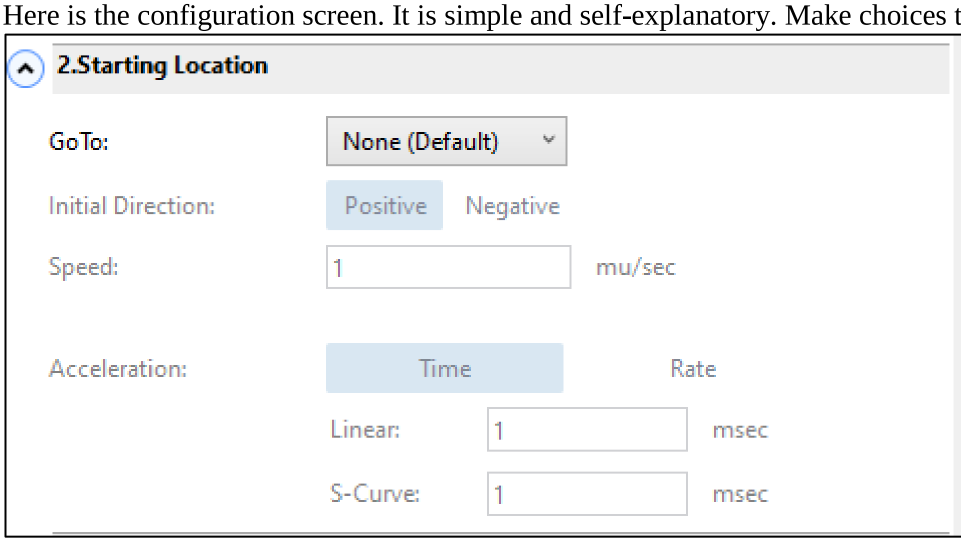
It is our recommendation to use motors that are that are part of the project and configured using system setup to get all the project integration benefits.

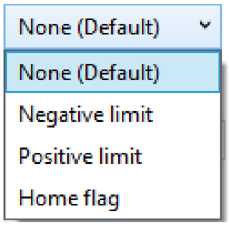
Press this to refresh and update drop down.

IF user is setting multiple homing configuration then after completing any configuration for motor then the motor will appear in the drop-down next to Copy Homing settings From Motor. This is beneficial if the homing configuration is same for other motors and will save the time.
Operation Timeout and Amp enable delay as they say are for compensating for homing condition delay and can be varied depending on the system.

2. Starting Location

Here is the configuration screen. It is simple and self-explanatory. Make choices to set Starting location

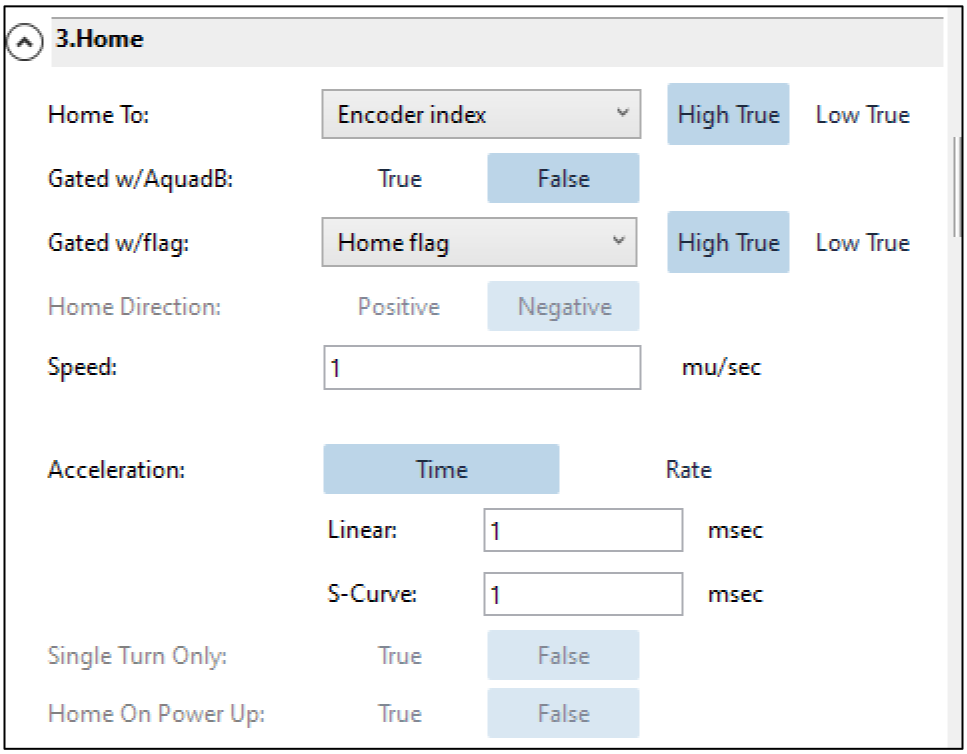


GoTo options are 

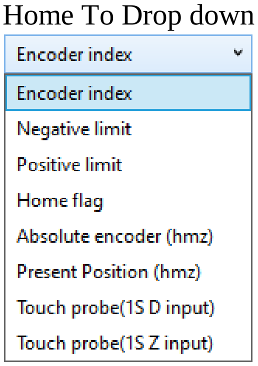
. These options are disabled if Home To option are selected as Touch Probe 1S D input or Touch Probe 1S Z input. Default is None.

3. Home

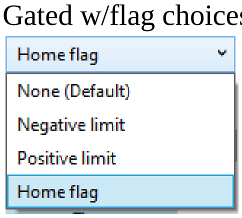
Here is the configuration screen. It is simple and self-explanatory. Make choices to set Home condition.



Home To Drop down choices



Gated w/flag choices



Default is Encoder Index and Gated w/flag is None
These choices are dynamic and depending on GoTo and Home To can change.
Gated w/AQuadB option only available if the Home To is selected as Index.

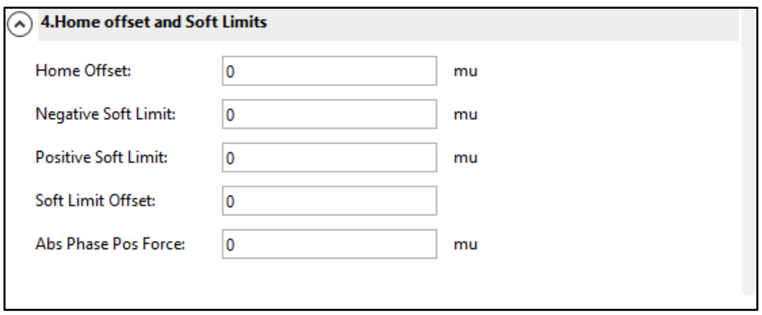
Touch Probe 1S D input and Touch Probe 1S Z input these are two methods for EtherCAT OMRON 1S drive only.
Power PMAC IDE will keep enhancing Homing for EtherCAT in future versions.

Note

EtherCAT Homing supprt for OMRON 1S drive only. Touch Probe 1S (D input/Z input) option will be only availabe if the Power PMAC IDE detects the Motor uses 1S drive.

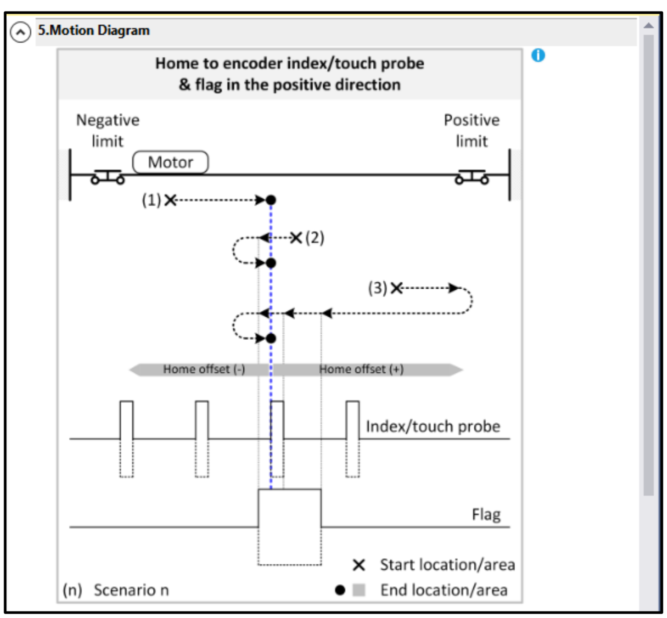
4. Home Offset and Soft Limits

Here is the configuration screen. It is simple and self-explanatory. Enter the appropriate value if needed.

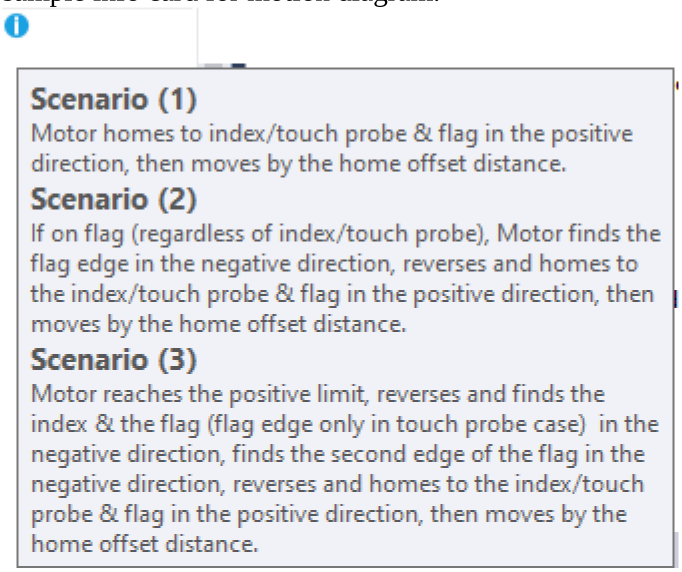


5. Motion Diagram

This section based on combination of GoTo, GoTo Direction, Home To, Home To Direction.
The diagrams are not available for Present Position (HMZ) and Absolute Encoder (HMZ.)
Here is the sample diagram for selected combination.



The info icon will provide Homing scenario for the current selction for the motion diagram. Here is the sample info card for motion diagram.



Following are the cases currently Motion diagram is available for..
Motion diagram number is not present on the user interface this is reference table for possible combinations. Each combination will have its own info card.

DIAGRAM	HOMETO	HOMETO DIRECTION
1	Index	Negative
	Touch probe (1S only)	
	Home flag	
2	Index	Positive
	Touch probe (1S only)	
	Home flag	
3	Negative limit	Must be negative
4	Positive limit	Must be positive
5	Index & flag	Negative
6	Index & flag	Positive
7	Index & negative limit	Must be negative
8	Index & positive limit	Must be positive

DIAGRAM	GOTO	GOTO DIRECTION	HOMETO	HOMETO DIRECTION
9	Negative limit	Must be negative	Index	Must be positive
			Home flag	
10	Positive limit	Must be positive	Index	Must be negative
			Home flag	
11	Negative limit	Must be negative	Index & flag	Must be positive
12	Positive limit	Must be positive	Index & flag	Must be negative
13	Home flag	Negative	Index	Negative
14	Home flag	Negative	Index	Positive
15	Home flag	Positive	Index	Negative
16	Home flag	Positive	Index	Positive

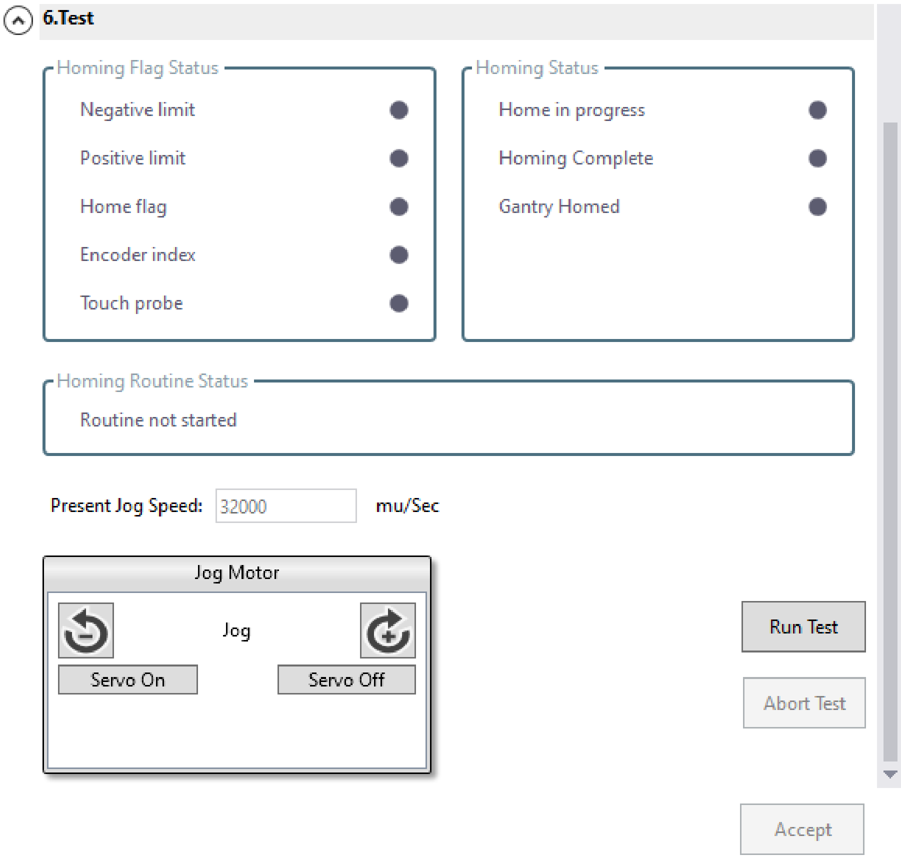
6. Test

This section as it said allows user to verify the Homing setup.
After setting all the section 1 to 4 user can Accept the setting.

Note

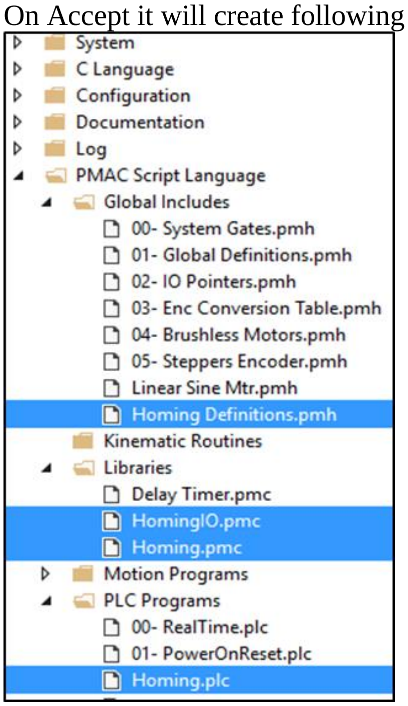
Homing Accept Expectation:
Build and Download of the project is necessary if user using C app otherwise Download All is necessary before testing the Homing configuration.

Here is the configuration screen

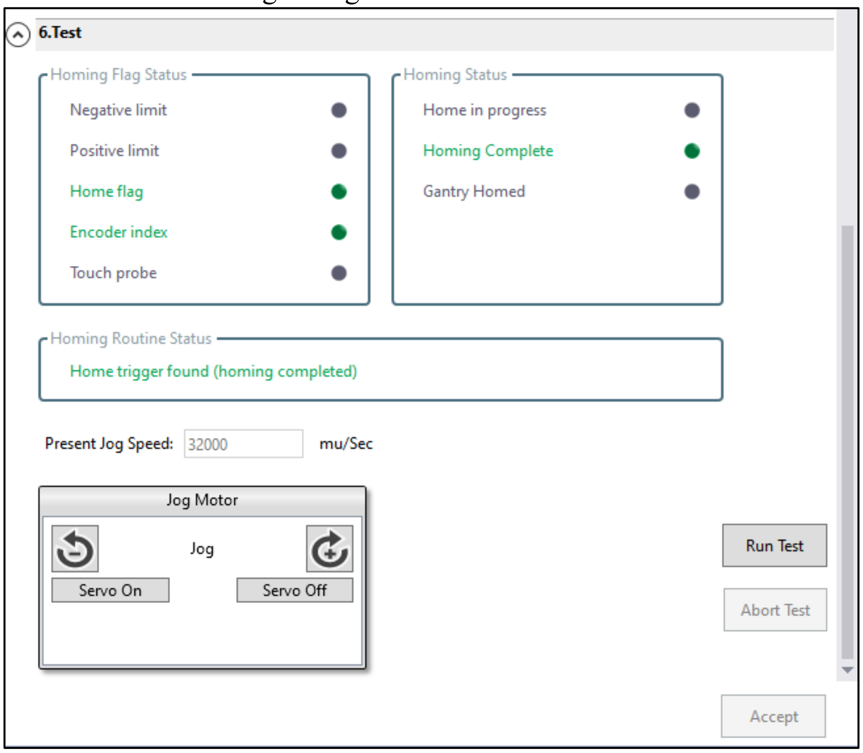


After Accept, build and Download the project. This will ensure the new configuration is uploaded to Power PMAC.

On Accept it will create following PLC and Motion program and automatically add to the project.



After Build and Download user can press Run Test and this will start Homing move for the selected Motor and configured condition. On success the screen will show like this … This is for current combination of homing configuration…



User can Abort the test using Abort Test button and only visible when Run Test is active.
User can use Homing status/Homing flag status to test the homing sequence by manually moving the motor or user can also check the limit or home switches and verify it's functioning using Homing status.

Typical Homing setup steps

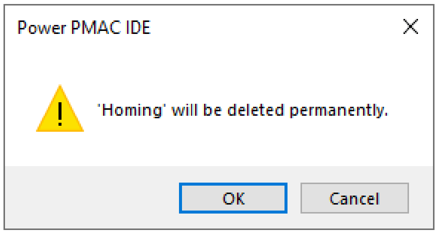
1. Open Project using Wizard and make sure to select Homing application
2. Setup motors using system setup
3. Open Homing setup screen by double clicking the Homing from Application node.
4. Select Motor type
5. Motor number will fill up automatically if Motor is setup else press refresh to update the list. Select the Motor from drop down.
6. If some other motor is already configured it will available to copy else 'Copy Homing Settings From Motor' drop down will say No motors.
7. Select Go To option and set Speed and Type of acceleration from Starting location
8. Select Home To option and set Speed and Type of acceleration from Home location
9. If needed setup Homing offset and soft limits. These value will be stored in the Motor setup file.
10. Make sure the Motion diagram shows the requested homing sequence.
11. Accept the settings to create necessary PLC and Motion program
12. Download all Programs or Build and Download the project.
13. Test the homing sequence using Run Test button. On pressing the button IDE issues following command

HmMtr(motor number) = 1
Enable PLC HomingPLC

Once the homing works satisfactorily user can invoke the Homing PLC for the appropriate motor from any other PLC based on the Input.

Removing Homing

To remove the homing configuration, simply right click on the homing and select Delete or select homing App and press Delete Key. On delete it will ask you to confirm the selection as shown below…



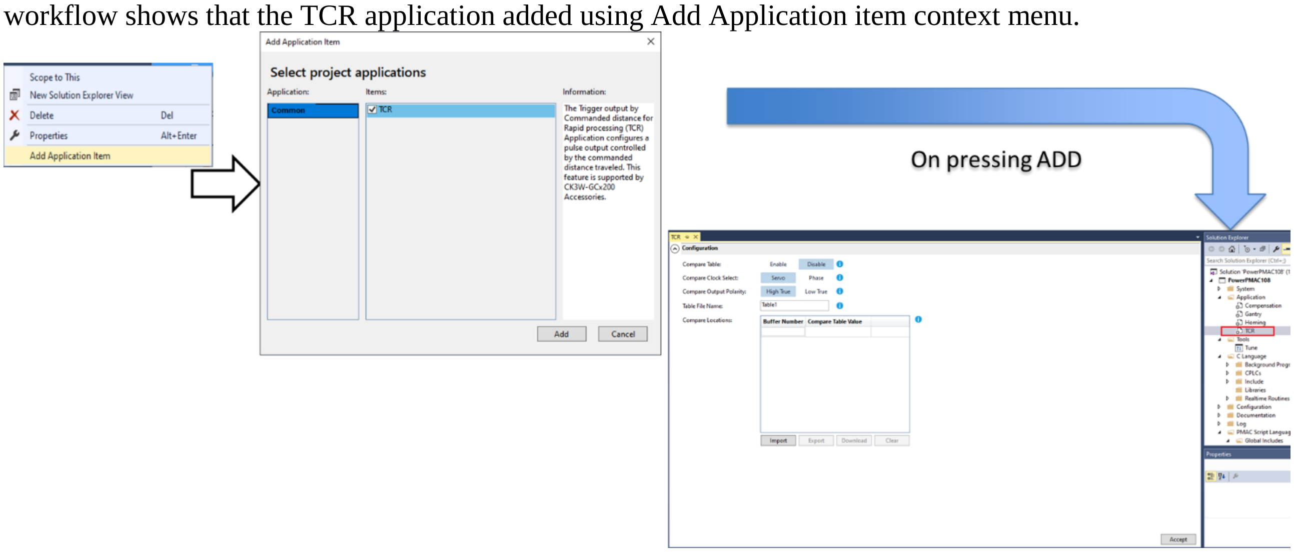
On selecting OK it will remove homing configuration for the motors. It will also remove the files created by homing setup on Accept . Following files will be removed on deleting homing application from project…
HomingDefinitions.pmh
HominIO.pmc
Homing.pmc
Homing.plc.

Note

Once Homing application is Delete there is no UNDO! User will need to reconfigure homing.

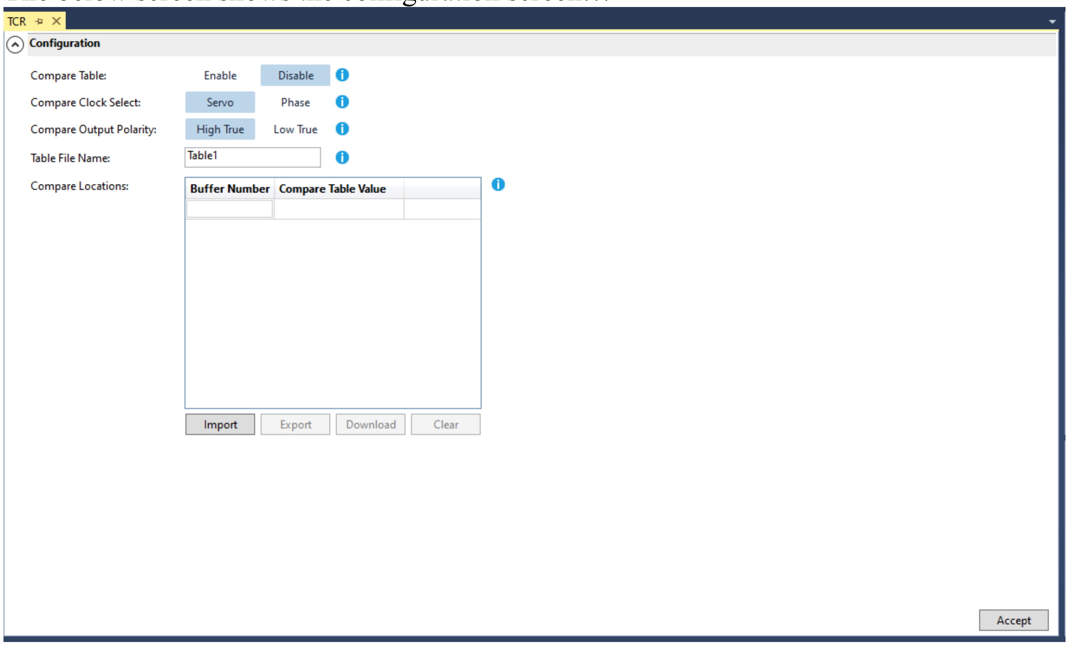
TCR

To add this App use the Add Application or use Project wizard. Typical workflow shown below. This workflow shows that the TCR application added using Add Application item context menu.



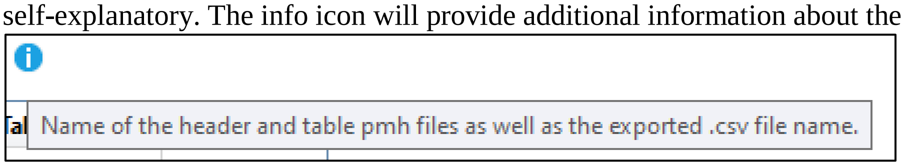
As shown above the TCR added under Application Node, marked with Red square.

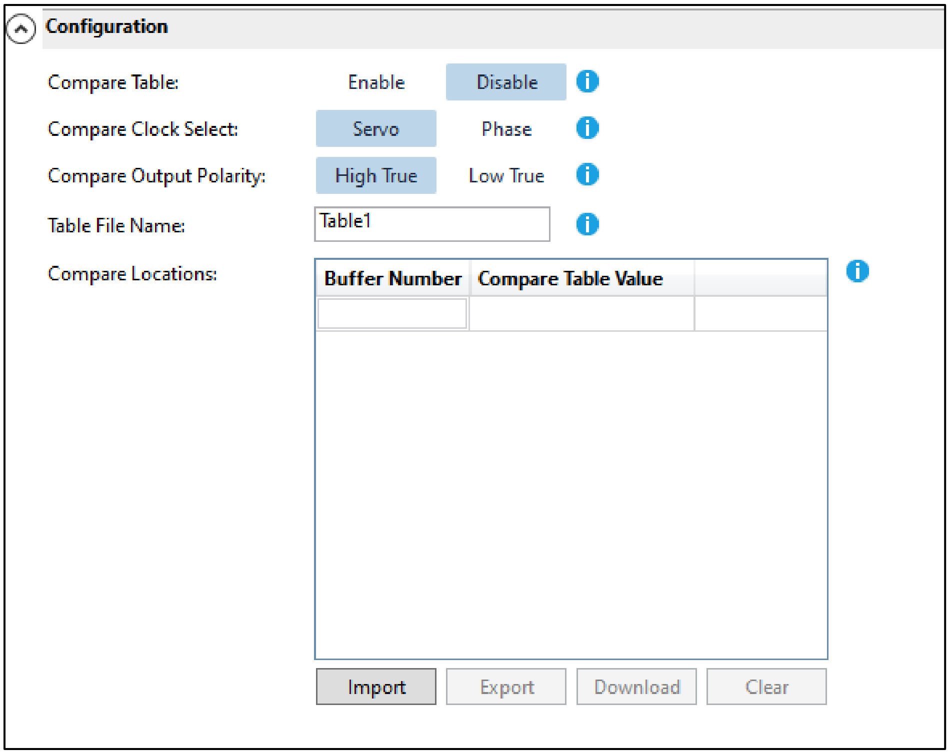
As it is part of the project it is integrated with project so all the setup parameters are stored with the project.
The below screen shows the configuration screen…



1. Configuration

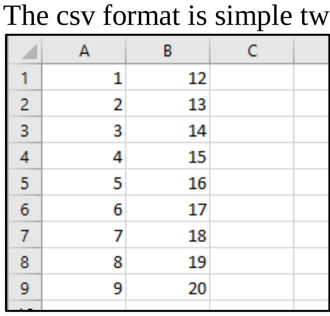
Below is the configuration Section. The user interface allows user to configure table. The table is used to generate Trigger output by Commanded distance for Rapid processing. The configuration parameter are self-explanatory. The info icon will provide additional information about the parameter as shown below..



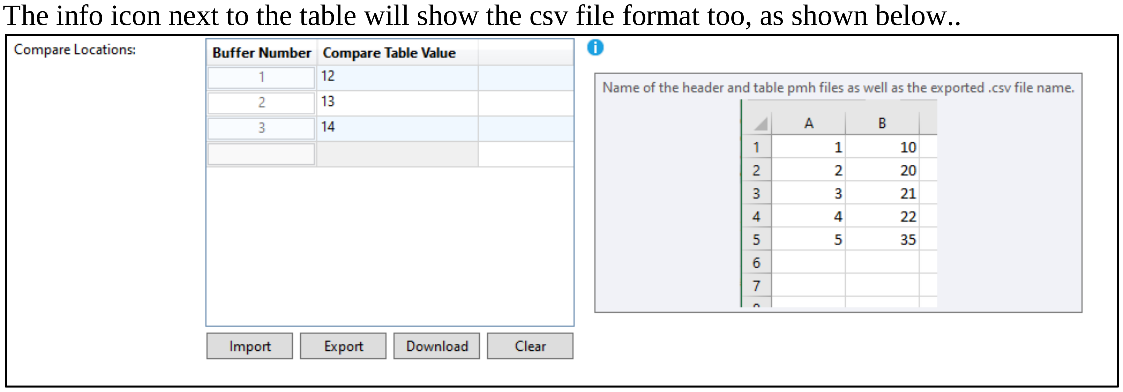


User can chose source of the clock , Polarity. Compare Location is fully editable table where user can import the table from csv file or type the table entry for quick testing the TCR feature. User can delete , export or clear the Table.
The Compare Table option must be disable while loading (Download) the table to the card. There are total 4095 entries possible in the table.

The csv format is simple two column, first the number and the second column value, as shown below…

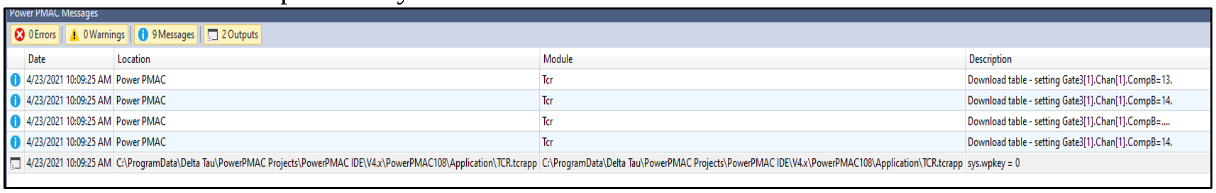


The info icon next to the table will show the csv file format too, as shown below..



On pressing Download it does following action

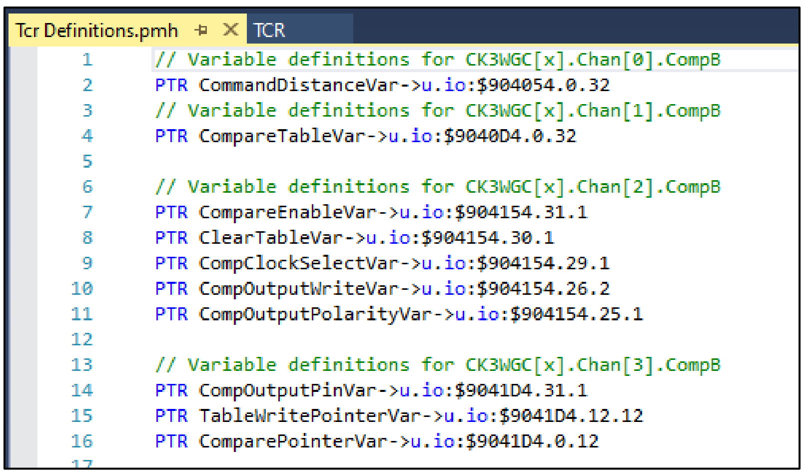
1. The table is downloaded CK3WGC hardware.
2. It also writes what we are writing to the Power PMAC Message window.
3. Generates file specified by user under Table file name under Global includes folder.



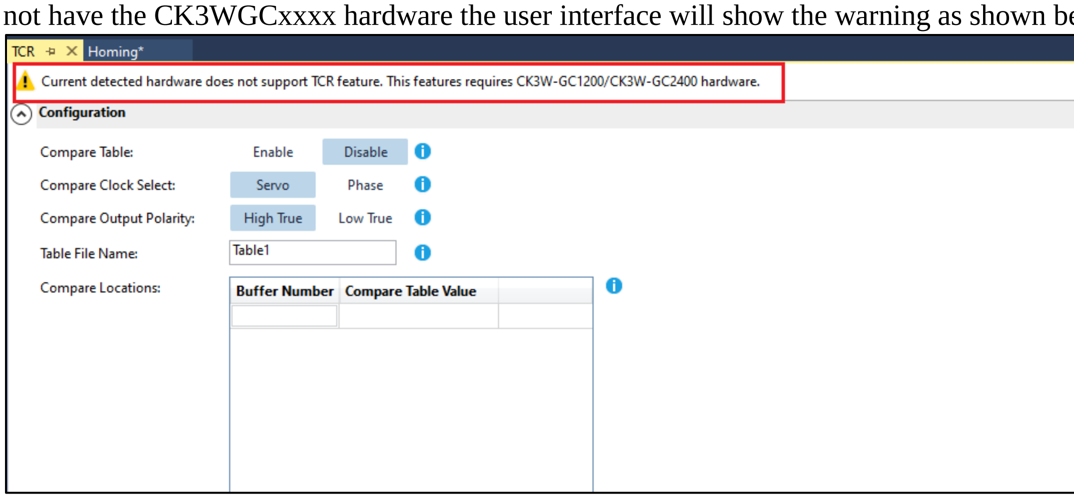
User can use this file from their HMI software and using gpascii command will be able to download the table to the hardware. The command for Table.pmh file is..

Gpascii –iTable1.pmh

On pressing Accept it generates TcrDefinition.pmh file that user can use in motion, plc script file. The file looks like..



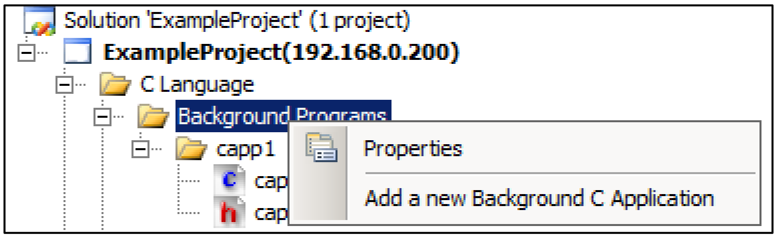
This Application is supported by CK3WGCxxxx hardware only if user opens TCR application that does not have the CK3WGCxxxx hardware the user interface will show the warning as shown below..



C Language

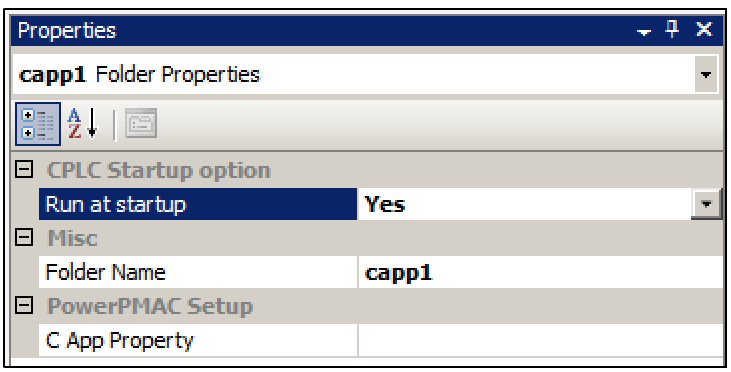
Background Programs

This folder contains the files for background C programs. These are applications that run in the free background time of Power PMAC. A new C application can be added by right-clicking the Background Programs folder and then clicking "Add a new Background C Application":



Give the application a name and the IDE will create a new folder for that application's source code files underneath the Background Programs folder.

If the application needs to run at startup right-click the application's subfolder e.g. called "capp1" in the screenshot above, click Properties and then in the Properties Window select "Yes" in the "Run at startup" field as shown below:

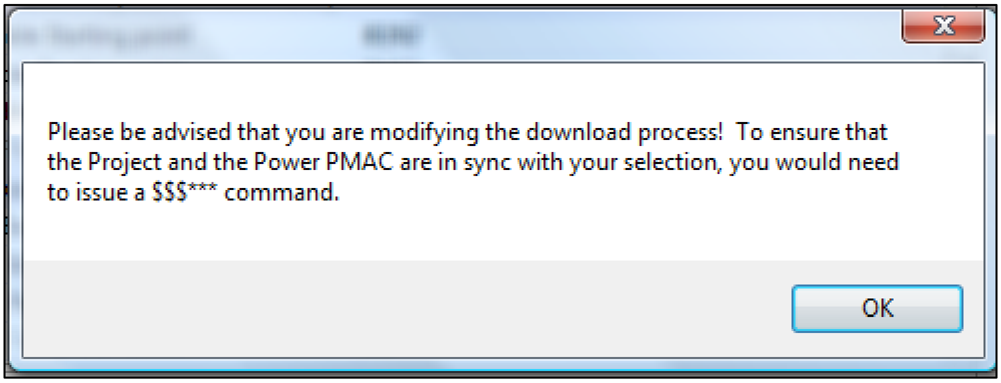


Downloading the C Source

C source files can be individually selected to be downloaded to the device. By default, no C source files are downloaded. To change this setting:

1. Right-click on the project and select **Properties.**
2. Locate the **Download C Source Files** option and select **Yes** or **No**.
3. If **Yes** is selected the project will download the C source files to the Power PMAC.

Any time changes are made to the **Download C Source Options** a message indicating that a **$$$*** ** command should be issued before downloading the project will be displayed. Since some of the C source files might be on the Power PMAC it is necessary to reset the device before downloading the project.



CPLCs

This folder contains folders for Background C PLCs (BGCPLCs) and Real-Time Interrupt CPLCs (RTICPLCs). To create a new BGCPLC right-click the CPLCs folder and click “Add a new User CPLC Project”:

Select the new BGCPLC’s number and the IDE will create a new folder for that BGCPLC’s source code.

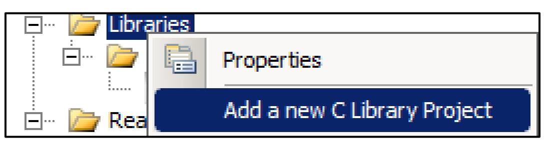
It is not possible to create a new RTICPLC folder because only one RTICPLC is permitted on the Power PMAC. This is found in the folder labeled “rticplc” under the “CPLCs” folder.

Include

The Include folder contains C header files (*.h) which can be included by any of the C program files (*.c).

Libraries

The Libraries folder contains subfolders which contain libraries which have been written or included. To create a new subfolder in the library right-click on Libraries and select “Add a new C Library Project”:



Give the library a name and the IDE will create a new subfolder where the source (*.c) and header (*.h) files can be placed.

Realtime Routines

The Realtime Routines folder contains the source and header files for user-written servo and phase algorithms. The source file is called usrcode.c and the header file is called usrcode.h.

usrcode.c contains the functions used in user-written servo and phase algorithms. usrcode.h contains the prototypes of these functions and exports the functions as symbols. Please refer to the section of the

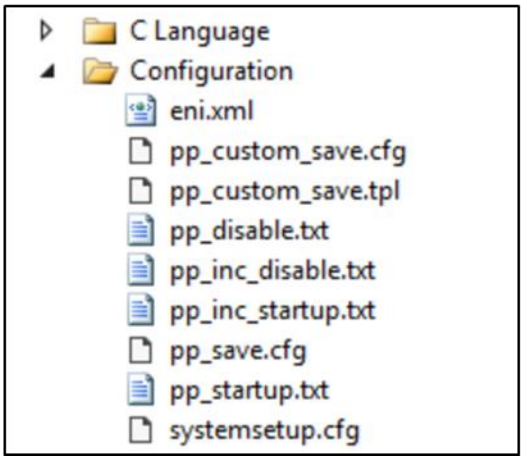
Power PMAC User Manual called “Writing C Functions and Programs in Power PMAC→User-Written Phase Routines and User-Written Servo Routines” for more details on how to write these files.

To learn how to associate motors with these routines please see the section labeled “Configuring User-Written Servo and Phase Algorithms”.

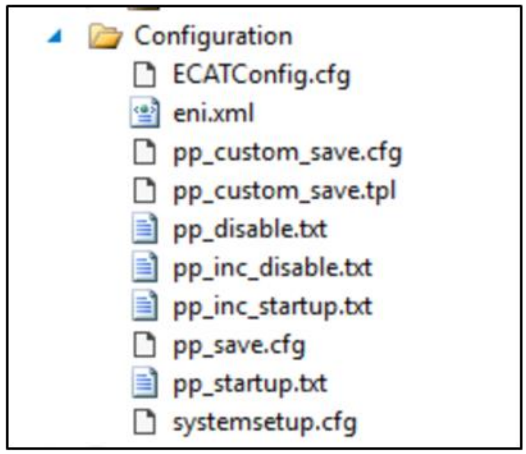
Configuration

This folder contains the files which control what is run upon downloading the project to Power PMAC, upon booting up Power PMAC and upon issuing a **save** command. The contents appear as follows:

Configuration with no EtherCAT



Configuration with EtherCAT



eni.xml

In a project system that is not using EtherCAT this file is empty. For projects with EtherCAT this file stores the EtherCAT information. This file is generated when the Load Mappings command is selected from context menu on Master node.

pp_custom_save.cfg

This file is automatically generated from pp_custom_save.tpl when a **save** command is issued to the Power PMAC. It contains a backup of the settings of any of the structures added to pp_custom_save.tpl. If this file is missing from the project e.g. if a project is opened that was created before IDE version 1.7.x.x, it can be generated by right-clicking the Configuration folder and then clicking “Download Config Files”.

Note

Do not modify this file. It is automatically generated.

This feature requires firmware version 1.6.1.1 or newer and IDE version 1.7.x.x or newer.

pp_custom_save.tpl

Add any Power PMAC parameter in this file and it will be added to pp_custom_save.cfg upon issuing a **save** command to PowerPMAC. For example typing **Motor[1].Servo.Kp** into this file and then issuing a **save** command to Power PMAC will result in the value of this parameter (**Motor[1].Servo.Kp=4** by default) being written to pp_custom_save.cfg.

To add a whole structure tree use the **backup** command. For example to back up every setting in the **Motor[1]** tree add the command **backup Motor[1].** Into the pp_custom_save.tpl. If this file is missing from the project e.g. if a project is opened that was created before IDE version 1.7.x.x, it can be generated by right-clicking the Configuration folder and then clicking “Download Config Files”.

Note

After modifying pp_custom_save.tpl right-click the Configuration folder and click “Download Config Files” so that the changes will take effect. After this when a **save** is issued the values of the parameters specified in this file will be added to pp_custom_save.cfg.

This feature requires firmware version 1.6.1.1 or newer and IDE version 1.7.x.x or newer.

pp_disable.txt

This file is the first file loaded on the download of the entire project. This file should cause programs to be aborted, PLCs to be disabled, motors to be killed, and buffers to be cleared; all for safety purposes. The following is an example:

```
&*A                    //Abort All Programs
disable plc 0..31      //Disable all Script PLCs by number

#*k                    //Kill all the motors is commented out
clear all buffers
```

pp_inc_disable.txt

This file is the first file loaded on the download of an incremental project, that is, selected files. This file should cause programs to be aborted, PLCs to be disabled, motors to be killed, and buffers to be cleared; all for safety purposes. The following is an example:

```
&*A                    //Abort All Programs
disable plc 0..31      //Disable all Script PLCs by number

#*k                    //Kill all the motors is commented out
clear all buffers
```

pp_startup.txt

This file is the last file loaded on the download of the entire project. The commands within this file will run when Power PMAC boots up. Typically this file starts the first programs to run on the Power PMAC on start up. The recommended way of starting Power PMAC is to enable PLC 1 which then initializes whatever parameters and starts whatever programs have been set. The following is an example:

```
enable plc 1;
```

pp_inc_startup.txt

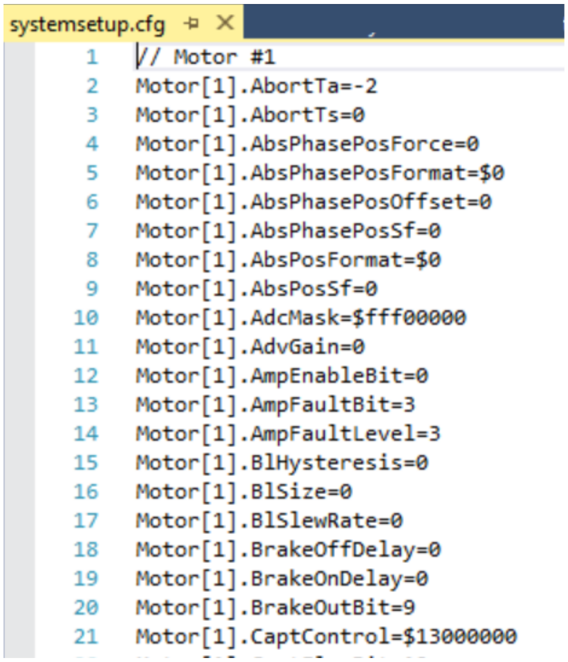
This file is the last file loaded on the download of an incremental project or selected files. Typically, this file starts the first programs to be run on the Power PMAC on start up. The recommended way of starting Power PMAC is to enable PLC 1 which then initializes whatever parameters and starts whatever other programs are needed. The following is an example:

```
enable plc 1;
```

systemsetup.cfg

This file is maintained by project system. It is generated when project is built.

The file looks like this:

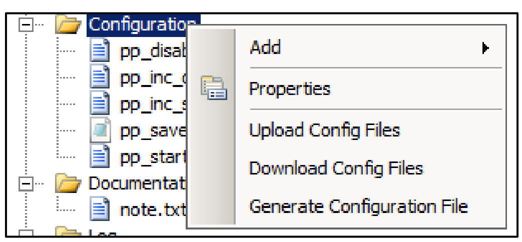


This file includes Motor, Coordinate system and Encoder table settings.

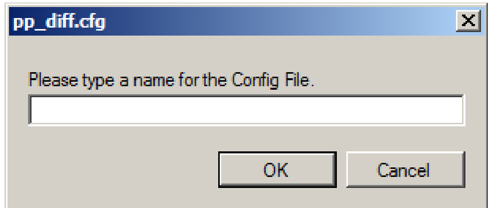
Generating Configuration Files

Another feature of the Configuration Folder is the ability to generate Configuration Files which are a file containing only the structures which have been modified from their default settings since the last factory reset ($$$***).

To access this feature right-click the Configuration Folder which shows this menu:



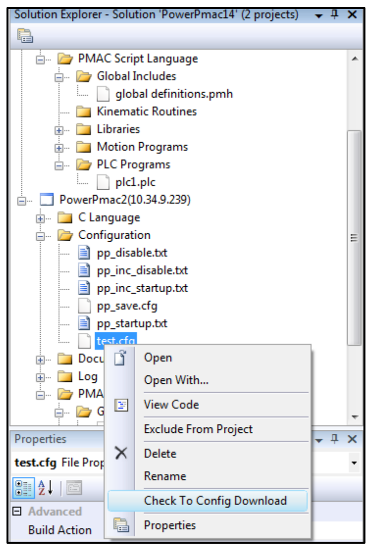
"Upload Config Files" will upload any Configuration Files which have been generated from the Power PMAC to the host computer. "Download Config Files" will download any configuration files which are stored on the host computer to the Power PMAC. "Generate Configuration File" will create a new configuration file containing the settings presently in Power PMAC at the time this was selected. Add a name for the configuration file and it will be stored in the Configuration Folder:



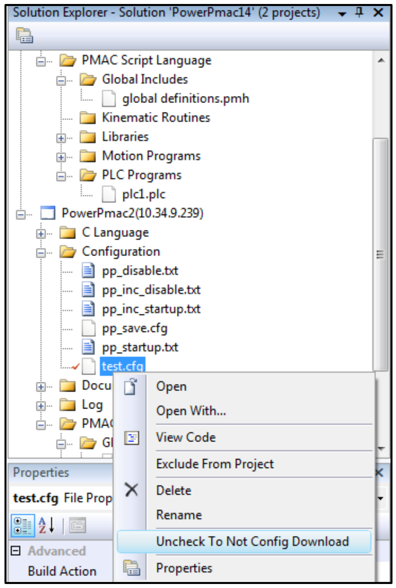
IDE V4.x maintains the systemsetup.cfg and automatically downloads so generating config file in no longer needed. This feature is available for backward compatibility.

Note

Spaces or dots are not supported in this filename. In order to download a configuration file right-click the file to be downloaded and then click “Check to Config Download”:



The file selected will receive a red check mark to the left of its filename. Then right-click the Configuration folder and click “Download Config Files” to download this file.
Only one configuration file can be checked at any one time. To deselect a file, preventing it from being downloaded, right-click the file and then click “Uncheck to Not Config Download”:



This will remove the check mark from the file and it will not be downloaded when "Download Config Files." is selected.

Documentation

This folder contains files for documentation purposes. Any text-based file can be added into this folder. None of these files get downloaded to Power PMAC but simply remain in the project folder on the host computer.

Log

This folder contains the log files created when a project is downloaded to the Power PMAC. These file should never be edited; they are for reference purposes only.

pp_proj.log

This file is a history log of files loaded to Power PMAC since Power On, $$$, $$$*** or downloading from the IDE.

This log is made up of the following 3 sections:
[PMAC_HARDWARE] Consists of values formatted as follows:

```
Gate1AutoDetect=0x50
Gate1AddrErrDetect=0x400
Gate2AutoDetect=0x0
Gate3AutoDetect=0x0
CardIOAutoDetect=0x1
CardDPRAutoDetect=0x0
```

Each bit of the AutoDetect represents a card being detected at the 16 to 20 different possible addresses for the particular card type. An "AddrErrDetect" non-zero value means that the gate-card was detected at a second location.

[PMAC_CONFIG] Consists of values formatted as follows:

```
Successful Configuration using "/var/ftp/usrflash/Project/Configuration/pp_save.cfg"
```

This section logs the success or failure of loading the saved configuration variables.

[PMAC_PROJECT] Consists of values formatted as follows:

```
Start of Project Loading using INI File: "/var/ftp/usrflash/Project/Configuration/pp_proj.ini"
Including Project File: /var/ftp/usrflash/Project/Configuration/pp_disable.txt
Including Project File: /var/ftp/usrflash/Project/PMAC Script Language/Global Includes/global
definitions.pmh
Including Project File: /var/ftp/usrflash/Project/PMAC Script Language/Libraries/subprog template.pmc
Including Project File: /var/ftp/usrflash/Project/PMAC Script Language/Motion Programs/prog
template.pmc
Including Project File: /var/ftp/usrflash/Project/PMAC Script Language/PLC Programs/plc template.plc
Including Project File: /var/ftp/usrflash/Project/Configuration/pp_startup.txt
Successful load of preprocessed File: "/var/ftp/usrflash/Temp/pp_proj.pma"
Run OK on Linux Program: /var/ftp/usrflash/Project/C Language/Background Programs/cplc1.out
```

This section logs the loading of the Power PMAC project files. This logging occurs after Power On, $$$, $$$*** and for each download from the IDE.

pp_error.log

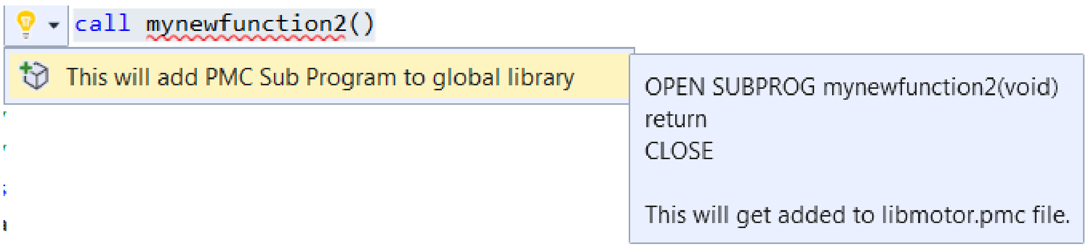
This file is a log of any errors on the last loading of the project into the Power PMAC.

pp_error_hist.log

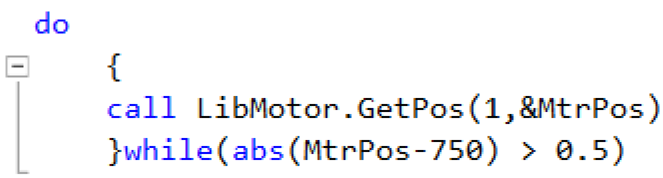
This file is a history log of any errors in the loading of the Power PMAC since Power On, $$$, $$$***, or downloading from the IDE. It is broken up into the same three sections as the **pp_proj.log** shown above

PMAC Script Language Folder

This folder contains all of the programs and header files which are written in the Script language. Language service for the PowerPMAC script language supports features like light bulb to suggest the fix or improved error and warning notification,

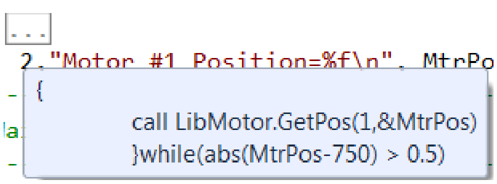


Also supported is indentation decision maker like do-while, if, for etc.



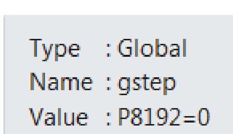
TO

And shows the code if by hovering the mouse on collapsed sections



The Language service for the Power PMAC script language supports ‘Go to definition’ for subprograms and ‘Go to Declaration’ for variables. Adding the typed variable, if it is not declared before, will add it to globaldefinition.pmh file as a potential light bulb fix.

The Language service for the Power PMAC script language reads the value from the Power PMAC and displays it as below:



Global Includes

This folder contains all of the header files which are to be downloaded before all other Script programs are downloaded. These header files usually contain **global**, **csglobal**, and **ptr** variable definitions which are used in the other programs. The variables can be initialized in these header files. These files can also contain preprocessor directives such as **#define** statements.

Kinematic Routines

This folder contains the files for Forward and Inverse Kinematic Subroutines. It is recommended to make separate files for each subroutine.

Libraries

This folder contains the files for subprograms which can be called by any program written in Script.

Motion Programs

This folder contains the files for motion programs.

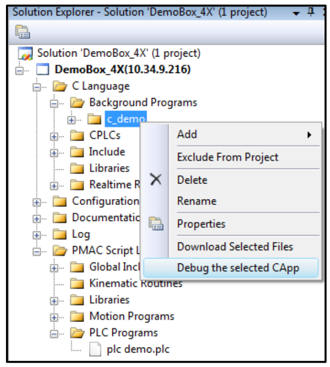
PLC Programs

This folder contains the files for PLC programs.

Debugger

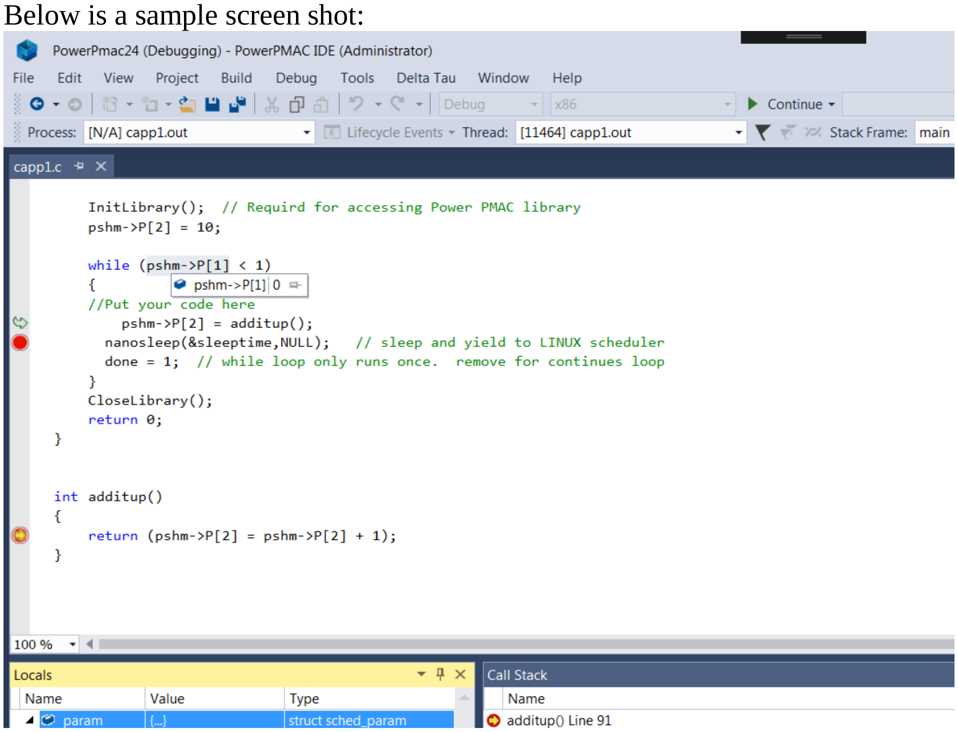
C language debugger

The IDE supports a fully featured GNU debugger which is integrated with modern Visual studio debug interface. This supports the majority of the debugger functionalities and interfaces including starting, stopping, breaking, setting break points, stepping, checking the execution stack, fully integrated watch table, local variables, Auto display, tooltip, and many other modern debugging functionalities.
After successfully downloading the Power PMAC project right-click the Background C Application (under Background Programs) that is to be debugged as shown below:



Select the context menu “Debug the selected CApp” to start the debugger. This will launch the same debug environment used when debugging a Script PLC.

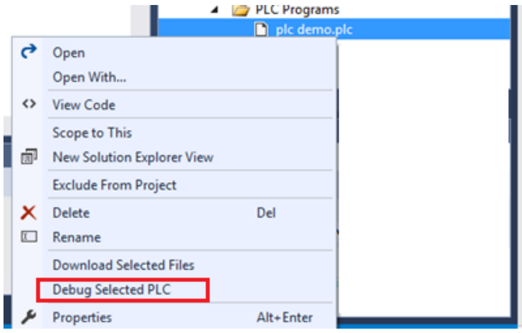
A breakpoint can be set before or after the debugger is launched. To set the breakpoint after the debugger is launched make sure that the Background C Application is in a loop; otherwise the program execution will be completed and it will not encounter the break point. Breakpoints can be set by pressing F9.
Below is a sample screen shot:



Script PLC Debugger

The IDE supports debugging of script PLC.

1. Open the project that needs to debug.
2. Build and download the project.
3. Right click on the script PLC to debug.



4. Make sure to have breakpoint on the line in the plc as shown below:

```
open plc step_demo
local axiscnt, mask, MtrPos;
//----------------------------------------------------------
// Display I130..133
//----------------------------------------------------------
send 2, "I130=%f\n&I131=%f\nI132=%f\nI133=%f\n", I130,I131,I132,I133

Ldata.Coord = 1     // set CS # = 1
mypvar1 = 0
myqvar2 = 0
mymvar1 = 0

```

5. Select to Debug PLC from Context menu.
6. The Debugger will be launched and the breakpoint is hit as shown below:

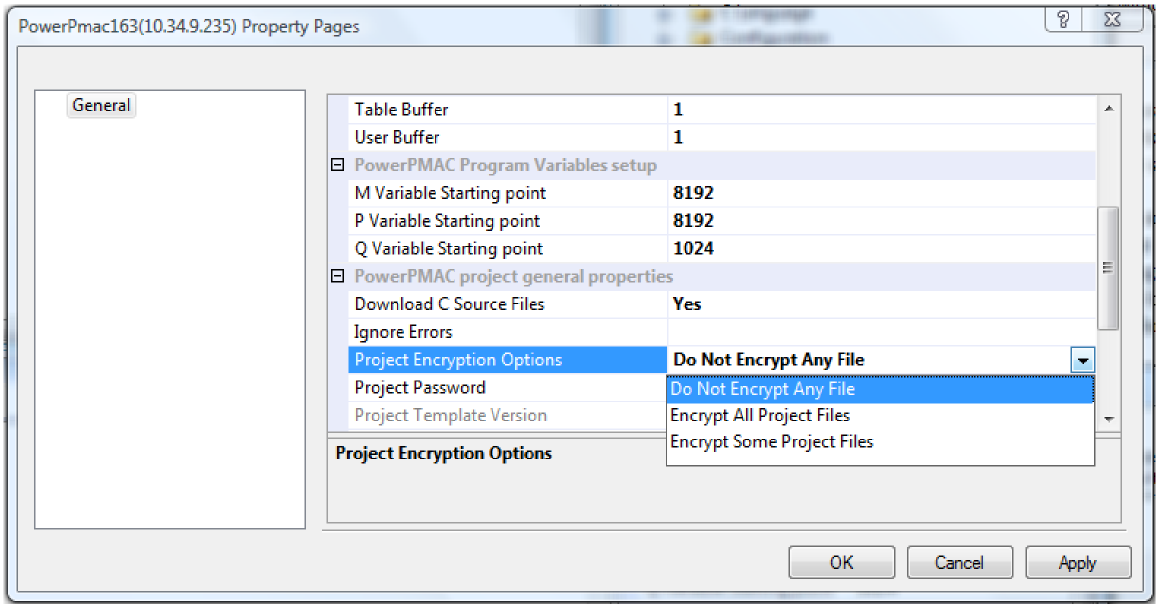
```
open plc step_demo
local axiscnt, mask, MtrPos;
//----------------------------------------------------------
// Display I130..133
//----------------------------------------------------------
send 2, "I130=%f\n&I131=%f\nI132=%f\nI133=%f\n", I130,I131,I132,I133

Ldata.Coord = 1     // set CS # = 1
mypvar1 = 0
myqvar2 = 0
mymvar1 = 0
```

7. Standard Visual studio debug keys like F10, F11, etc. are supported.

PROJECT ENCRYPTION

To encrypt the project and then download the encrypted project to Power PMAC do the following:

1. Right click on the project in the Solution Explorer and select Properties.
2. From the Properties window choose one of the following 3 options, as shown in the screenshot below, before downloading the project:

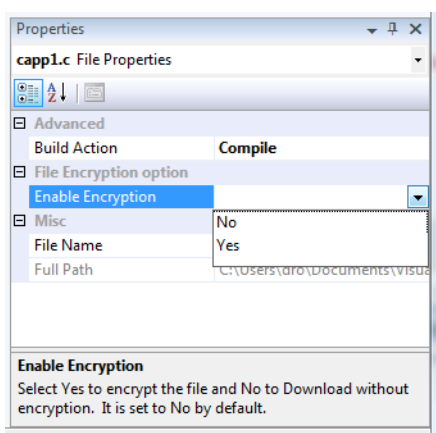


The options are described as follows:

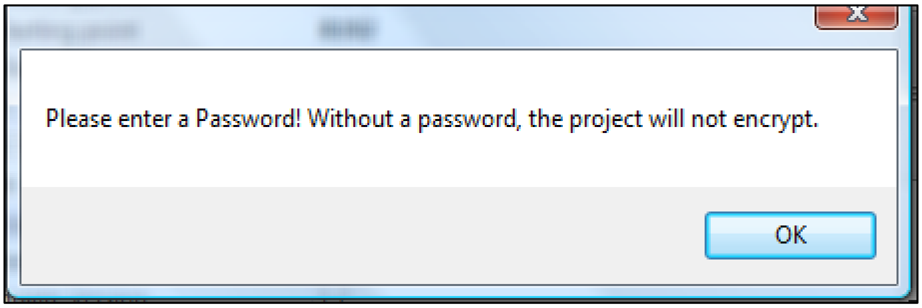
a. **Do Not Encrypt Any File**: This option is for downloading the project "as-is", i.e. the project will get downloaded to the Power PMAC without any encryption. The C source files will be downloaded if the **Download C Source File** option is set to **Yes.** Otherwise, no C source files will be downloaded.

b. **Encrypt All Project Files**: This option will force the IDE to encrypt all project files before downloading them to the Power PMAC.

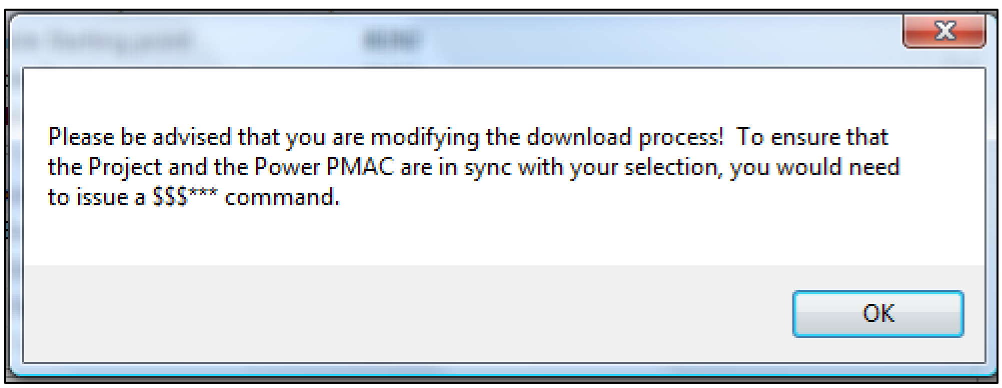
c. **Encrypt Some Project Files**: This option will allow only certain files within the project to get encrypted and be downloaded to the Power PMAC. Once this option is selected the user should select the files that will be encrypted by right-clicking the file, selecting Properties and then setting **Enable Encryption** to **Yes,** as shown in the screenshot below



3. Project Password: Once an encryption option is selected a password should be provided to be used by the encryption tool. The Power PMAC will use the same password to decrypt the files and load them into Power PMAC. If no password is provided the project will not be encrypted and the following message will be displayed:



4. Encryption Message: Any time the **Project Encryption Option** is changed a message indicating that a "$$$***" should be issued before downloading the project will be displayed. Since some files might be on the Power PMAC it is necessary to reset the Power PMAC before downloading the encrypted (or Original project) to the Power PMAC.
5.



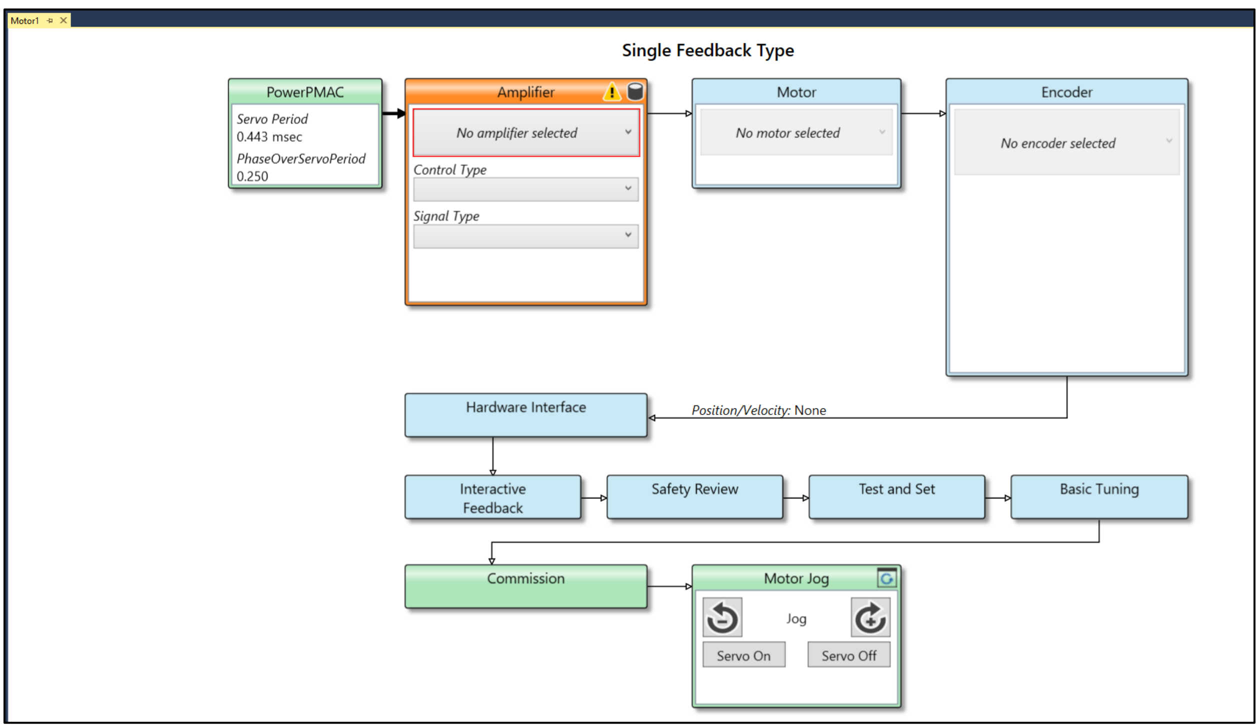
MOTOR SETUP

The following section describes setting up the local motor and EtherCAT motor.

Refer Project folder section for details about each blocks from Topology. To avoid duplication only steps are listed. In some cases detail information is provided.

Local Motor: (Single or Dual feedback)

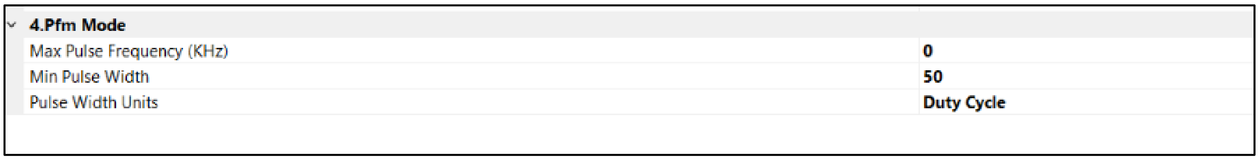
1. Open a new project
2. Setup Power PMAC clock settings by clicking System – CPU folder or double-clicking the Power PMAC block on the Topology view, to open the Global Clock view
3. Go to the Motors Node, right click and Add Motor. Choose either Single Feedback or Dual Feedback. The feedback type can be changed if necessary.
4. On successful addition of the Motor, the Motor Toplogy View will be displayed.
5. As explained in the Motors section follow the Topology Block flow to setup the Motor. The Toplogy block coloring will acts as a guide. The User Units block is part of Encoder block and it is not mandetory though we do recommend it is set.
6. The Commissioning blocks are, also, not mandatory.
7. To complete each block the setting must be Accepted.



Local Motor: No Feedback Motor (Step & Direction)

1. Open a new project
2. Setup Power PMAC clock settings by clicking System – CPU folder or double-clicking the Power PMAC block on the Topology view, to open the Global Clock view
3. Go to the Motors Node, right click and Add Motor. Choose No Feedback (Step & Direction).

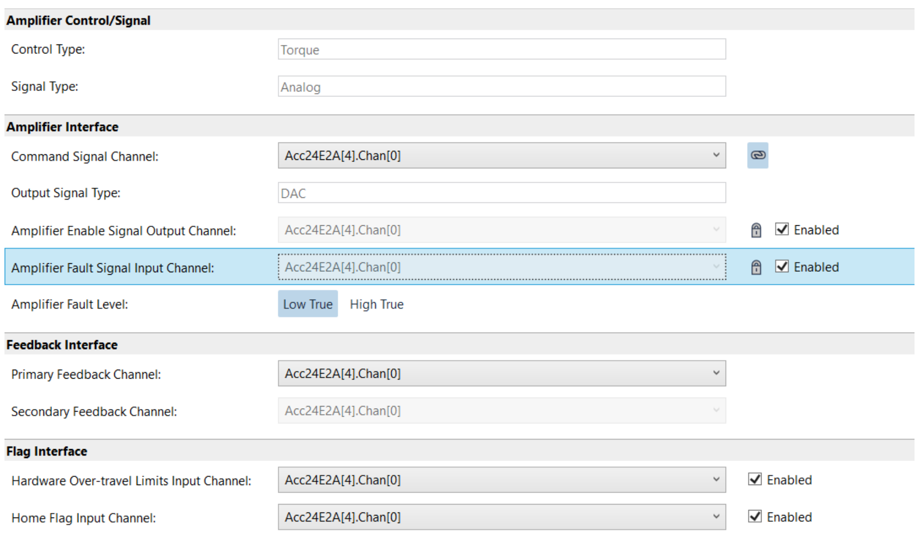
4. On successful addition of the Motor the Motor Toplogy View will be displayed. See the No Feedback topology Under Toplogy Section.
5. Click on the Amplifier page and add the amplifier that supports Pulse and direction. This is the first block in Topology view. The important Amplifier settings for this mode to work are shown below.



4.Pfm Mode	
Max Pulse Frequency (KHz)	**0**
Min Pulse Width	**50**
Pulse Width Units	**Duty Cycle**

The Max. Pulse Freq and Pulse width unit comes from the Amplifier manufacturer. These settings are enabled when the control type is Velocity control and signal type is Pulse and Direction from the amplifier page.

6. Click Motor block to select Stepper or to add Stepper motor.
7. Select Hardware Interface block for making Motor structure element connection. The hardware inerface page will loook like this. This is for Acc242A and for CK3M the connection will say CK3M[x].Chan[y]. Note Output Signal type.



8. Select PFM block to set PFM clock and pulse width.
9. The last step is commisioning to set Motor parameters like acceleration, decelaration etc.
10. Once all these steps are foloowed use Jog Ribbon to test the motor moving in both direction positive and negative

EtherCAT Network and Motor Setup

EtherCAT is supported when Power PMAC is ordered with the EtherCAT option. EtherCAT option on Power PMAC CPU (UMAC) comes with PCI Express accessory board plugged directly into the Power PMAC CPU.
CK3E always come with EtherCAT option and for CK3M EtherCAT is an option.

Note	All Power PMAC CPU support the Acontis stack from Firmware version 2.4 and above. CK3E supports Acontis stack for Firmware versions before 2.4

All the necessary hardware connections need to be setup, and if it is a drive, the separate configuration of the drive. This is typically by means of the drive manufacturer's software. The Power PMAC tuning utility can be used only if the EtherCAT drive is used in torque mode.

There are three steps in setting up EtherCAT devices

1. Setup EtherCAT network configuration
2. Load mappings to Power PMAC
3. If the EtherCAT device is an Amplifier, then add and configure the motor.

Note	Prior to configuring EtherCAT network it is necessary to setup EtherCAT Amplifier (Drive) used in Cyclic Synchronous Torque mode (CST) or Cyclic Synchronous Velocity mode(CSV) using vendor tool software.

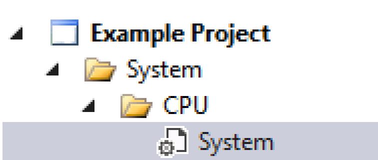
Step 1: Setup ECAT network configuration

Check and set Power PMAC Clock

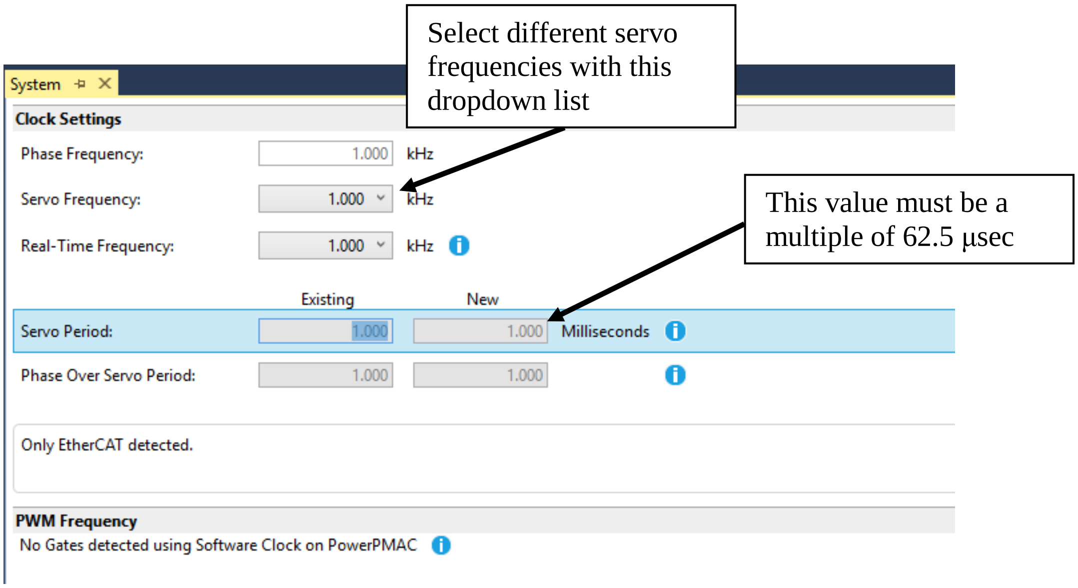
For all EtherCAT devices Power PMAC's servo frequency must be a multiple of 62.5 µsec.

Note	EtherCAT standard specifications can be found at http://ethercat.org/.

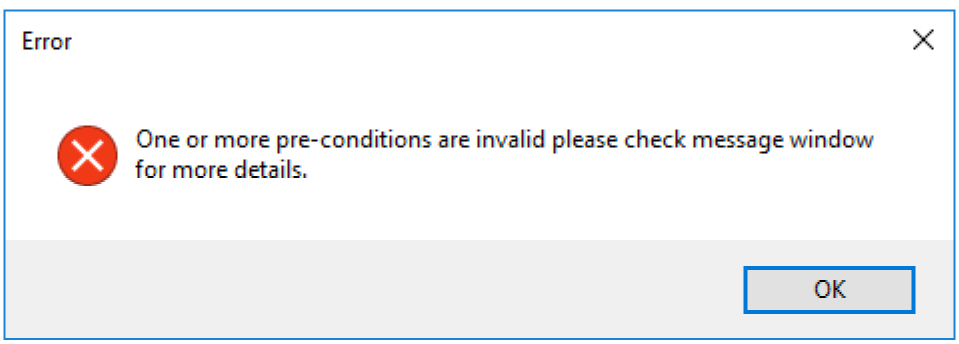
1. Open Power PMAC clock settings by clicking System – CPU folder or double-clicking the Power PMAC block on the Topology view, to open the Global Clock view



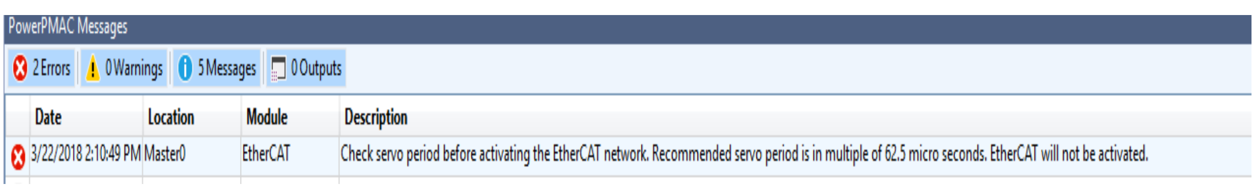
The required clock rate for the device should be defined in the device's manual. Most EtherCAT devices accept clock periods of 250 µsec, 500 µsec, and 1 msec. Set the Power PMAC servo clock frequency to one of the required frequencies as shown below:



If the frequency is not a multiple of 62.5 µsec then when the EtherCAT device is enabled by right-clicking on one of the Master nodes will generate an error. The error shown below will be displayed.

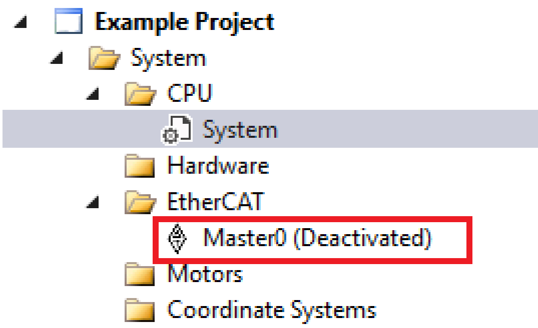


The details about the error are displayed in the Power PMAC message window as shown below:

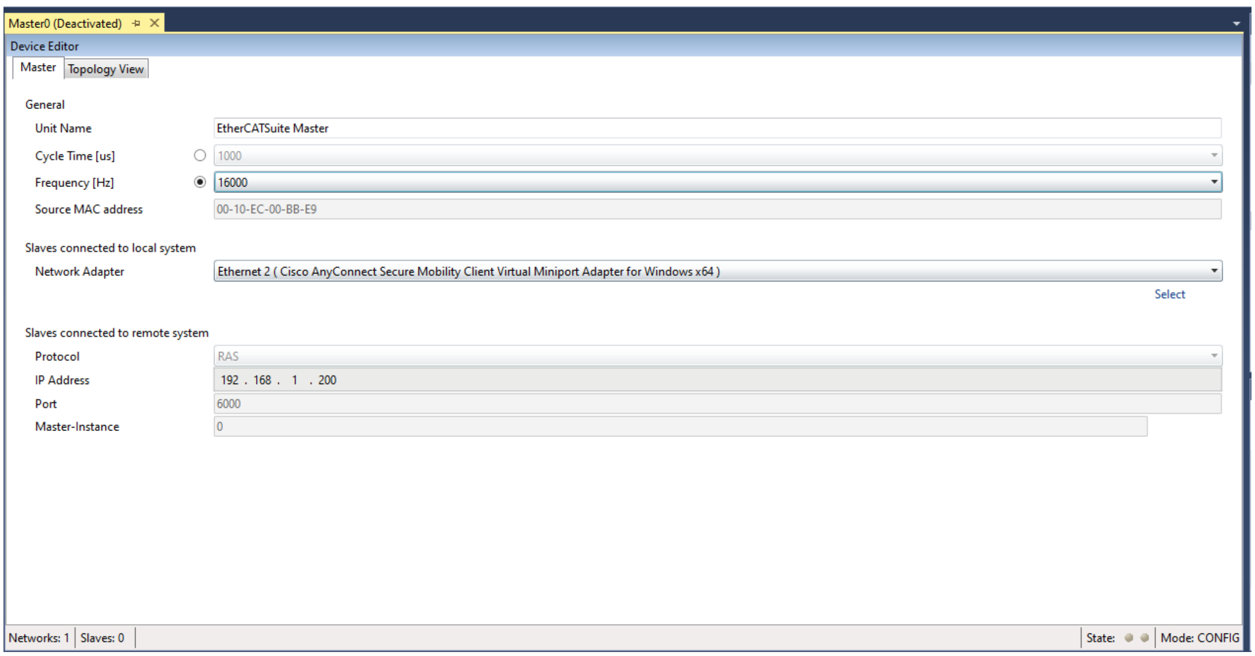


Configure the EtherCAT Device

Open the Master view by clicking the Master node under EtherCAT folder



This will open the Master view in the editor area.



Select a clock frequency that is the same as the Power PMAC servo clock frequency from Cycle Time element. User can choose to program clock either in time or in frequncy mode . If ECAT drive supports 16 KHz then user must select frewquency mode and then from drop down select 16000 Hz.

Note — Power PMAC servo clock and Cycle Time from Master must match.

The currently connected Power PMAC's IP address is displayed in the IPAddress field.

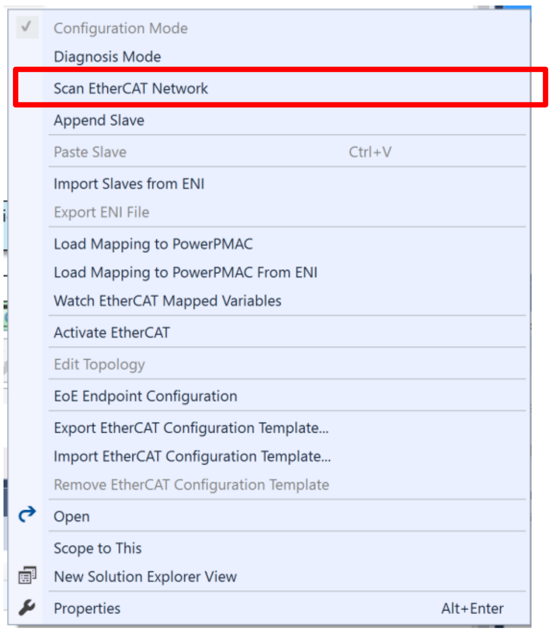
Appending or scanning the slave

To add a slave device to master:

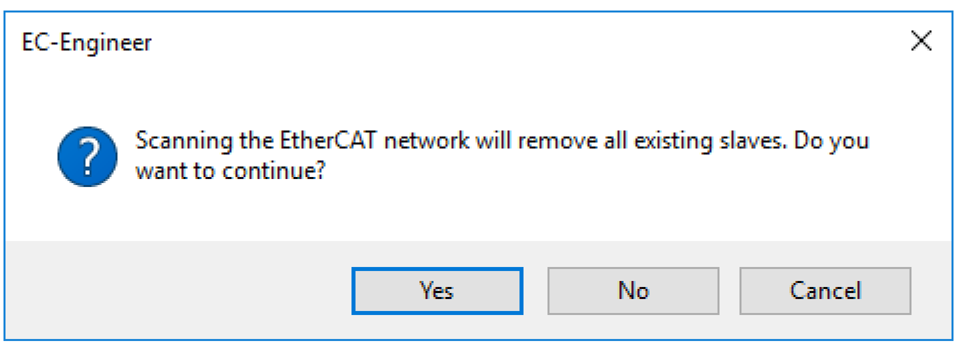
1. Scan the network if the devices are connected to the network
2. Append the slave from the list

Adding slave device to Master using Scan network

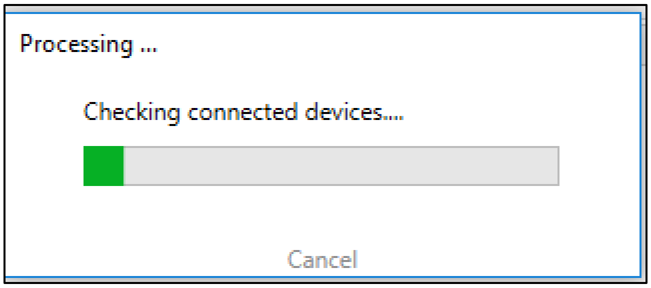
Right click on the Master node to open the context menu and select the Scan EtherCAT Network option:



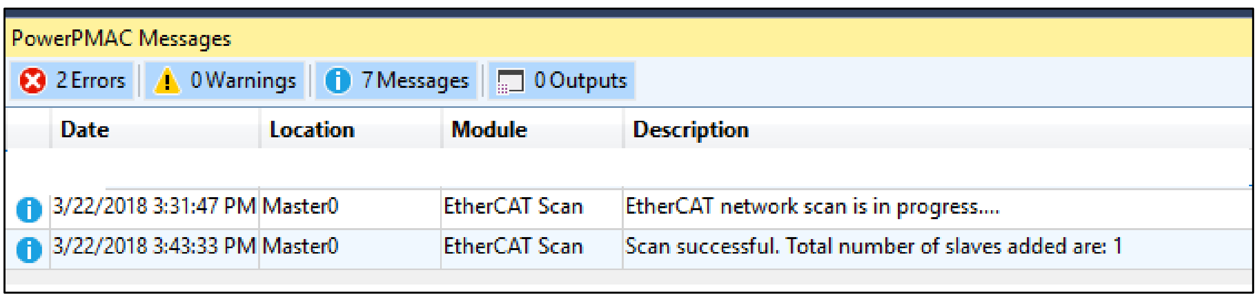
On selecting Scan EtherCAT Network the network scan will begin.
If there are devices already present under the Master node, permission will be requested to remove the nodes before scanning as shown below:



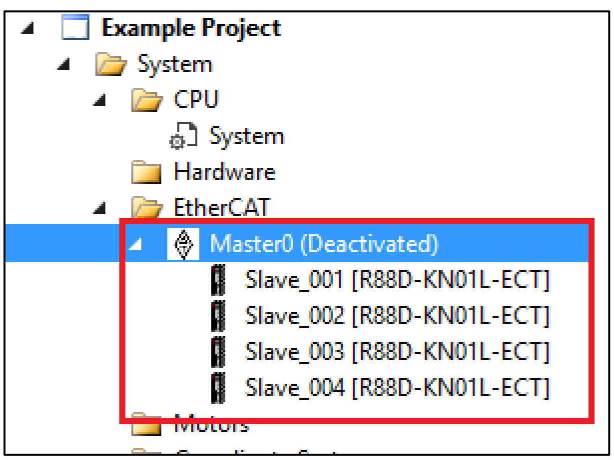
When there are no slave devices under Master node then scan will continue.



On completion of the scan a message will be displayed in the Power PMAC messages and the detected slave device/s will be added under the master node as shown below:

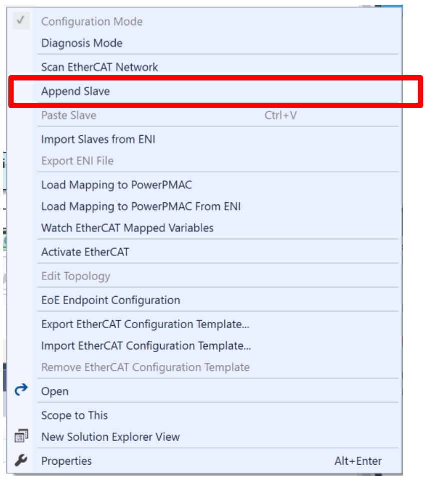


Slave devices will be added to the Master node as shown below:

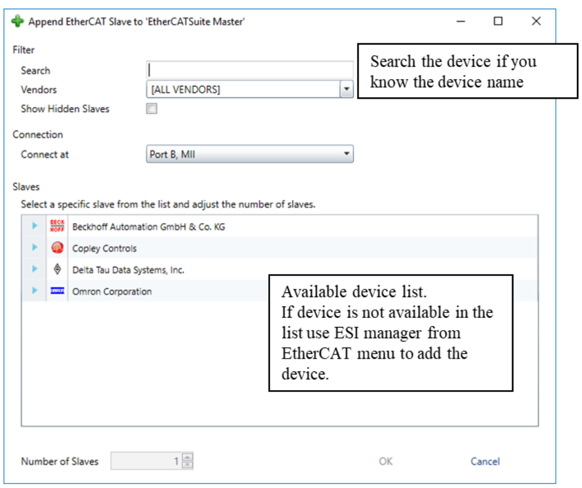


Adding Slave device to Master using Append Slave

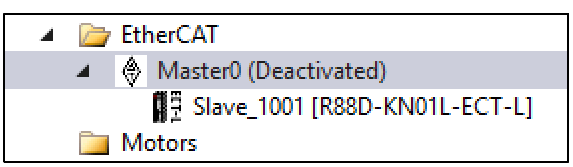
If there are no devices connected on the network, it is still possible to configure the EtherCAT network. In this case there is a Power PMAC but no EtherCAT device connected. Right click on the Master node and select the Append Slave option as shown below:



Selecting Append Slave will open the Append Slave dialog.

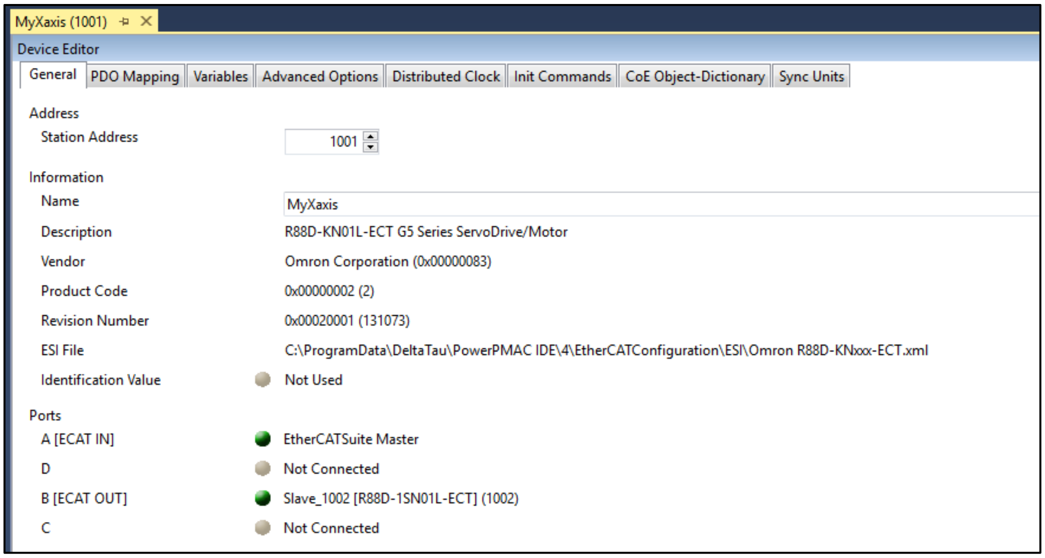


Choose the device to add. More than one slave can be added by using Number of Slaves counter. The default is 1. When a device is added a message will be displayed in the Power PMAC message box and the Slave will be added under the Master node. For example, as shown below the R88D-KNO1L-ECT-L Omron Drive slave device is appended to Master.



Naming Slave device

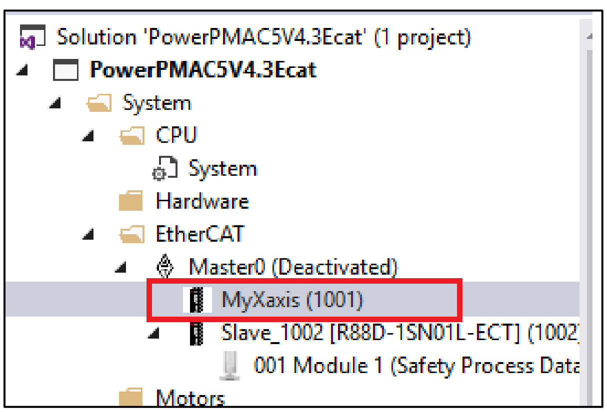
IDE V4.3 onwards supports naming the slave. User can open the slave dialog by clicking the slave from the project. Select General tab page and change the name. For example, in the following screen the slave name is change to MyXAxis.



As per our specification Slave name must start with alphabet character. Valid names are My_Axis, X_Axis, Y_Axis etc. Invalid names are 1_Xaxis, 234_Ypos, _12YAxis etc.

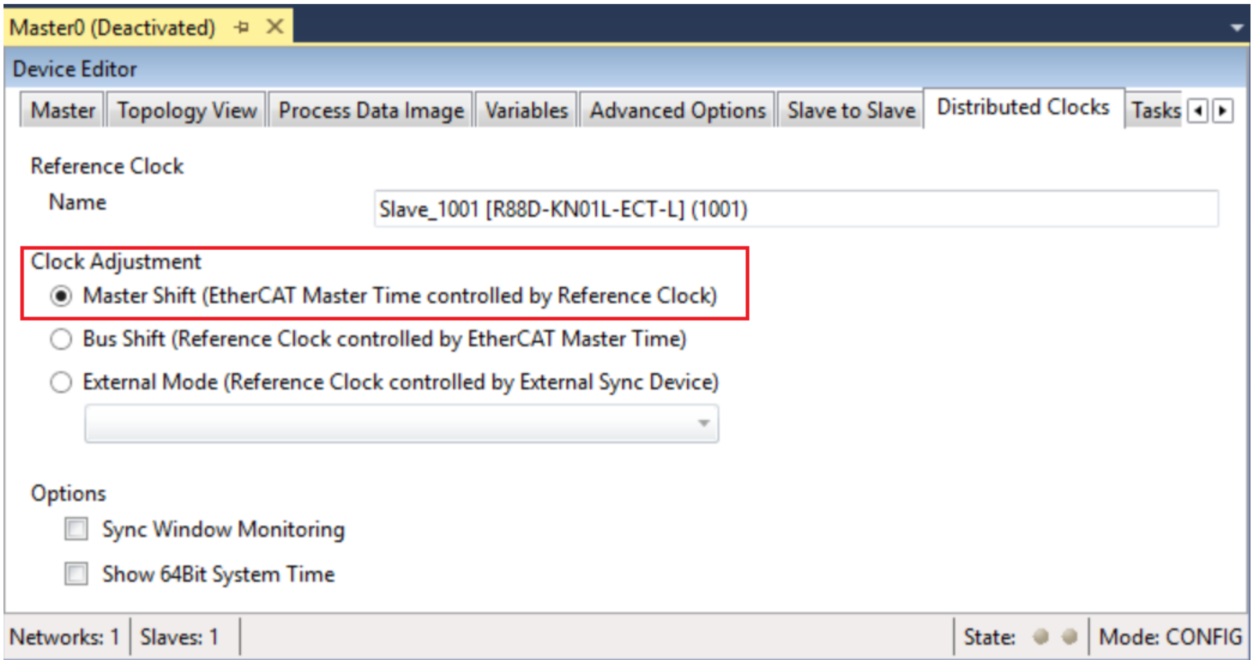
Note

As soon as the name change is successful project will be updated too.



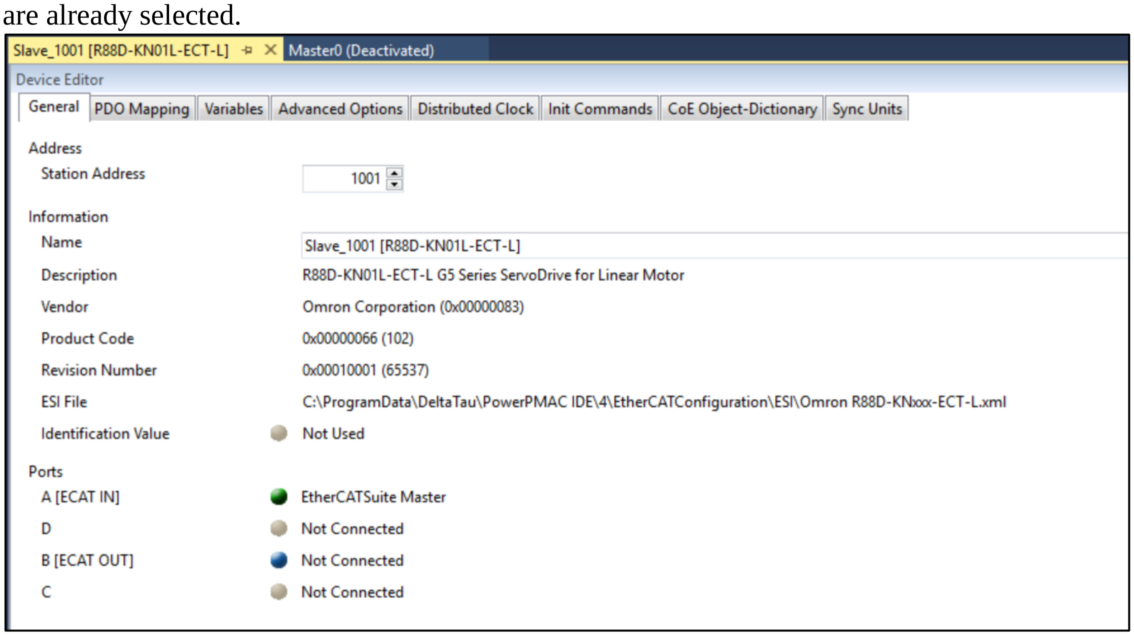
Configuring Slave device and Master device

Once the slave is available there are more Tab pages added dynamically to the Master view. For correct distributed clock operation make sure for Omron devices the distributed clock – Clock Adjustment is set to master shift mode as shown below:



The other settings from Master tab pages are rarely changed.

Click on the appended or scanned slave to open the slave device view. Most of the time the default PDOs are already selected.



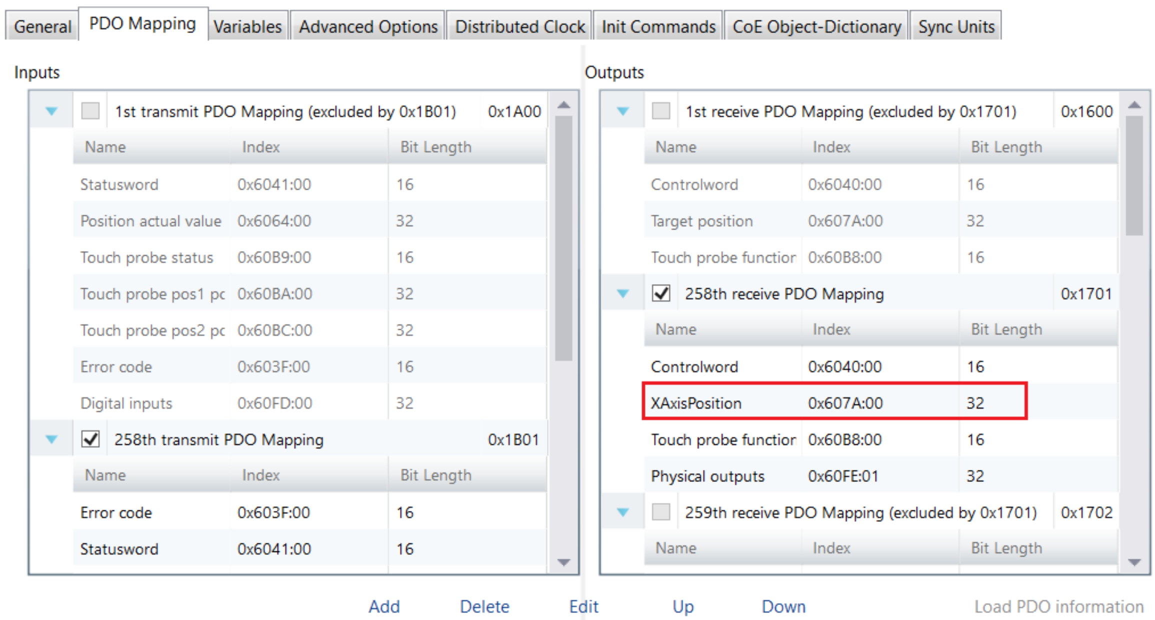
Configuring PDO mapping and renaming pdos

To change the PDO selection, select PDO mapping tab and choose new PDO.

Typically, the default PDO's are set as the settings are read from esi file.
This manual refers to 1S and G5 drives, thus the following PDO mappings are for these drives.

The majority of users will use CSP mode so most of the default PDO are set to choose mapping for CSP mode. If the User wants to change the type of control to CST or CSV, then a different set of PDO needs to be selected.

The User can edit the default name that are read from esi file. The newly named pdos are written to the header file on Load Pdo mapping context menu command.



The following describes choosing PDO for different types of control for 1S / G5 drive.

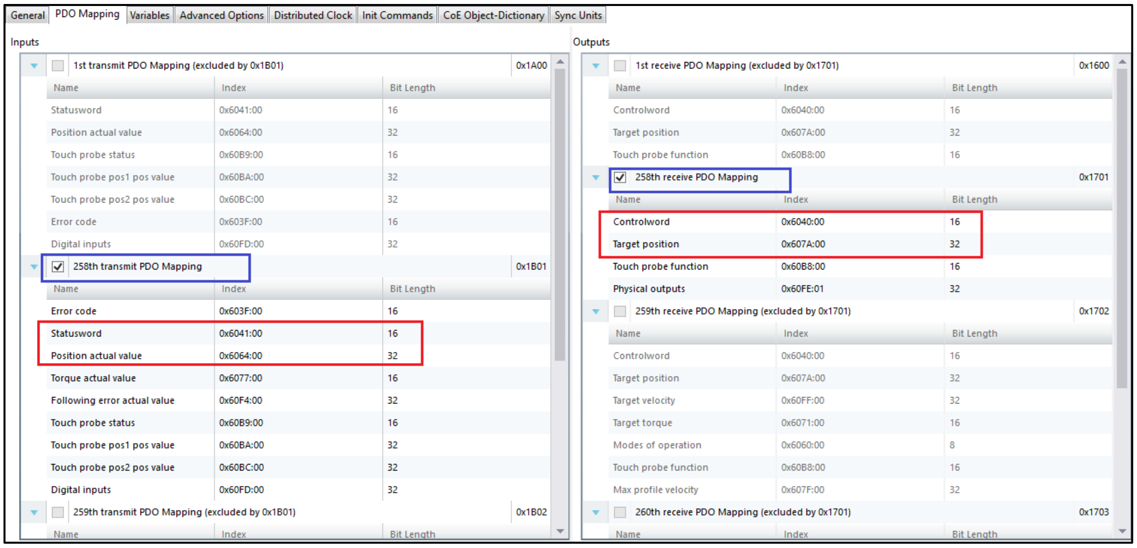
Note

For EtherCAT drives, other then OMRON (1S or G5), please refer the vendor manual for the correct set of PDO.

1S/G5 drive CSP (Cyclic Synchronous Profile/Position mode)

In the figure below the default PDO's are marked with a blue rectangle whereas the red rectangle marks Items that must have PDO for the selected cyclic mode.

The User needs to make sure for CSP mode that the PDO must have 0x6n7A where n is axis. So 0x607A defines mapping for axis 0.



Note

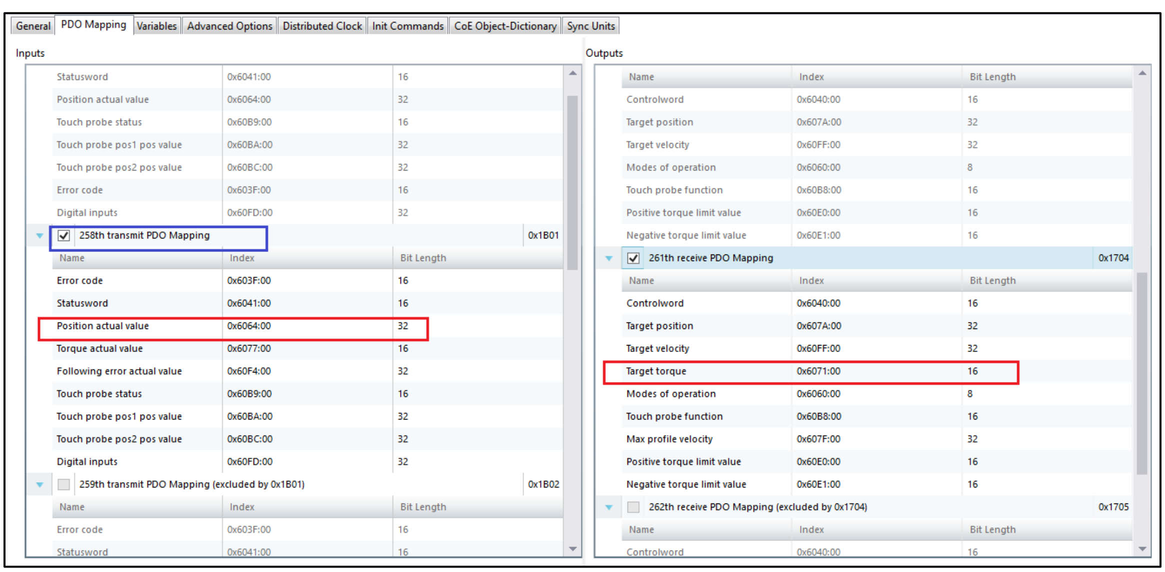
PDO name can be customize but make sure that customize name must be in English enven if the IDE is opened in the laguage other than English.

1S/G5 drive CST (Cyclic Synchronous Torque mode)

This is **not** default PDO's and the user has to unselect default PDO and select this PDO.

In the figure below the default PDO's are marked with a blue rectangle whereas the red rectangle marks Items that must have PDO for the selected cyclic mode.

The User needs to make sure for CST mode that the PDO must have 0x6n71 where n is axis. So 0x6071 defines mapping for axis 0.

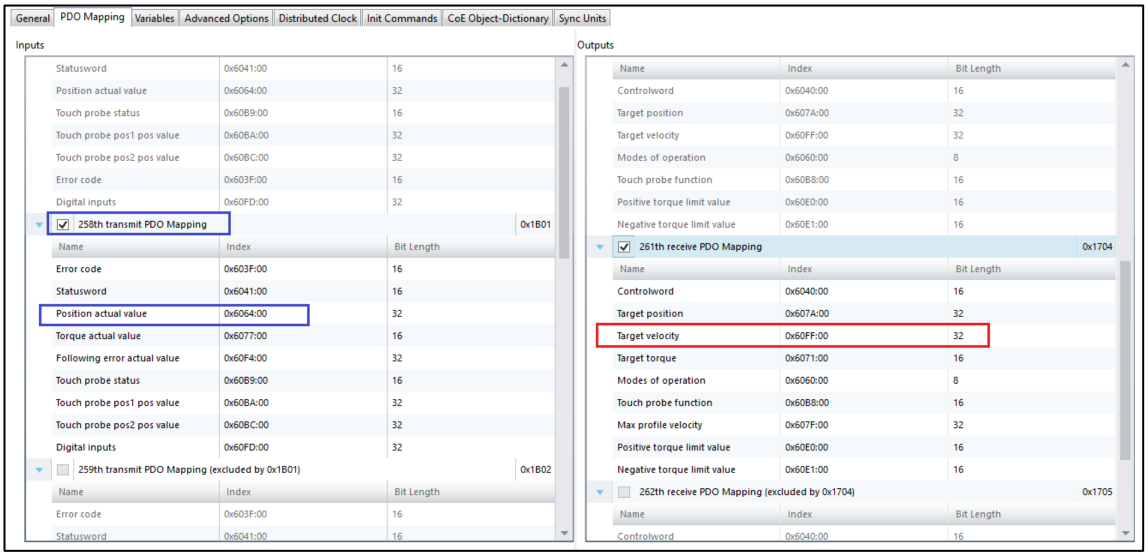


1S/G5 drive CSV (Cyclic Synchronous Velocity mode)

This is **not** default PDO's and user has to unselect default PDO and select this PDO.

In the figure below the default PDO's are marked with a blue rectangle whereas the red rectangle marks Items that must have PDO for the selected cyclic mode.

The User needs to make sure for CSV mode that the PDO must have 0x6nFF where n is axis. So 0x60FF defines mapping for axis 0.



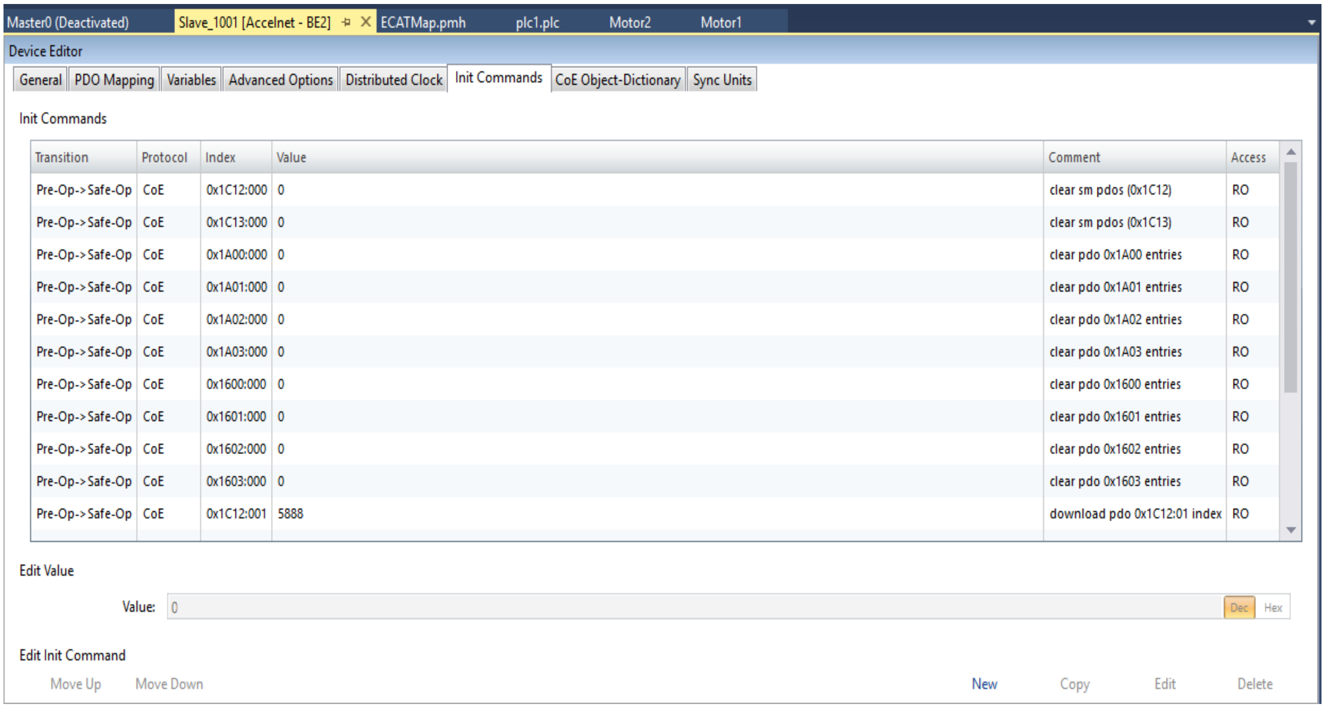
After selecting the appropriate PDO the next step in slave configuration to choose the Init commands.

Init Commands

In this tab the current configured init commands read from device esi file can be viewed and, if allowed, the add/edit/delete init commands can be used.

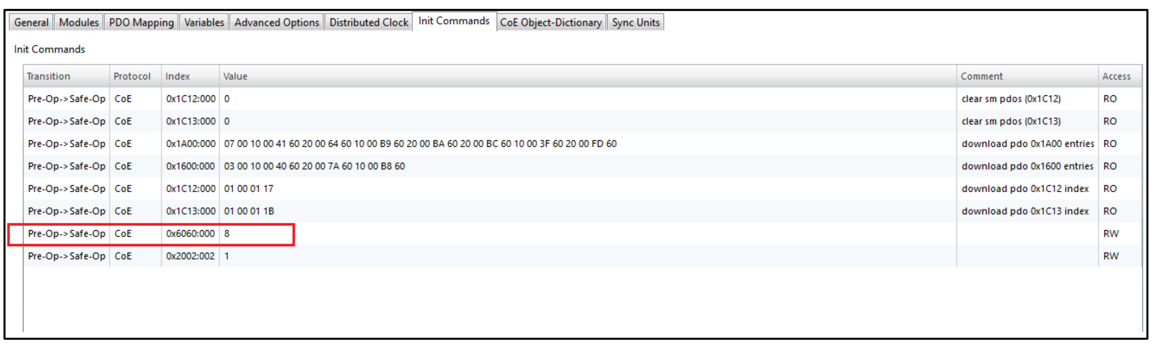
1. Lists of Init Commands
   - Init Commands comes from the ESI file or will be generated from the configurator. The "Access" column tells the user if this Init Command can be edited (RW = Read/Write) or not (RO = Read-Only).
2. Buttons
   - New/Copy/Edit/Delete: Used for changing the list
   - Up/Down: Moving the selected Init Command up or down

At the moment only Init Commands of the CoE- and SoE- Protocol can be added or changed.



Transition	Protocol	Index	Value	Comment	Access
Pre-Op->Safe-Op	CoE	0x1C12:000	0	clear sm pdos (0x1C12)	RO
Pre-Op->Safe-Op	CoE	0x1C13:000	0	clear sm pdos (0x1C13)	RO
Pre-Op->Safe-Op	CoE	0x1A00:000	0	clear pdo 0x1A00 entries	RO
Pre-Op->Safe-Op	CoE	0x1A01:000	0	clear pdo 0x1A01 entries	RO
Pre-Op->Safe-Op	CoE	0x1A02:000	0	clear pdo 0x1A02 entries	RO
Pre-Op->Safe-Op	CoE	0x1A03:000	0	clear pdo 0x1A03 entries	RO
Pre-Op->Safe-Op	CoE	0x1600:000	0	clear pdo 0x1600 entries	RO
Pre-Op->Safe-Op	CoE	0x1601:000	0	clear pdo 0x1601 entries	RO
Pre-Op->Safe-Op	CoE	0x1602:000	0	clear pdo 0x1602 entries	RO
Pre-Op->Safe-Op	CoE	0x1603:000	0	clear pdo 0x1603 entries	RO
Pre-Op->Safe-Op	CoE	0x1C12:001	5888	download pdo 0x1C12:01 index	RO

Just like default PDO are set to CSP mode the Init command matches the control mode and the default is CSP mode. This is marked by red rectangle below.



Transition	Protocol	Index	Value	Comment	Access
Pre-Op->Safe-Op	CoE	0x1C12:000	0	clear sm pdos (0x1C12)	RO
Pre-Op->Safe-Op	CoE	0x1C13:000	0	clear sm pdos (0x1C13)	RO
Pre-Op->Safe-Op	CoE	0x1A00:000	07 00 10 00 41 60 20 00 64 60 10 00 B9 60 20 00 BA 60 20 00 BC 60 10 00 3F 60 20 00 FD 60	download pdo 0x1A00 entries	RO
Pre-Op->Safe-Op	CoE	0x1600:000	03 00 10 00 40 60 20 00 7A 60 10 00 B8 60	download pdo 0x1600 entries	RO
Pre-Op->Safe-Op	CoE	0x1C12:000	01 00 01 17	download pdo 0x1C12 index	RO
Pre-Op->Safe-Op	CoE	0x1C13:000	01 00 01 1B	download pdo 0x1C13 index	RO
Pre-Op->Safe-Op	CoE	0x6060:000	8		RW
Pre-Op->Safe-Op	CoE	0x2002:002	1		RW

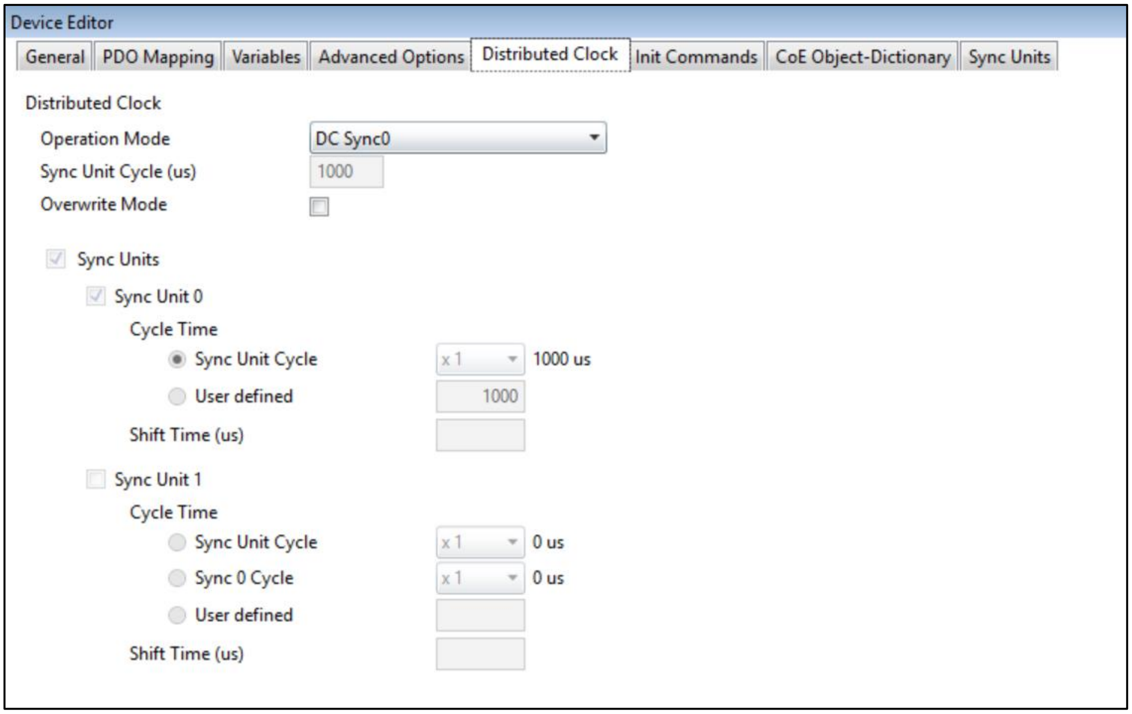
Object 6060h: Modes of operation is set to 8 for CSP mode. Follow the table for correct operation mode and edit the value for object 6060h.

Note

To use control type other than CSP object 6060h must be change to appropriate mode.

+8	Cyclic sync position mode
+9	Cyclic sync velocity mode
+10	Cyclic sync torque mode

Distributed Clock



- Operation Mode: Selectable DC operation modes. The modes cannot be edited.
- Sync Unit Cycle: Base interval in microseconds which will be used from the master
- Overwrite Mode: Overwrites the settings of the selected operation mode (might be necessary, if the slave doesn’t offer the right operation mode)

Sync Units

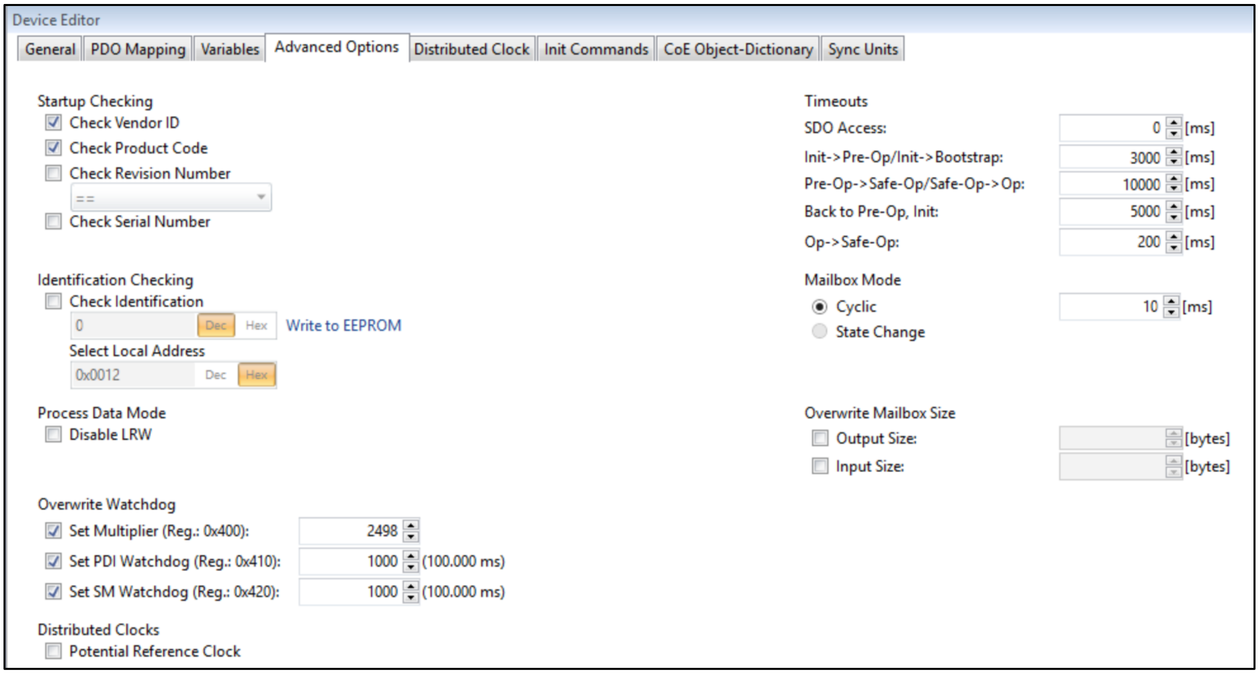
▪ Sync Unit 0
  - Cycle Time
    - Sync Unit Cycle: Unit is synchronized relative to the Unit Cycle
    - User defined: Unit has its own interval
    - Shift Time: Unit is adjusted by the shift time. Typically, one half or one quarter of the EtherCAT cycle time works for most devices. For example, if the clock is 1 msec (1 kHz) then shift time of 250usec or 500usec will work for most devices.

Refer to the Device manual for guidelines on the exact shift time.

Note

Advanced settings

For the slave to be a potential reference clock then select the Advanced Options Tab to make the changes.



1. Startup Checking
   - Master will check the Vendor ID, Product code and Revision number if the state machine changes from INIT to PREOP of the slave
   - Revision number can be verified six ways:
     - "==" HI word is equal, LO word is equal
     - ">=" HI word is equal or greater, LO word is equal or greater
     - "LW ==" HI word is equal
     - "LW ==, HW >=" LO word is equal, HI word is equal or greater
     - "HW ==" LO word is equal
     - "HW ==, LW >=" HI word is equal, LO word is equal or greater

2. Identification Checking
   - If 'Check Identification' is selected the Identification Value of the slave is checked. The 'Select Local Address' Box is the register of the Identification Value.

3. Process Data Mode
   - Disable LRW: Determines whether LRD/LWR command or the LRW command is used for accessing process data. Cable redundancy needs LRD/LWR, slave-to-slave-copy needs LRW.

4. Watchdog
   - Set Multiplier: Writes the configured value to the corresponding slave register: 0x0400
   - Set PDI Watchdog: Writes the configured value to the corresponding slave register: 0x0410

5. Distributed Clocks

- Potential Reference Clock: Set to use the slave as a potential reference clock

6. Timeouts
   - SDO Access: Internal master timeout which is used for accessing the SDO (0 = Use internal default value of the master)
   - Init PreOp: Internal master timeout which is used for changing slave state.
   - Pre-Op Save-Op or Safe-Op Op: Internal master timeout which is used for changing slave state.
   - Back to Pre-Op, Init: Internal master timeout which is used for changing slave state.
   - Op Safe-Op: Internal master timeout which is used for changing slave state.

7. Overwrite Mailbox Size
   - Output Size: Overwrites mailbox output size.
   - Input Size: Overwrites mailbox input size.

8. Mailbox Mode
   - Cyclic: Interval in milliseconds within which the input mailbox will be read (polling mode)
   - State Change: The input mailbox will be read only if the status bit is set

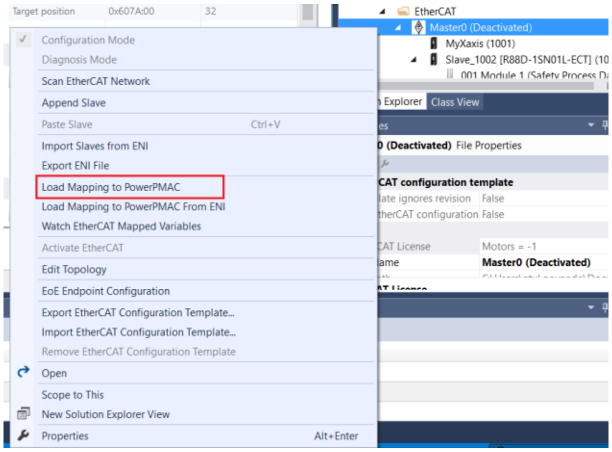
Continue setting the PDO and distributed clock setting for rest of the slave devices in the network.

Step 2: Load mappings to Power PMAC

This step is necessary for the EtherCAT network to function properly. Without this step the EtherCAT motor setup will not work properly.

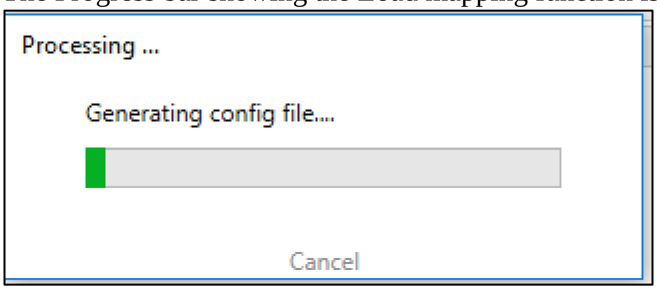
It is expected that the user has completed configuring the necessary settings for the Master device and the Slave devices for the network.

Right click on master node to select Load mappings to Power PMAC option, as shown below.

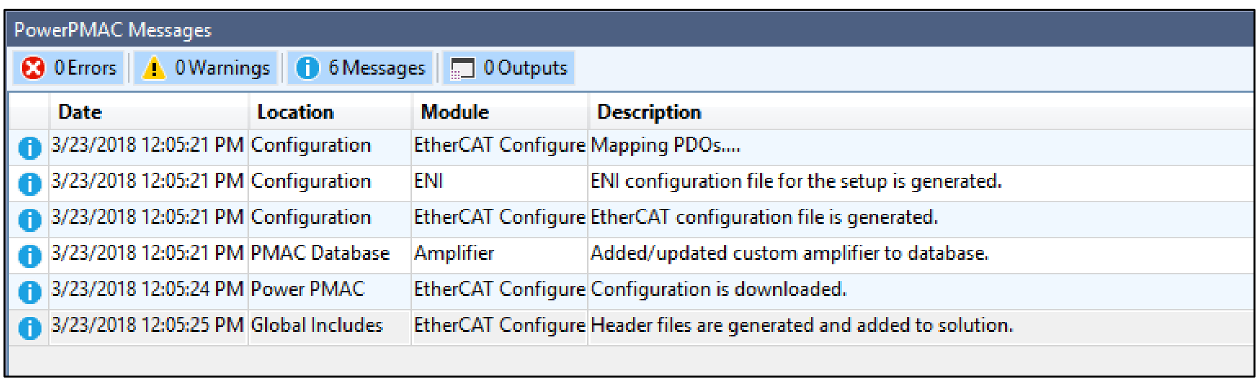


On selecting Load mapping to Power PMAC, the process indicates it's progress by showing a dialog and a message in the Power PMAC message box.

The Progress bar showing the Load mapping function is shown below



The Progress status messages are shown below.



PowerPMAC Messages

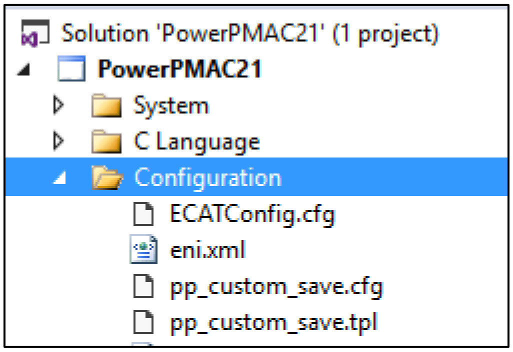
0 Errors | 0 Warnings | 6 Messages | 0 Outputs

Date	Location	Module	Description
3/23/2018 12:05:21 PM	Configuration	EtherCAT Configure	Mapping PDOs....
3/23/2018 12:05:21 PM	Configuration	ENI	ENI configuration file for the setup is generated.
3/23/2018 12:05:21 PM	Configuration	EtherCAT Configure	EtherCAT configuration file is generated.
3/23/2018 12:05:21 PM	PMAC Database	Amplifier	Added/updated custom amplifier to database.
3/23/2018 12:05:24 PM	Power PMAC	EtherCAT Configure	Configuration is downloaded.
3/23/2018 12:05:25 PM	Global Includes	EtherCAT Configure	Header files are generated and added to solution.

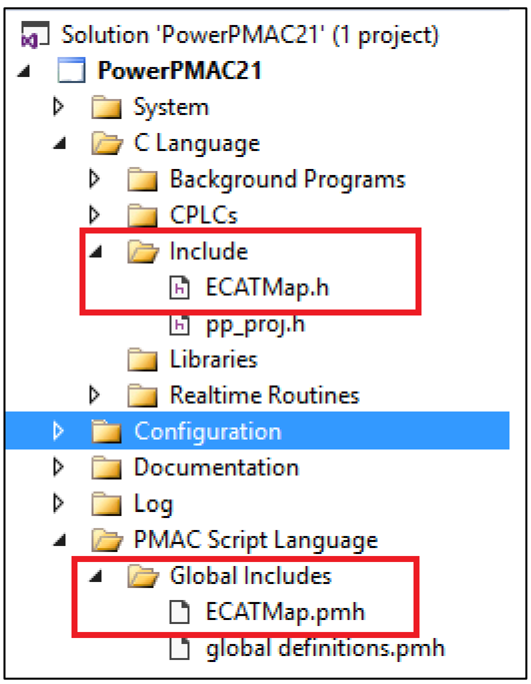
On successful completion of the Load mapping to Power PMAC the following actions are completed.

4. The eni.xml (EtherCAT network information) is generated and copied to the Project-Configuration folder and then downloads the file to the Power PMAC /var/ftp/usrflash/Project/Configuration folder.

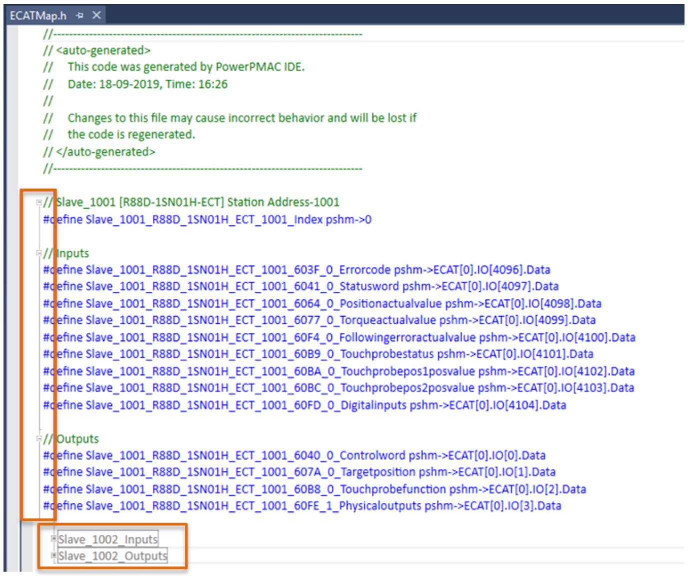
5. The mapping file ECATConfig.cfg is created and copied to the Project-Configuration folder and downloaded to the Power PMAC /var/ftp/usrflash/Project/Configuration folder. After downloading, the file is loaded to Power PMAC using gpascii –iECATConfig.cfg command.



6. The ECATMap.pmh and ECATMap.h files are created and copied to the Power PMAC Script Language-Global Includes and C Language-Includes folders for use in C app and script languages. These header files consist of #defines values to access ECAT mappings in C app or script languages.



A Header file for script looks like this:



```
ECATMap.h
//-------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by PowerPMAC IDE.
//     Date: 18-09-2019, Time: 16:26
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//-------------------------------------------------------------------------------

// Slave_1001 [R88D-1SN01H-ECT] Station Address-1001
#define Slave_1001_R88D_1SN01H_ECT_1001_Index pshm->0

// Inputs
#define Slave_1001_R88D_1SN01H_ECT_1001_603F_0_Errorcode pshm->ECAT[0].IO[4096].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_6041_0_Statusword pshm->ECAT[0].IO[4097].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_6064_0_Positionactualvalue pshm->ECAT[0].IO[4098].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_6077_0_Torqueactualvalue pshm->ECAT[0].IO[4099].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_60F4_0_Followingerroractualvalue pshm->ECAT[0].IO[4100].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_60B9_0_Touchprobestatus pshm->ECAT[0].IO[4101].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_60BA_0_Touchprobepos1posvalue pshm->ECAT[0].IO[4102].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_60BC_0_Touchprobepos2posvalue pshm->ECAT[0].IO[4103].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_60FD_0_Digitalinputs pshm->ECAT[0].IO[4104].Data

// Outputs
#define Slave_1001_R88D_1SN01H_ECT_1001_6040_0_Controlword pshm->ECAT[0].IO[0].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_607A_0_Targetposition pshm->ECAT[0].IO[1].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_60B8_0_Touchprobefunction pshm->ECAT[0].IO[2].Data
#define Slave_1001_R88D_1SN01H_ECT_1001_60FE_1_Physicaloutputs pshm->ECAT[0].IO[3].Data

Slave_1002_Inputs
Slave_1002_Outputs
```

PDO names constructed with the slave name and pdo name. The user has the ability to collapse/expand EtherCAT slaves mappings marked with // <Slave Name> and shown above with an orange Square.
This is explained above in step 1.

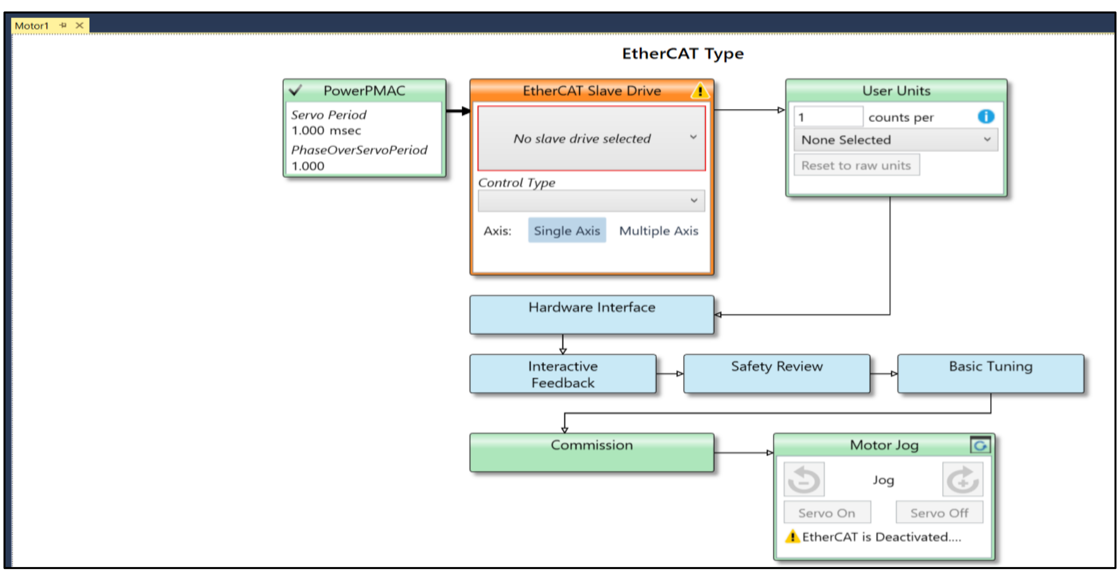
Note

3. It is not necessary to copy the EtherCAT files manually to the project like in V2.x and V3.x; V4.x automatically manages these files.
4. EtherCAT header files collapse/expand feature is available in te IDE 4.3.2.x and above

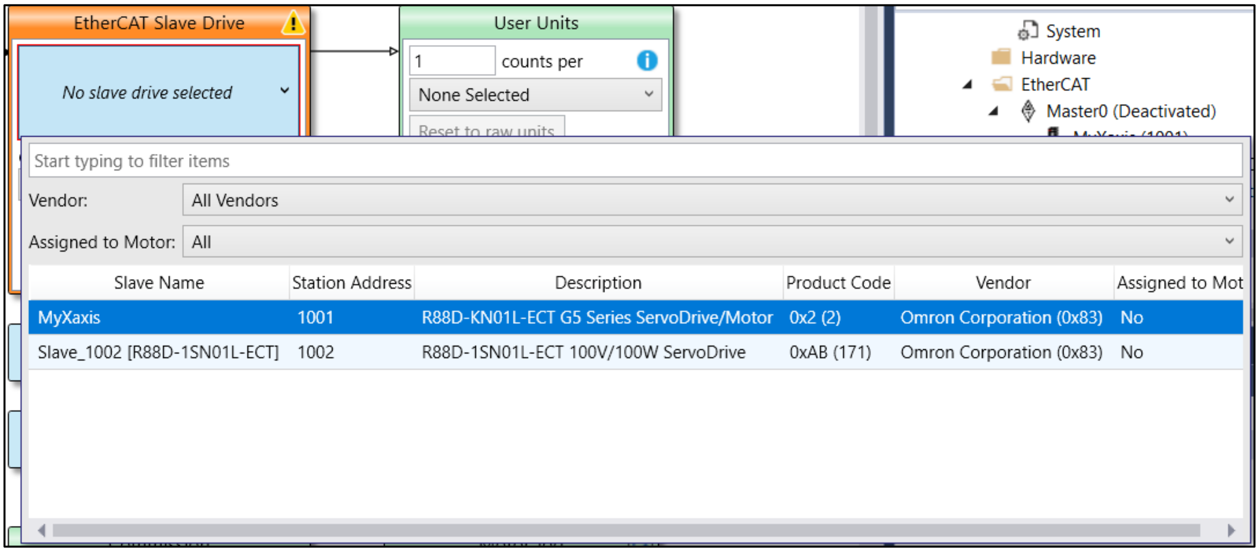
---

Step 3: Add EtherCAT Motor (Method 1)

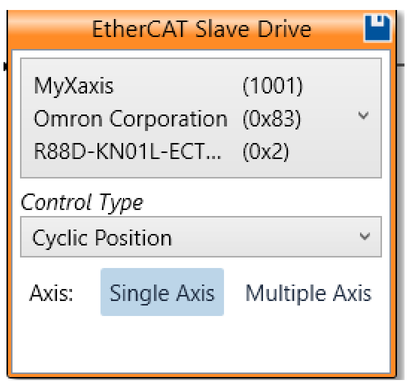
Go to the Motors node in the project and right click Add Motor and select EtherCAT Topology.



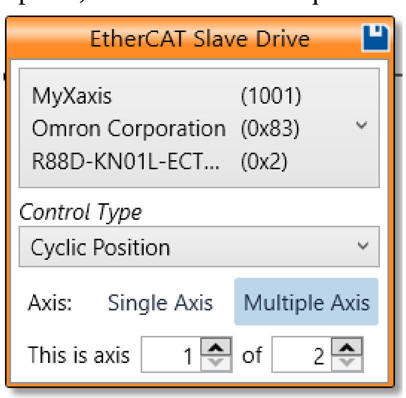
The user needs to select a slave drive in the orange colored block, EtherCAT Slave Drive. The drop-down list automatically populates with all the available slave drives from the Project-EtherCAT master node, as shown below…



The list shows the all the details about the slaves, including whether the slave is already assigned to a motor.
Select the appropriate slave from the list and enter the control type. Once selected, the EtherCAT Slave Drive block will look like this…

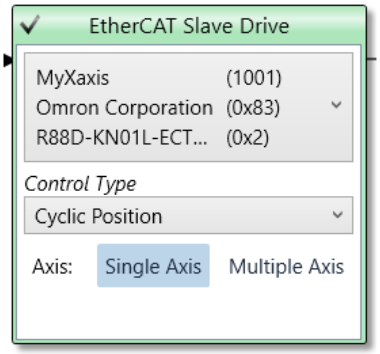


If the slave drive has more than one axis, then choose Multiple Axis. On selecting the Multiple Axis option, the user will be required to enter the axis number as shown below…



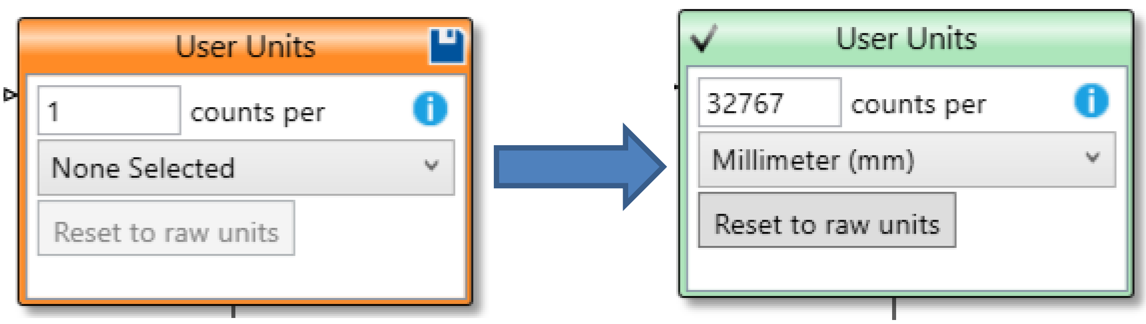
Entering appropriate values in the EtherCAT Slave Drive block will automatically load the data in the Hardware Interface page from the available mappings.

On entering the correct settings press the  symbol to save the changes. On success, the color of the Amplifier Block will change to green with a check mark as shown below:



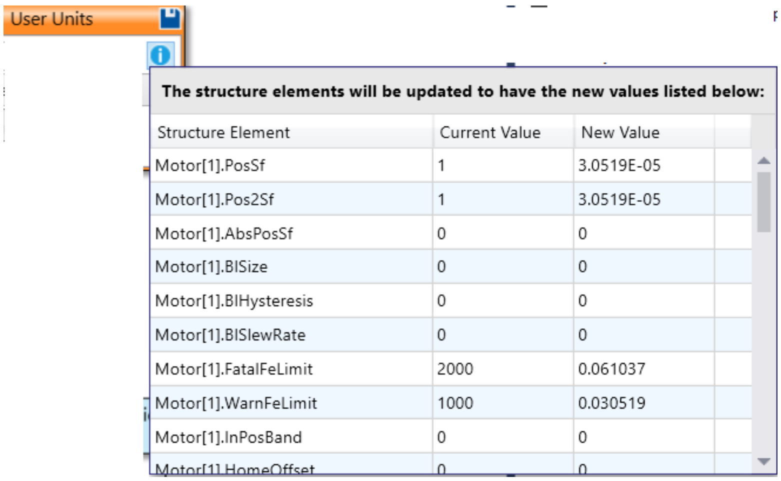
The next block to configure will be the User Units.

This is not a mandatory block and it is entirely the user's choice to define how many counts correspond to a machine unit. For example, on a machine that needs 32767 counts to move 1 mm due to its mechanism, then the user would enter the following…



On pressing the save symbol , the block will set all the necessary Power PMAC structure elements to reflect the User Unit change so the user can program in the User Units (i.e. mm in the previous example).

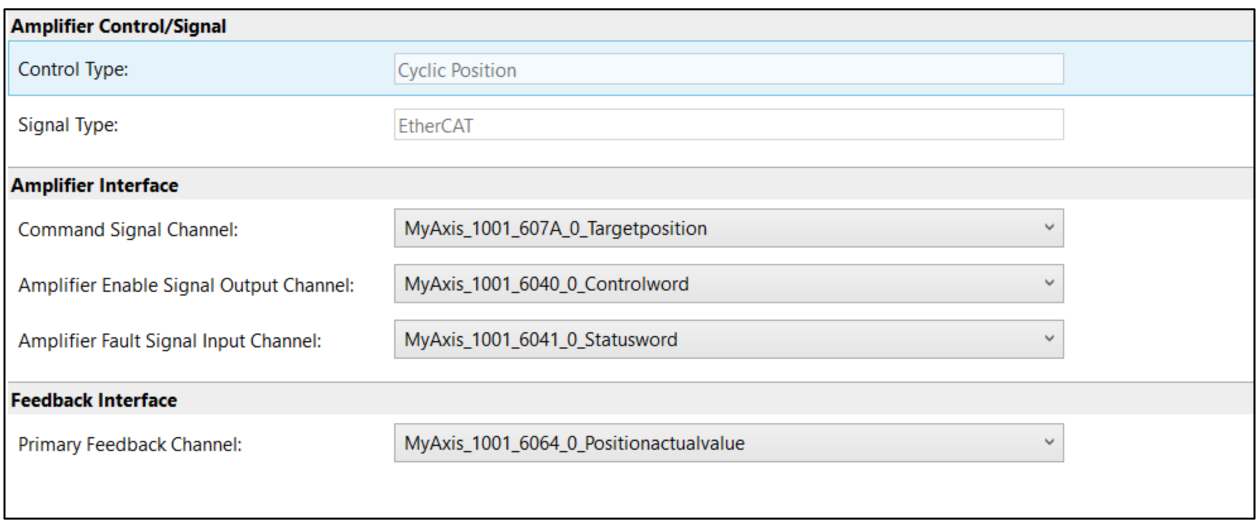
Click on the information icon to check the affected structure elements. The view will look like this…



Structure Element	Current Value	New Value
Motor[1].PosSf	1	3.0519E-05
Motor[1].Pos2Sf	1	3.0519E-05
Motor[1].AbsPosSf	0	0
Motor[1].BlSize	0	0
Motor[1].BlHysteresis	0	0
Motor[1].BlSlewRate	0	0
Motor[1].FatalFeLimit	2000	0.061037
Motor[1].WarnFeLimit	1000	0.030519
Motor[1].InPosBand	0	0
Motor[1].HomeOffset	0	0

The next block to configure is the Hardware Interface block.

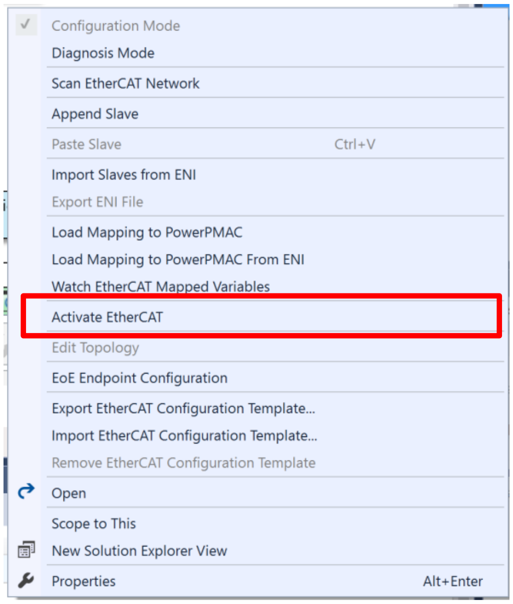
This will associate the EtherCAT connection with the Power PMAC motor and encoder structures. If the slave values are set correctly in the Amplifier Block, then the Hardware Interface Block will populate with the correct connection. Verify the entries and press Accept and the Hardware Interface block will be marked as complete on the topology view. The Hardware Interface screen is shown below:



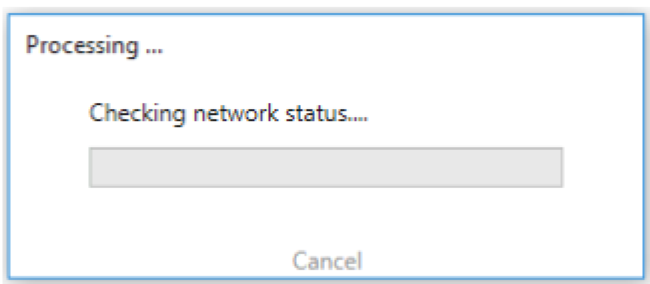
The selected items are for Cyclic Position mode.

The next block to configure is the Interactive Feedback block to test the encoder feedback. To do this the EtherCAT need to be activated.

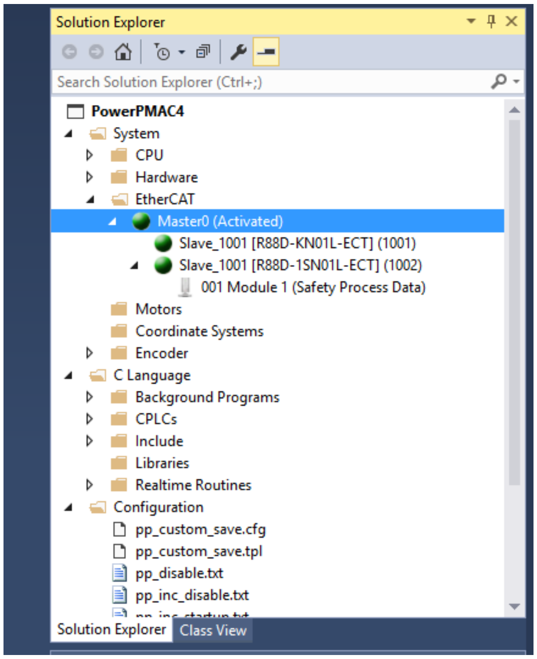
Right click on the Master Node and click on Activate EtherCAT as shown below:

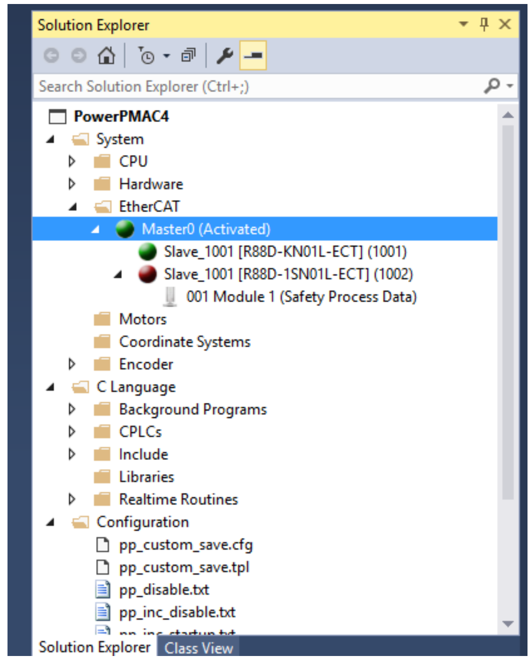


If the Activation fails, the reason for this will be displayed in the Power PMAC message box. While Activating, status will indicate the progress as shown below:



On a Successful Activation the Master Node will display “Activated” as shown below:





The Green circle icon indicates the EtherCAT is activated.
The Red circle indicates the Slave device is deactivated.

On successful activation the User can verify the encoder feedback my moving the motor by hand (if possible)

The next two blocks are for Safety review and Basic Tuning.

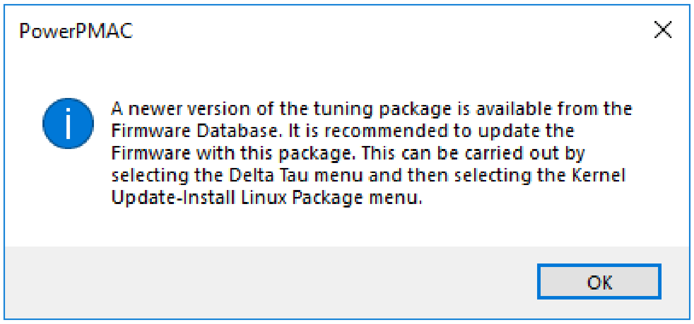
For the Cyclic Position mode these two blocks are not required, and the User can move forward to the Commissioning block and Motor Jog.

If the Control type is Cyclic Torque or Cyclic Velocity, then the User follow safety review Basic tuning block.

Note

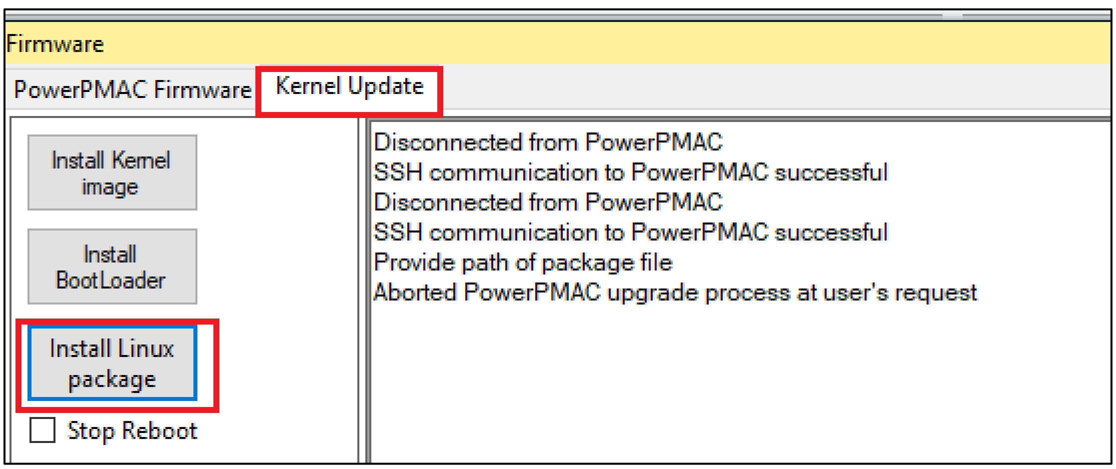
Safety Review and Basic Tuning Toplogy blocks are enabled only if the Control type is Cyclic Torque or Cyclic Velocity.

If the Control type is either Torque or velocity, then selecting the Basic Tuning block will generate the warning if user is using the FW 2.5.1.7 without a new Tuning package as shown below…



It is not mandatory to upgrade the tuning package, but the User does not then they will not get the benefit of improvements in the tuning and setup algorithms.

If the User wants to upgrade the tuning package, they can download this from the Delta Tau Firmware location and use the Update Firmware dialog from the Delta Tau menu and select Kernel Update- Install Linux Package like this…



Once the package is updated then the User can use the Basic tuning block to tune the Torque or velocity mode and on success proceed to Commissioning and Motor Jog Block to test the motor.

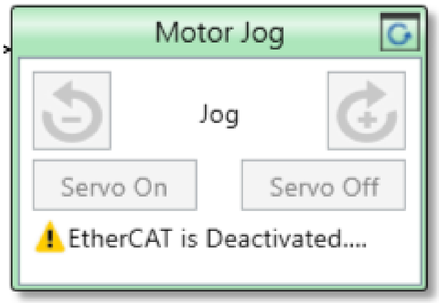
Note

The User only needs to install the Tuning package once. For any following set up's the Warning message will not be displayed.

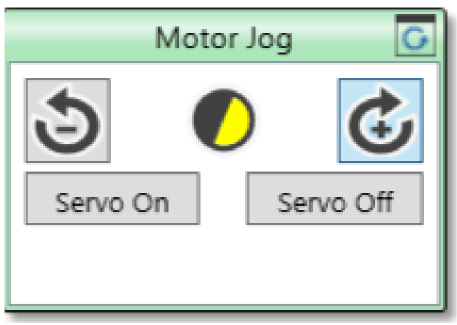
When all the necessary Topology blocks are Green the User can test the EtherCAT Motor using Motor Jog block.

This is a simple Jog block for testing the Motor settings. For any advance Jog functionality, the User can click on the Jog block to open the Jog Ribbon Menu.

If the EtherCAT is not Activated, then the User cannot Jog the motor and it will be indicated on the Motor Jog Block.



If the EtherCAT is Activated, then the User can Jog the motor and the movement will be indicated on the Motor Jog Block by the rotating of the circular icon .



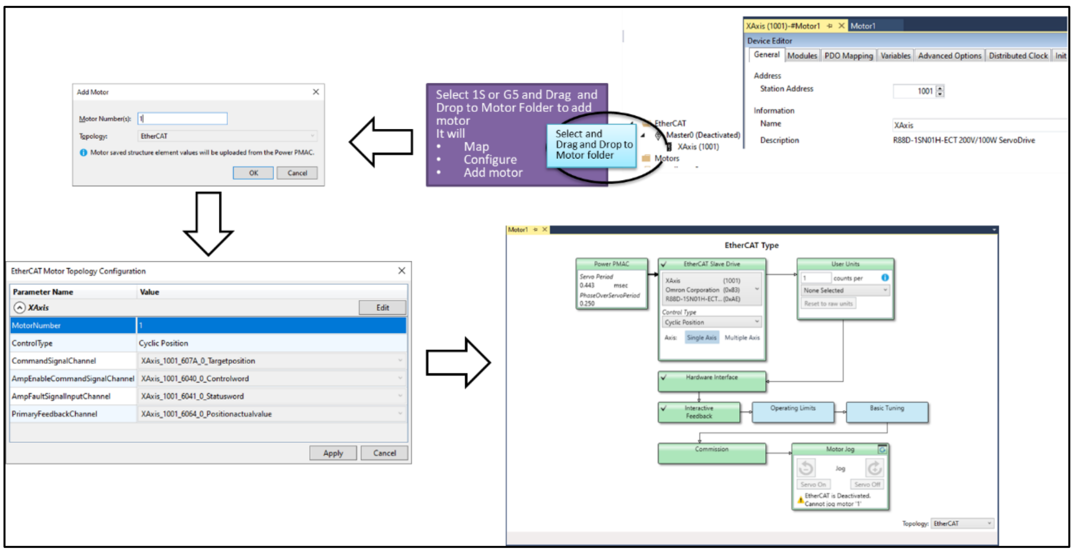
The User can Servo ON and Servo OFF the axis. These commands are basically “#<Motor Number> Jog” and “#<Motor Number> Kill”.
At this point for the Cyclic Position mode EtherCAT Motor setup is complete.

Step 3: Add EtherCAT Motor (Method 2-Drag and Drop)

This method is only available for our EtherCAT devices (OMRON-1S or G5). If you are using multiple EtherCAT drive either 1S or G5 then it possible to use Drag and Drop method. You can either select one OMRON EtherCAT drive or Multiple OMRON EtherCAT drive. Once select Drag and Drop on to Motor folder and setup system will configure the motor based on the type of PDO mapping. This is best used with Cyclic Position mode, as default PDO configuration is set for cyclic position in EtherCAT drive.

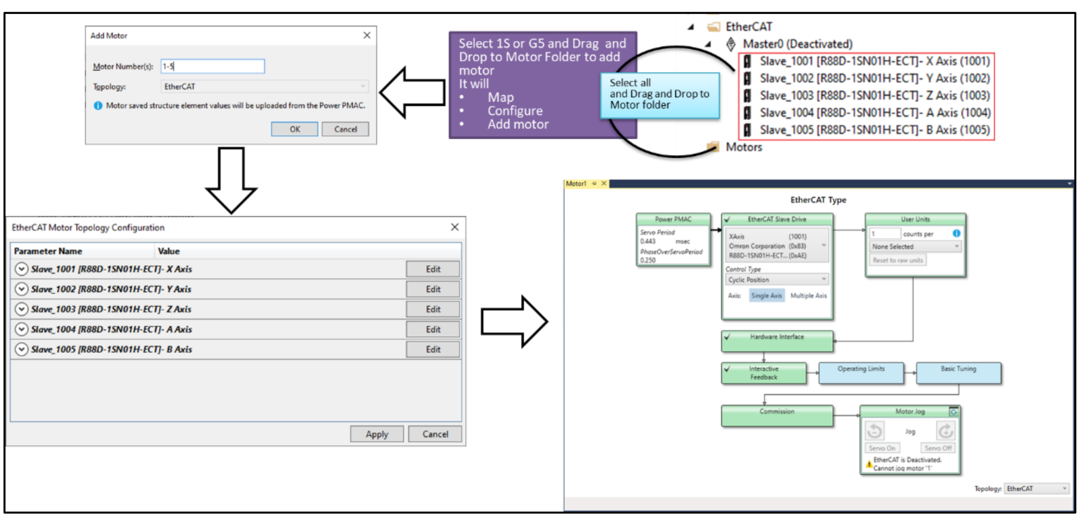
Single EtherCAT drive Drag and Drop:

Following picture shows the flow for setup. On success user will need to Enable the EtherCAT network and Jog the Motor and verify the setup. This is for Cyclic Position mode. For Cyclic Torque Basic Tuning is needed.

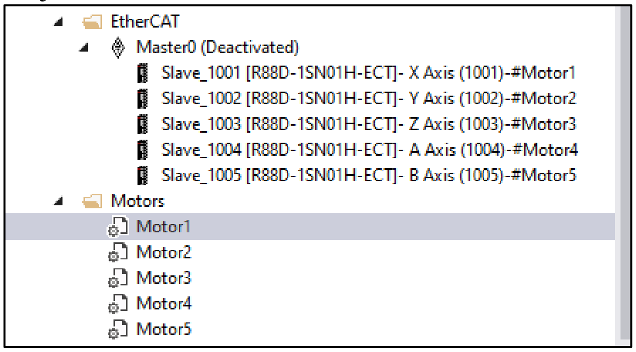


Multiple EtherCAT drive Drag and Drop:

Following picture shows the flow for setup. On success user will need to Enable the EtherCAT network and Jog the Motor and verify the setup. This is for Cyclic Position mode. For Cyclic Torque Basic Tuning is needed.



Project tree will look like this…

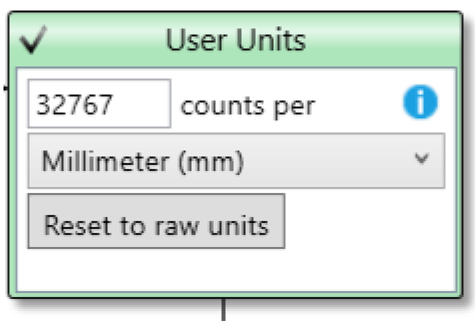


Note

1. Only OMRON EtherCAT slave drives support drag and drop.
2. Drag and Drop requires initial PDO mapping for type of Cyclic control. Default is cyclic position.

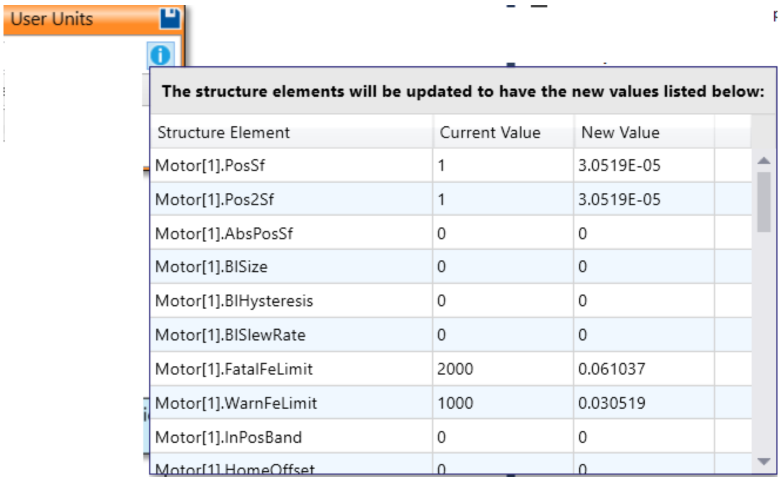
Once user uses one of the Drag and Drop it is possible to user other blocks as explained below. The next possible block the user can configure will be the User Units.

This is not a mandatory block and it is entirely the user's choice to define how many counts corresponds to a machine unit. For example, on a machine that needs 32767 counts to move 1 mm due to its mechanism, then the user would enter the following…



On pressing the save symbol , the block will set all the necessary Power PMAC structure elements to reflect the User Unit change so the user can program in the User Units (i.e. mm in this example).

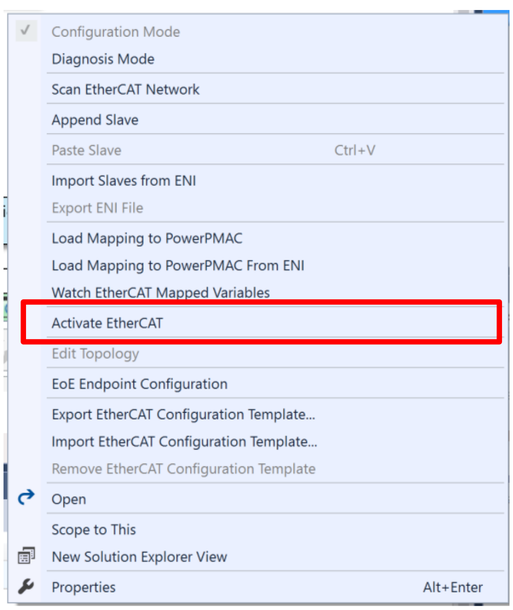
Click on the Information icon to check the affected structure elements. The view will look like this…



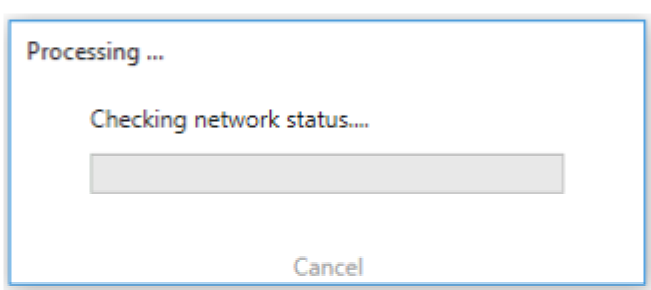
Structure Element	Current Value	New Value
Motor[1].PosSf	1	3.0519E-05
Motor[1].Pos2Sf	1	3.0519E-05
Motor[1].AbsPosSf	0	0
Motor[1].BlSize	0	0
Motor[1].BlHysteresis	0	0
Motor[1].BlSlewRate	0	0
Motor[1].FatalFeLimit	2000	0.061037
Motor[1].WarnFeLimit	1000	0.030519
Motor[1].InPosBand	0	0
Motor[1].HomeOffset	0	0

EtherCAT need to be activated.

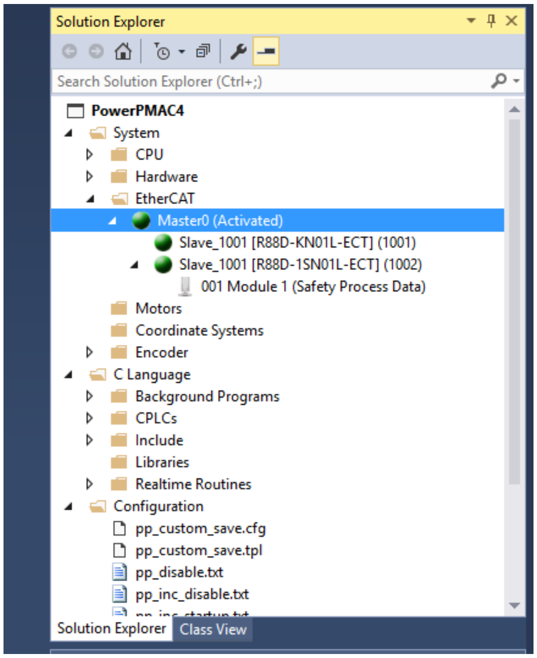
To activate EtherCAT, right-click on the Master Node to open the context menu and click Activate EtherCAT as shown below:

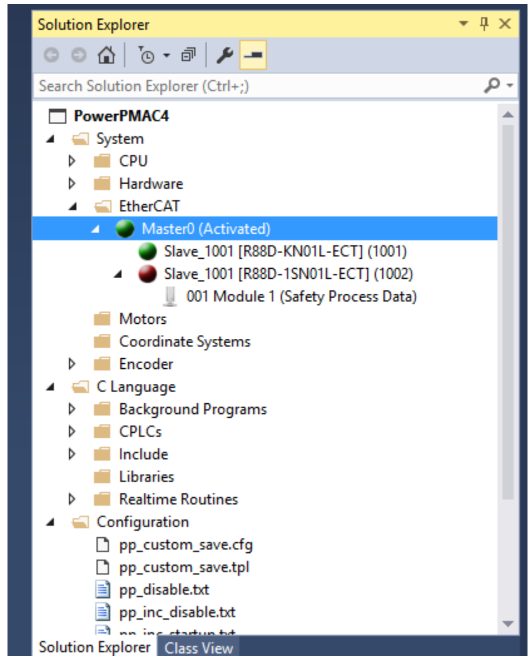


If the activation fails, the reason for this will be displayed in the Power PMAC message box. While activating a progress dialog will indicate the progress, along with the status, as shown below:



On a successful activation the Master Node will display "Activated" as shown below:





A green circle icon indicates that EtherCAT is activated for the slave.
A red circle indicates the slave device is deactivated.

On successful activation the user can verify the encoder feedback my moving the motor by hand (if possible).

The next two blocks are for Safety review and Basic Tuning.

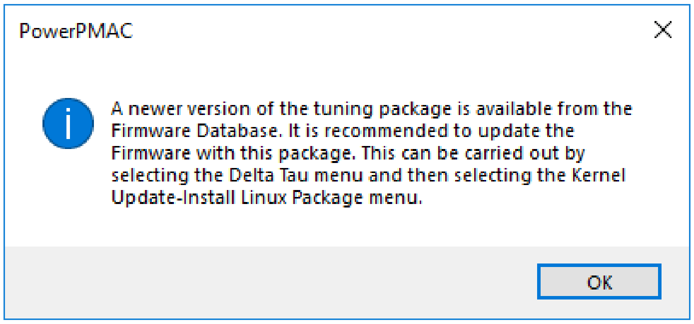
For the Cyclic Position mode these two blocks are not required, and the user can move forward to the Commissioning block and Motor Jog.

If the Control Type is Cyclic Torque or Cyclic Velocity, then the next item for the user to configure will be the Safety Review and Basic Tuning blocks.

Note

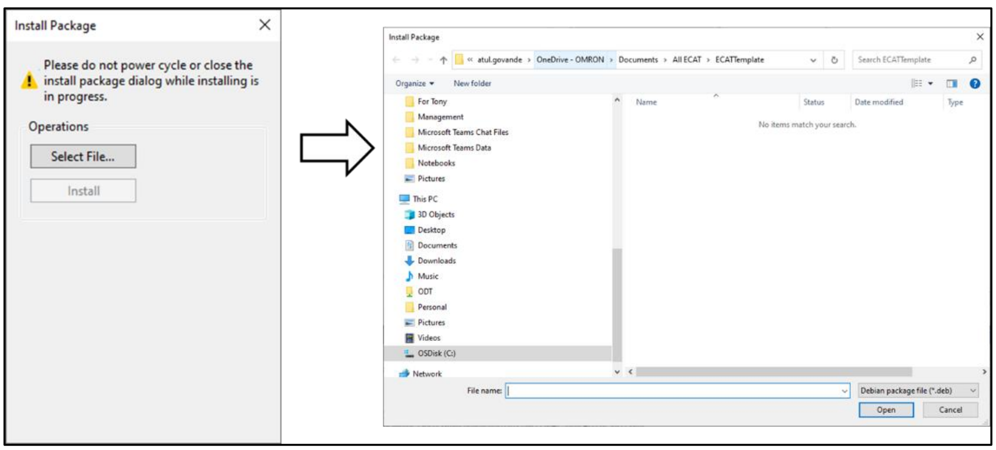
Safety Review and Basic Tuning Toplogy blocks are enabled only if the Control Type is Cyclic Torque or Cyclic Velocity.

If the Control Type is either Torque or Velocity, then selecting the Basic Tuning block will generate the warning if the user is using FW 2.5.1.7 without a new tuning package as shown below…



Upgrading the tuning package is not mandatory, but if the user doesn’t then they will not get the benefit of improvements in the tuning and setup algorithms.

If the user wants to upgrade the tuning package they can download this from the Delta Tau Firmware location, then use the Update tuning package dialog from the Delta Tau menu and select Install package It looks like this… enter the appropriate debian package file (.deb) and press Install.



Once the package is updated, the user can use the Basic Tuning block to tune the Torque or Velocity mode and on success proceed to Commissioning and Motor Jog blocks to test the motor.

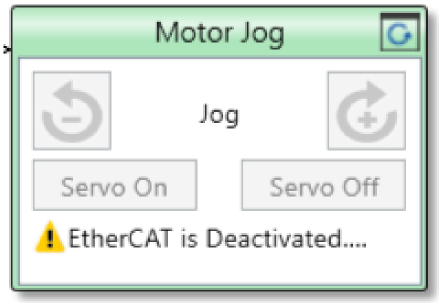

Note

The user only needs to install the Tuning package once. For any following setups the warning will not be displayed.

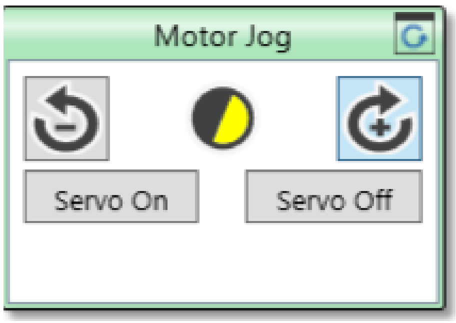
When all the necessary topology blocks are green the user can test the EtherCAT motor using the Motor Jog block.

This is a simple jog block for testing the motor settings. For any advanced jog functionality, the user can click on the jog block to open the Jog Ribbon menu.

If EtherCAT is not activated, then the user cannot jog the motor and it will be indicated on the Motor Jog block.



If EtherCAT is activated, then the user can jog the motor and the movement will be indicated on the Motor Jog block by the rotation of the circular icon .

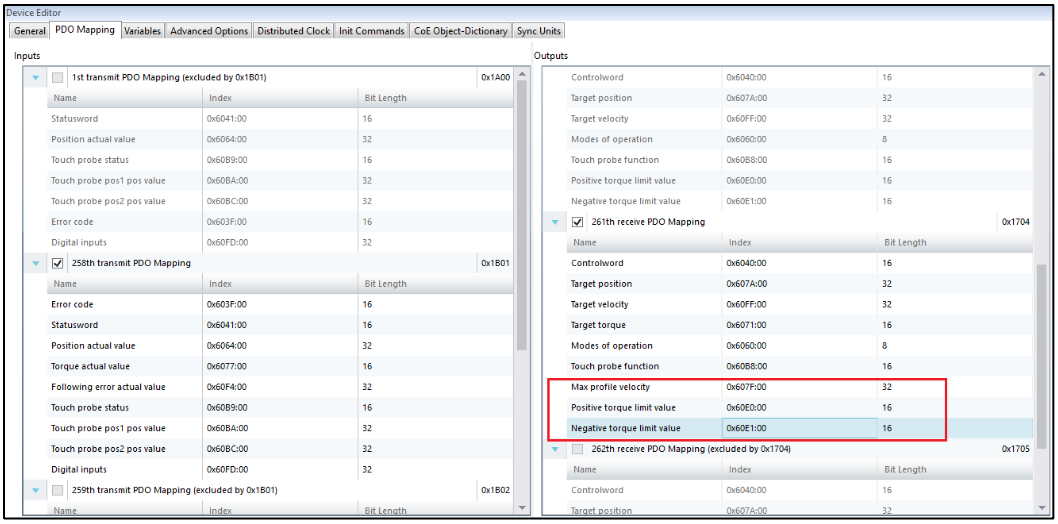


The user can Servo ON and Servo OFF the axis. These commands are equivalent to “#<Motor Number> Jog/” and “#<Motor Number> Kill”.

At this point, if configuring for Cyclic Position mode, EtherCAT Motor setup is complete.

Additional necessary settings for 1S and G5 drive to be used in CST and CSV mode

To use 1S or G5 in CST or CSV mode the user needs to change the settings on objects marked by the red rectangle below.



After completing PDO mapping steps the mapping will be available in the .pmh files under the Global Includes folder of the project. For example:

Slave_0_607F_0_Maxprofilevelocit (or ECAT[0].IO[6].Data)
Slave_0_60E0_0_Positivetorquelim (or ECAT[0].IO[7].Data)
Slave_0_60E1_0_Negativetorquelim (or ECAT[0].IO[8].Data)

The user can write to this object from the terminal window. These values can be changed even after the network is activated. The details on these settings can be found in the 1S or G5 drive manual. In our test we have set these values as start point to …

Slave_0_607F_0_Maxprofilevelocit (or ECAT[0].IO[6].Data) = This is dependent on encoder resolution. For 1S the resolution is 2^17 bit so for the nominal rpm of 3000 the minimum value of 2^17 * 50 user can change this as necessary.
Slave_0_60E0_0_Positivetorquelim (or ECAT[0].IO[7].Data) = 5000
Slave_0_60E1_0_Negativetorquelim (or ECAT[0].IO[8].Data) = 5000

Note

The User is advised to set these values appropriately for 1S and G5 as per the requirement and referring to the device manual.

Note

Additional settings must be set in order for 1S and G5 drive to work in CST or CSV mode. For drive's other than 1S and G5 similar settings are needed.
Please check vendor manual for these settings.

The motor can now be set to Jog by either typing the following command in terminal window, “#<n>J/” where n is motor number or the Jog Ribbon number, or by using Jog Ribbon.

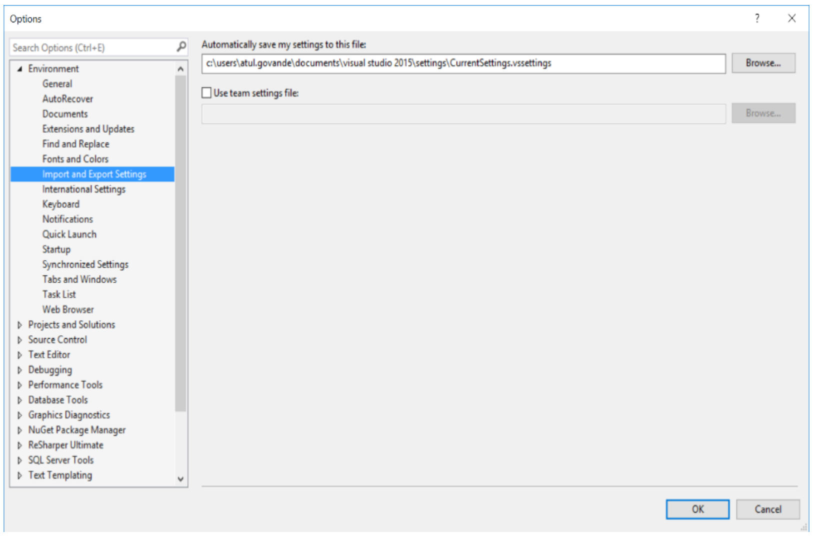
MISCELLANEOUS FEATURES OF THE IDE

There are various features available within the Visual Studio based Power PMAC IDE.

Import/Export Settings

This feature allows the layout of the Power PMAC IDE to be changed from the default and saved.

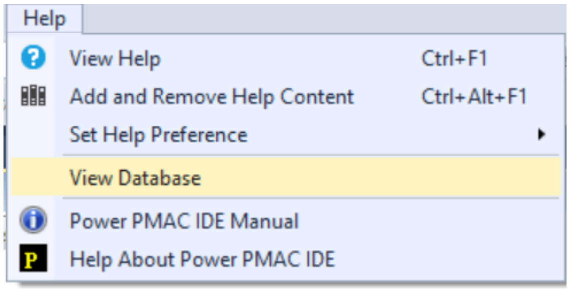
This can be accessed using Tools-Option-Import and Export Settings. Use the location path and name to store a personal IDE layout.



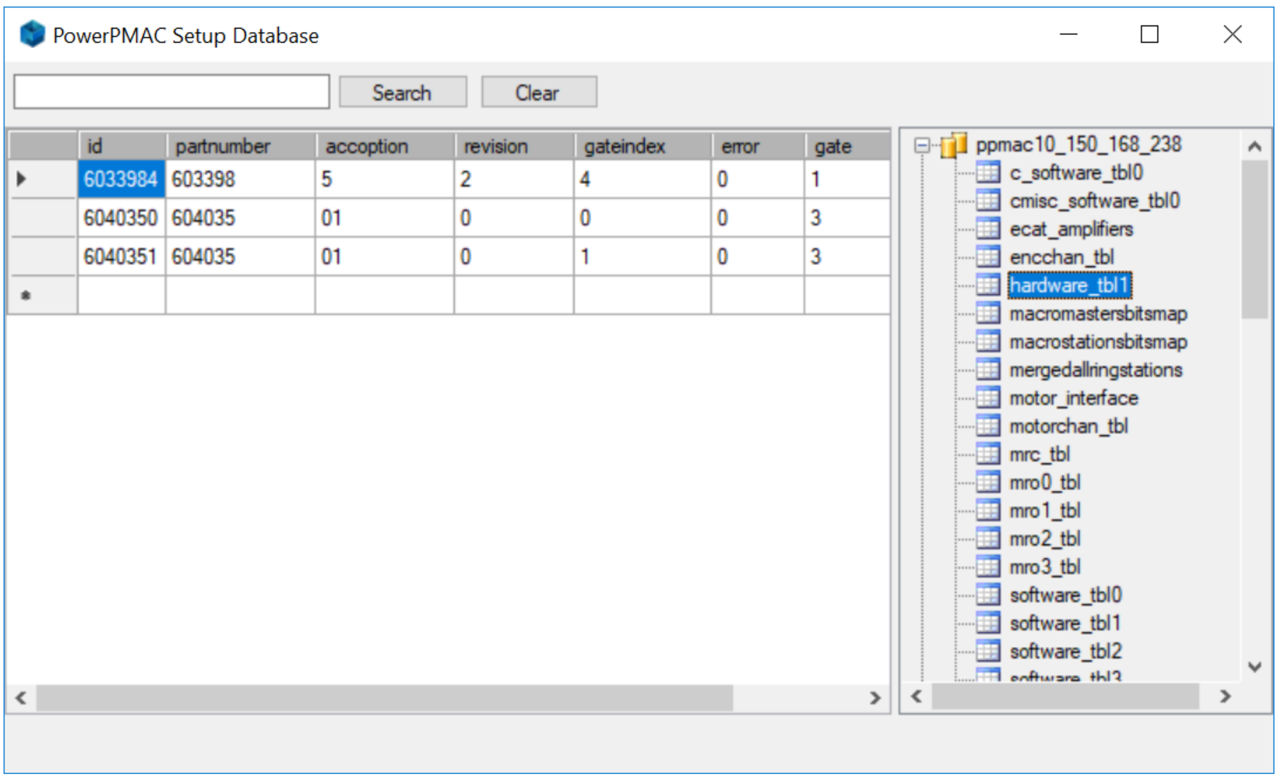
In a case when the layout is modified for any reason then this can be set back to the newly created layout by accessing the layout from the windows menu. The image below shows a user loading a layout they have named “MyNewLayout”.

View Database

This command is useful for helping to identify any Database related issues. This can be accessed from the Help menu as shown below:

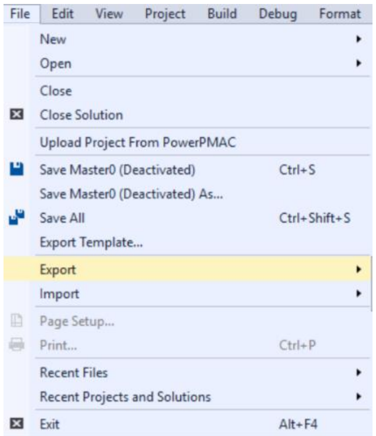


This is a simple database viewer and will display the tables that are used by the IDE. The viewer looks like this:

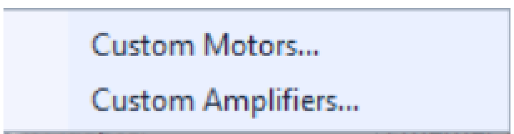


Import/Export Database

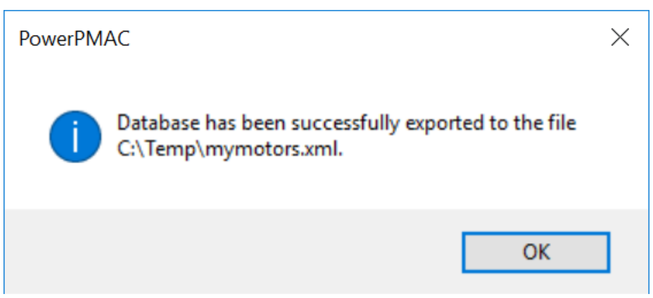
It is possible to enter custom Amplifiers and Motors to be used in the Motor setup. These databases can be exported or imported using the Import and Export functions in the File menu.



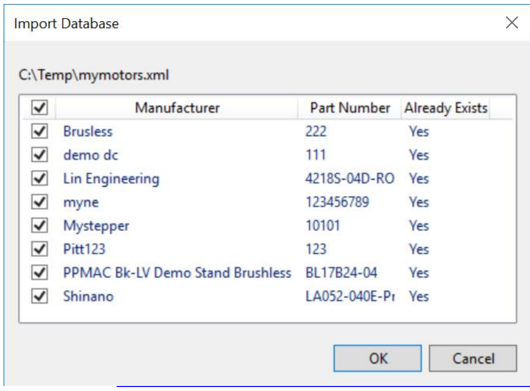
On clicking either option, a choice can be made of which databases to import or export.



On export, a location will be requested to store the file and, on success, a message will be displayed as shown below:



Similarly for Import, the user will be reqiuired to select a file to import. If items to import already exist in the database, a dialog will be displayed where individual items can be selected, as shown below:

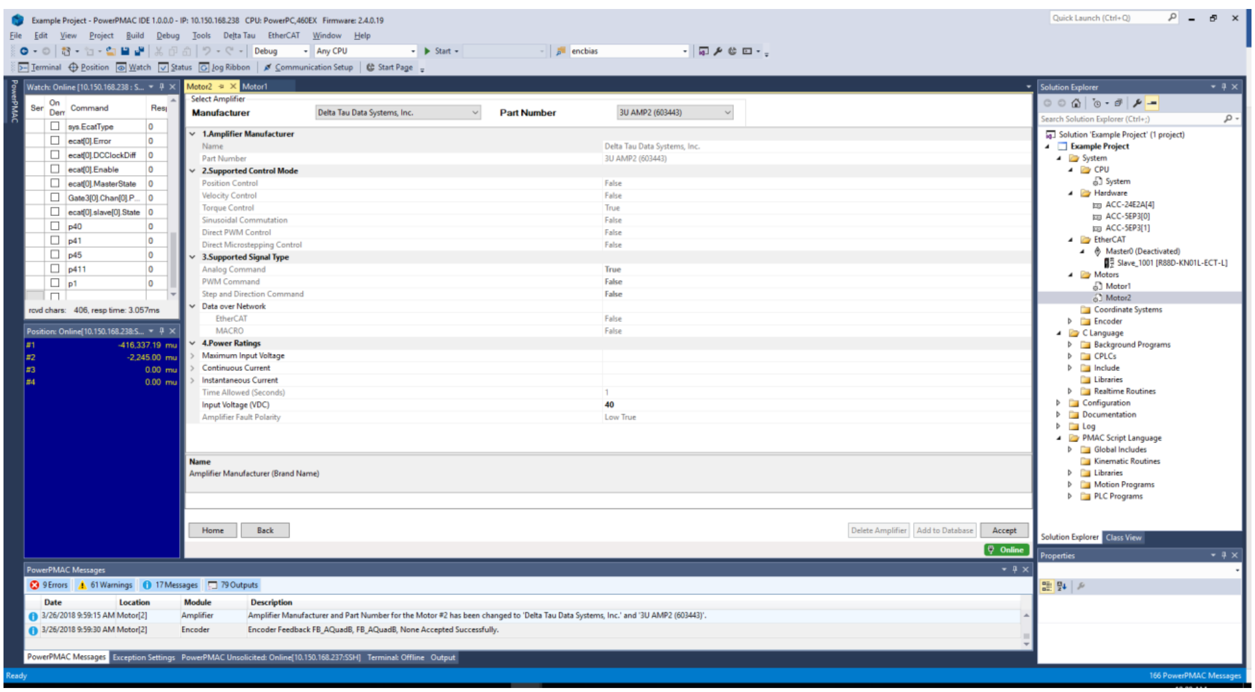


Note

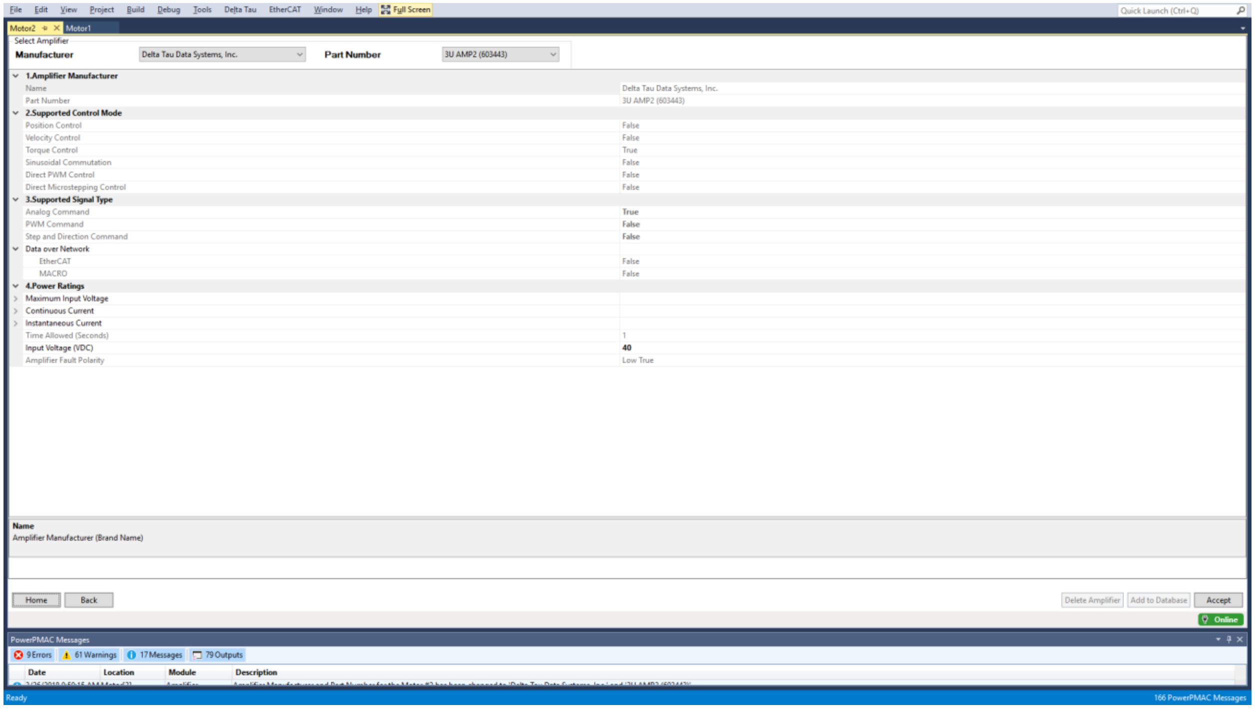
The main function of Import and Export is sharing Motor and Amplifier databases.

Set the Editor area to Full Screen

To set the Editor to Full Screen simply press Alt+Shift+Enter.

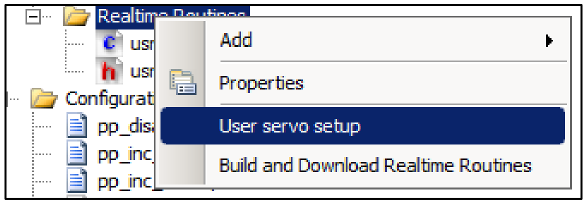


The Editor area will be displayed full screen as shown below:

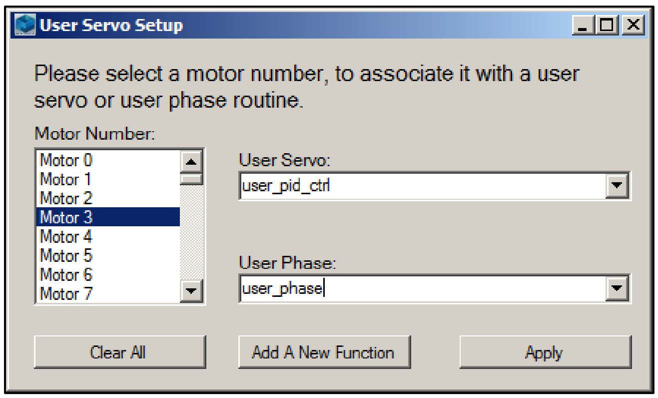


ASSOCIATING MOTORS WITH USER-WRITTEN SERVO AND PHASE ALGORITHMS

After writing usrcode.c and usrcode.h files these can be assigned to certain motors to run their algorithms through the IDE. To do this, right-click the Realtime Routines folder and click on “User servo setup”:

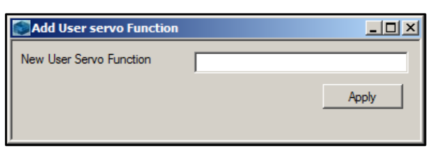


Selecting this opens the following window:



In this window select the motor to associate with a user-written algorithm. Then select the User Servo or Phase algorithm which to associate with the motor selected using the dropdown boxes on the right. Finally click “Apply” to apply the selection. This can be disassociated with all motors from user-written algorithms by clicking “Clear All.”

New user-written algorithm functioncan be added to the usrcode.c and usrcode.h files by clicking “Add a New Function” which opens this dialog box:



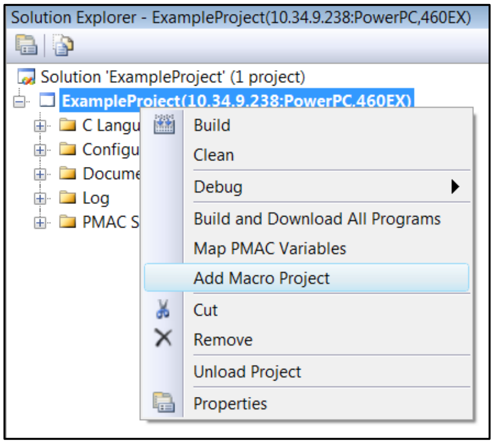
Name the function and it will be generated in usrcode.c. It’s prototype and symbol exportation will be generated in usrcode.h.

Note

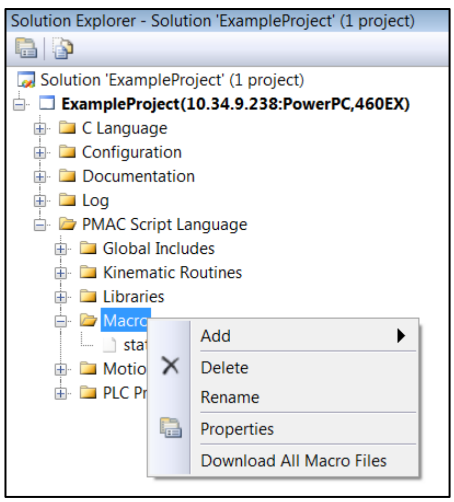
Setting up custom servo algorithms with this screen will modify **Motor[x].Ctrl**, setting it to **UserAlgo.ServoCtrlAddr[x]**; for phase, **Motor[x].UserPhase** will be set to **UserAlgo.PhaseAddr[x]**.

MACRO PROJECT

To add a MACRO project to the main project right click on the main menu and select Add Macro Project.



The MACRO Project will be added to the Power PMAC Script language folder. The MACRO project contains a default file named station1.pmh. The MACRO project acts as an independent project and its contents can only be downloaded to the Power PMAC through a menu available by right clicking on the MACRO folder. A MACRO file can also be downloaded by right-clicking on a MACRO file and selecting Dowload Selected Files.



The MACRO project can be used to isolate the main system files from the MACRO-related files such as PLCs, local settings and station settings.

DEBUGGER

The Power PMAC IDE supports Visual Studio-style debugging for Script PLCs and Background C Applications. The Debugger's environment layout is different than the standard IDE environment layout.

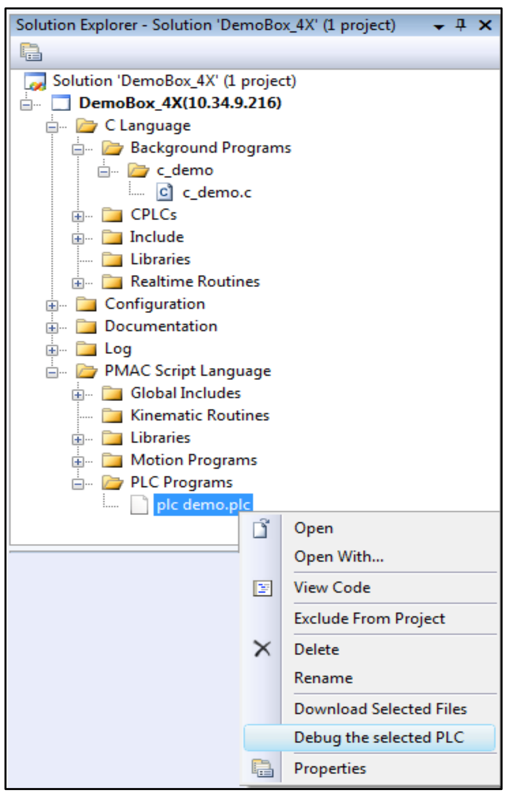
There are two prerequisites and for debugging the program:

1. The Power PMAC firmware version must be 1.5.x or greater.
2. Power PMAC project must be built and downloaded at least once before debugging.

Note If Unsolicited Response window is opened while in Debug mode, make sure that it is closed before exiting Debug mode. If the window is not closed, then it will not be possible to establish communication through another Unsolicited Response window as Unsolicited Response windows only permit one communication channel at a time.

Debugging a Script PLC

After successfully downloading the Power PMAC project right-click the Script PLC to debug:

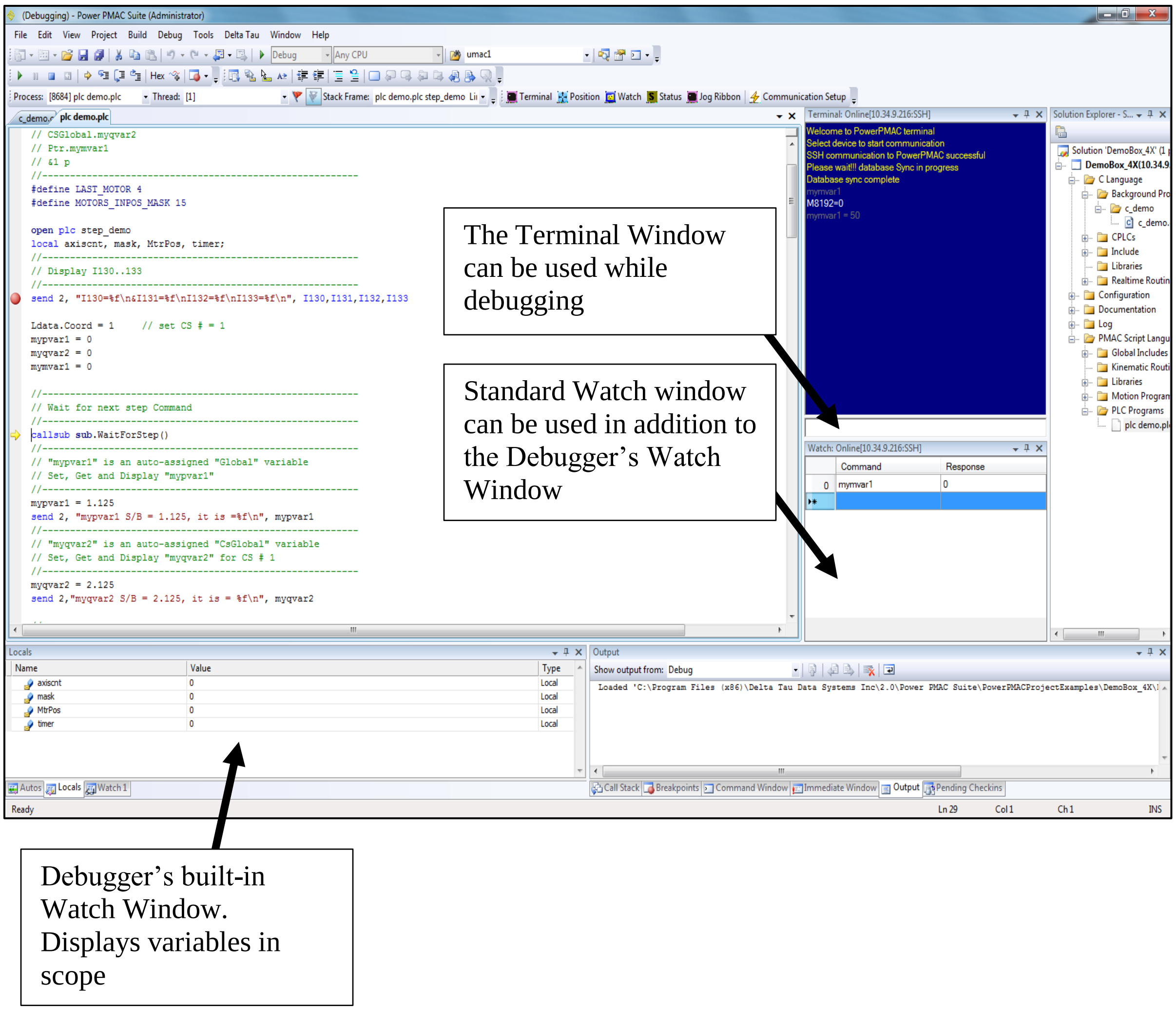


Select the context menu "Debug the selected PLC" to start the debugger.

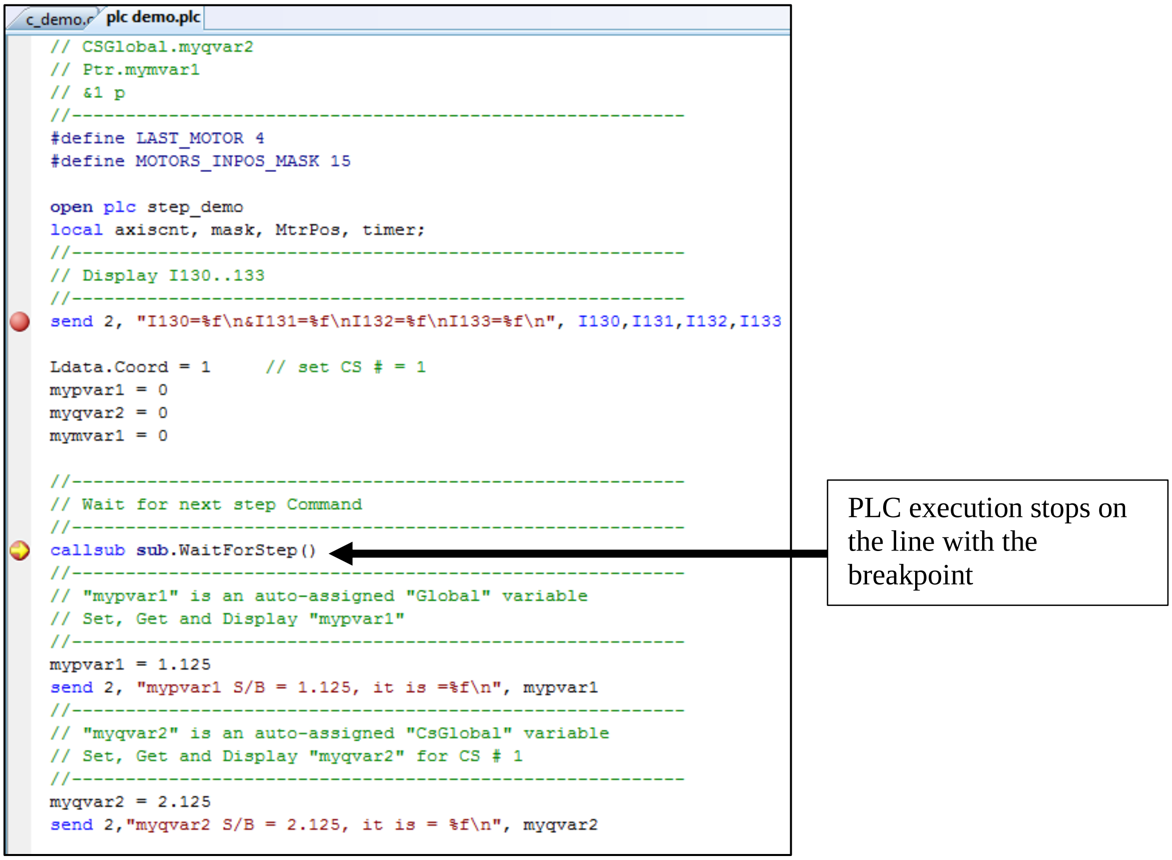
This will launch the Debug environment, as shown in the image below. In this environment the Terminal and the Watch Window are visible. The Debug environment can be customized by adding additional controls from the Delta Tau→View menu or from the Delta Tau toolbar. These controls can be helpful in viewing the variables or debugging programs interactively. This layout is automatically stored by the Power PMAC IDE as the Debug environment layout and is displayed every time a Debug session is launched thereafter.

A breakpoint can be set before or after the Debugger is launched by placing the mouse cursor onto the line to break and then pressing F9. More information about the Debug menu is available under the IDE Layout section of this manual.

The Debug Environment is shown and annotated below:



PLC execution will be stopped on the breakpoint indicated by a red dot to the left of the selected line as shown below:



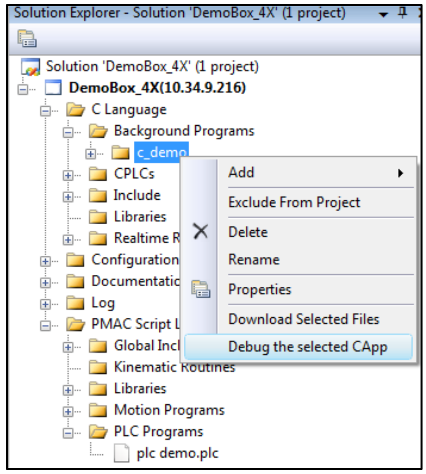
```
c_demo.c   plc demo.plc
// CSGlobal.myqvar2
// Ptr.mymvar1
// &1 p
//-----------------------------------------------------------
#define LAST_MOTOR 4
#define MOTORS_INPOS_MASK 15

open plc step_demo
local axiscnt, mask, MtrPos, timer;
//-----------------------------------------------------------
// Display I130..133
//-----------------------------------------------------------
send 2, "I130=%f\n&I131=%f\nI132=%f\nI133=%f\n", I130,I131,I132,I133

Ldata.Coord = 1     // set CS # = 1
mypvar1 = 0
myqvar2 = 0
mymvar1 = 0

//-----------------------------------------------------------
// Wait for next step Command
//-----------------------------------------------------------
callsub sub.WaitForStep()
//-----------------------------------------------------------
// "mypvar1" is an auto-assigned "Global" variable
// Set, Get and Display "mypvar1"
//-----------------------------------------------------------
mypvar1 = 1.125
send 2, "mypvar1 S/B = 1.125, it is =%f\n", mypvar1
//-----------------------------------------------------------
// "myqvar2" is an auto-assigned "CsGlobal" variable
// Set, Get and Display "myqvar2" for CS # 1
//-----------------------------------------------------------
myqvar2 = 2.125
send 2,"myqvar2 S/B = 2.125, it is = %f\n", myqvar2
```


Once the program has stopped view the Debugger's Watch Window for the values of variables that are in scope. In the current version of the IDE the local variables are automatically displayed and the user is not allowed to add other variables. Additional variables can be added to watch by opening the standard IDE Watch Window (Delta Tau→Watch); set these variables' values using the Terminal Window.

Use F11 to step into function calls or use F10 to step over functions. To stop the debugger simply press the  button from the toolbar, press Shift+F5 or select the menu item Debug→Stop Debugging. Once the debugging ceases the standard IDE environment is launched.

Debugging a Background C Application

After successfully downloading the Power PMAC project right-click the Background C Application (under Background Programs) that is to be debugged as shown below:



Select the context menu "Debug the selected CApp" to start the debugger. This will launch the same debug environment used when debugging a Script PLC.

A breakpoint can be set before or after the debugger is launched. To set the breakpoint after the debugger is launched make sure that the Background C Application is in a loop; otherwise the program execution will be completed and it will not encounter the break point. Breakpoints can be set by pressing F9. More information about the Debug menu is available under in the IDE Layout section of this manual.

The application will stop at the breakpoint set as shown below:

```
#define CHAR_BUF_SIZE  0x10000
#define LAST_MOTOR 4
#define MOTORS_INPOS_MASK 15
void WaitForStep(void);
int main(void) {

  int fd, nok;
  double getvar, getvar2, setvar;
  unsigned mask, status = 0;
  int axiscnt = 0;
  struct coorddata coord_pos[32];
  long long lldata;
  char  *szdata;

//-------------------------------
// Required Startup Code
//-------------------------------
  InitLibrary ()  ;

#ifdef SampleCPLC
//---------------------------------
// Your code starts here
//---------------------------------
//----------------------------------------------------------
// Shared MEM Ptr "pshm" is automatically available with Ini
//----------------------------------------------------------
//---------------------------------
// Set Motor IN POS band
//---------------------------------
  for(axiscnt=0; axiscnt < MAX_MOTORS; axiscnt++)
    pshm->Motor[axiscnt].InPosBand=10;

//----------------------------------------------------------
// Get a chunk of memory
//----------------------------------------------------------
```

The C Application's execution stops on the breakpoint.

At this point check the Debugger's Watch Window for variables that are in scope as shown below:

Locals

Name	Value	Type
fd	0	int
nok	0	int
getvar	0	double
getvar2	0	double
setvar	0	double
axiscnt	0	int
status	0	unsigned int

In the current version of the IDE the variables are automatically displayed and the user is not allowed to add the variables. Add additional variables to watch by opening the standard IDE Watch Window (Delta Tau→Watch); set these variables' values using the Terminal Window.

Use F11 to step into a function or use F10 to step over a function. When stepping into a function the Call Stack Window will display the calling sequence. The Debugger supports multiple levels of call-stacks.

To stop the debugger, simply press the  button from the toolbar or press Shift+F5 on the keyboard or select the Debug→Stop Debugging menu item. Once the debugging ceases the standard IDE environment is launched.

MATLAB/SIMULINK TARGET FOR POWER PMAC

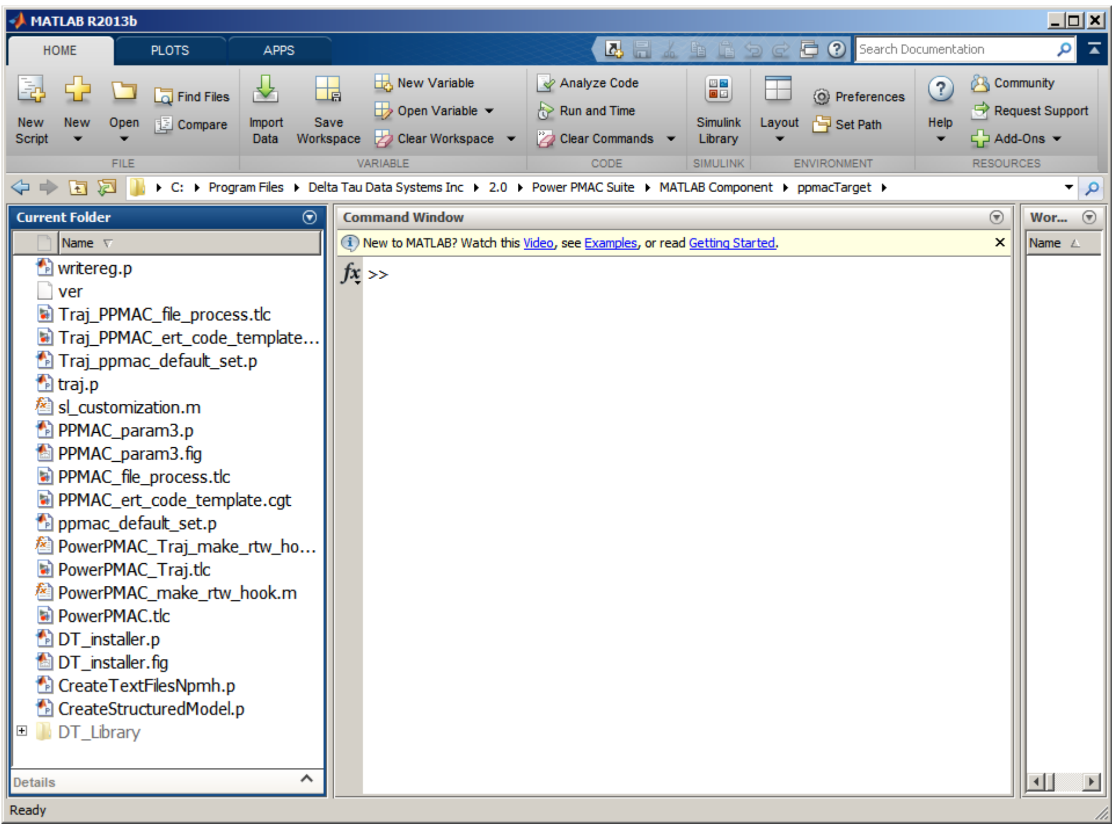
Installing the Power PMAC Target on MATLAB

By default, the MATLAB Component's installation folder is installed with the Power PMAC IDE. If the PC's operating system is 32-bit, it can be found at:

C:\Program Files\Delta Tau Data Systems Inc\2.0\PowerPMAC Suite\MATLAB Component \ ppmacTarget

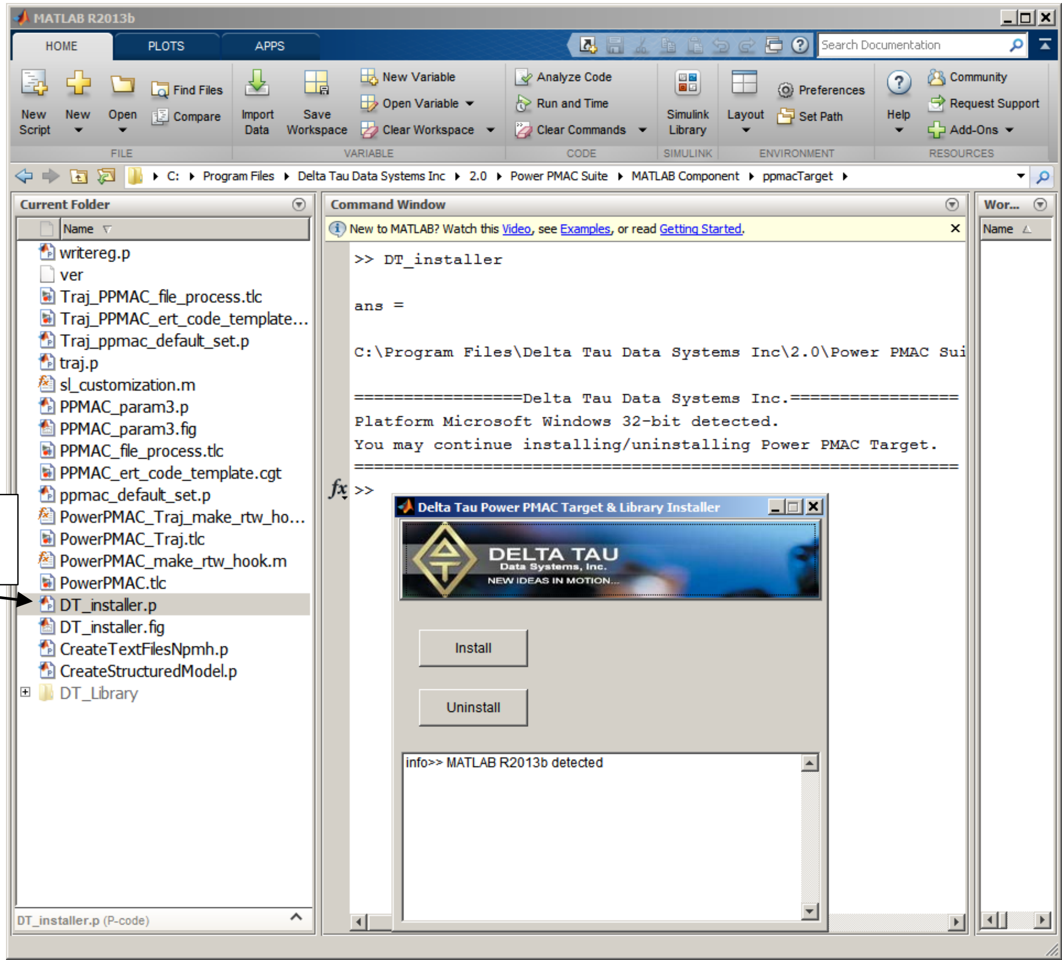
To install the component in MATLAB, do the following:

1. Launch MATLAB 2013b .
2. Change the "Current Folder" to the above folder.

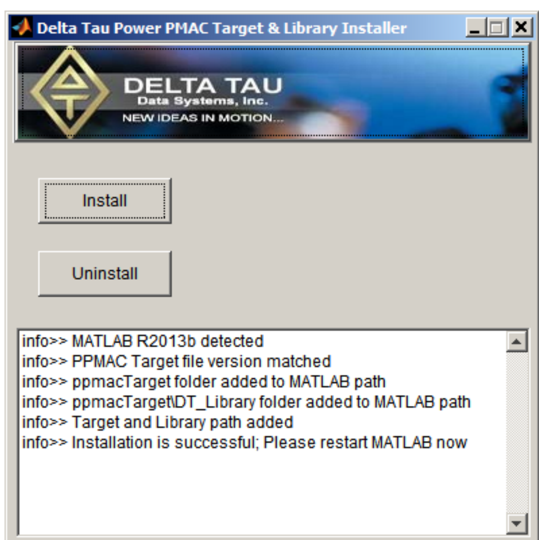


3. Run the DT_installer.p file by either right-clicking on the file in MATLAB's "Current Folder" window and selecting "Run" or by typing **DT_installer** and pressing enter in MATLAB's "Command Window". The installation interface should then launch.

This file needs to be run



4. Press Install and if the MATLAB version is 2013b installation will complete successfully, as shown below:



5. Exit MATLAB and launch it again.

How to use Simulink to Generate User-Servo C Code

After installing the Power PMAC Target on MATLAB Simulink can be used for model development and C code generation. The C code can do user servo algorithm tasks or any mathematical calculation that needs to be run at a determined interrupt (i.e. at a multiple of Power PMAC's servo interrupt).

The following example shows how the user can design a PID algorithm in Simulink, use the Target to generate the C code expressing the algorithm and then deploy the C code through the Power PMAC IDE as the control algorithm for any motor (virtual or real).

Example: Modeling PID Control of a Brush Motor

Step 1: Design the Model

First, the model should be designed in Simulink with the proper parameters and then verified using the Simulink source and sink blocks if necessary. The following is an example PID control algorithm model for a brush motor whose transfer function is approximated by

$$\frac{Y(s)}{R(s)} = \frac{183}{s^2},$$

where ***Y(s)*** is the output from the motor and ***R(s)*** is the input to the motor.

Note

To learn more about how to find an approximation for the motor, Delta Tau's Servo Analyzer application can be used. The Tuning application in Power PMAC IDE can also be used for this purpose.

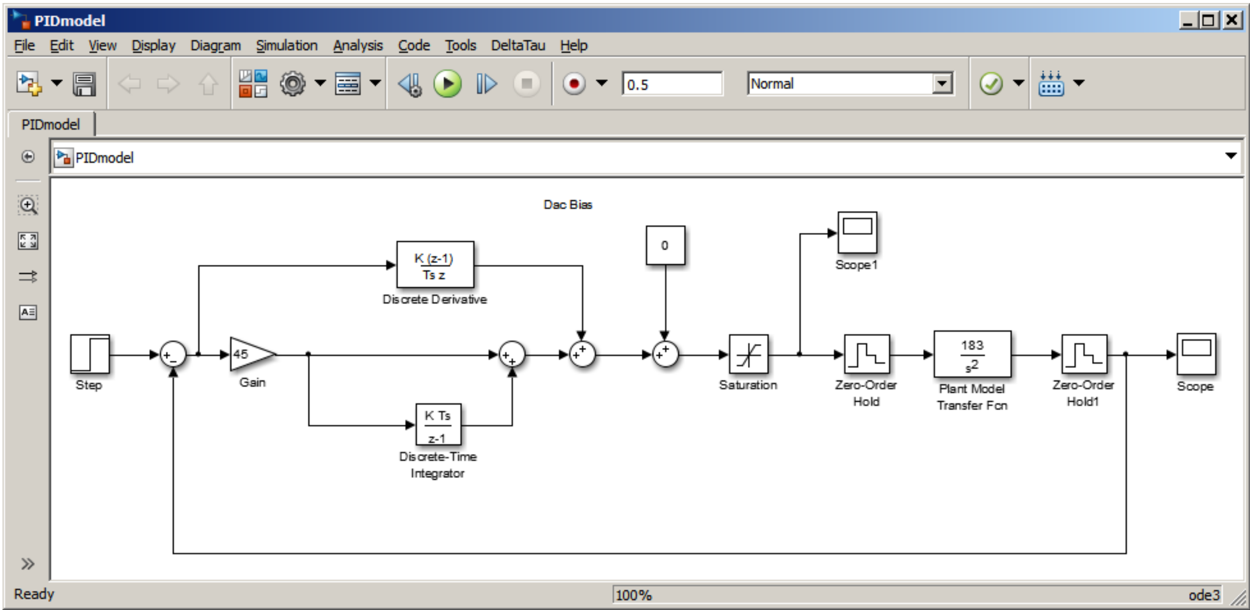
Note that the values put for the derivative blocks need to be multiplied by this motor's servo rate, which is the rate at which the servo algorithm will be executed, and the integration gains need to be divided by that value. For example if the algorithm is going to be used as the user-servo routine for Motor 1, then in the Gain Value property of the derivative block, put the numerical value of

Motor[1].Servo.Kvfb*Sys.ServoPeriod*Motor[1].Stime,

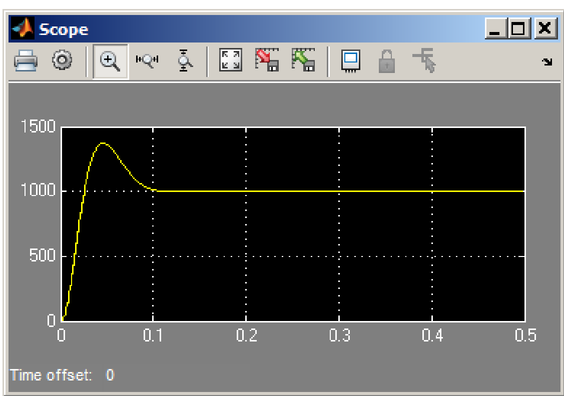
and use the numerical value of

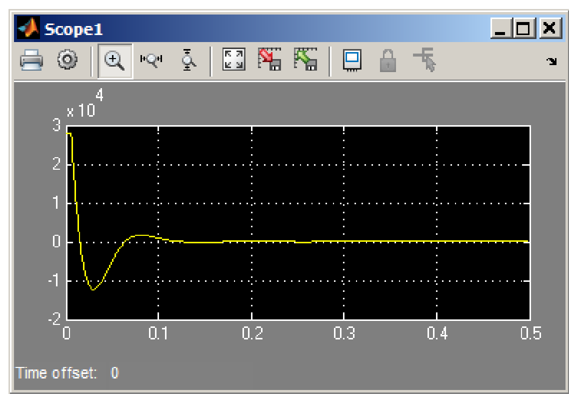
Motor[1].Servo.Ki / (Sys.ServoPeriod * Motor[1].Stime)

for the integrator gain. In this example, Kvfb=1500 and Ki=0.01 are used and the servo rate is the default value of Sys.ServoPeriod=0.00044274211. Motor[1].Stime=1 and Kp=45 are set. In other words, 1500*0.00044274211 and 0.01/0.00044274211 are the values put in the Gain values of the derivative and the integrator gains, respectively.



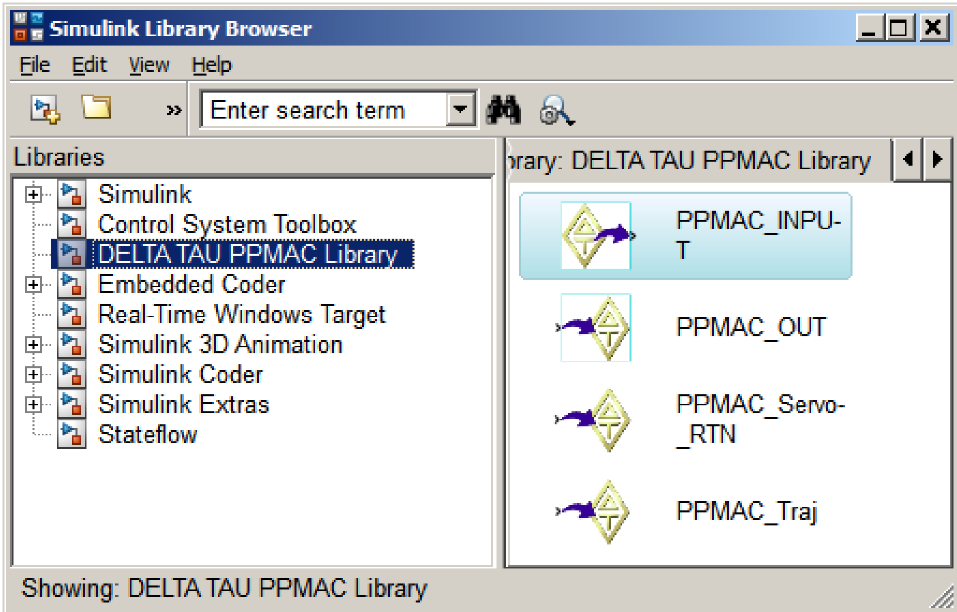
The model can be tested by starting the simulation and checking the results. If the results are not satisfactory, the parameters can be changed and tuned in Simulink before code generation starts. For this example, here are the plots Scope and Scope1, showing a step response:





Step 2: Include Delta Tau Library Blocks in Simulink

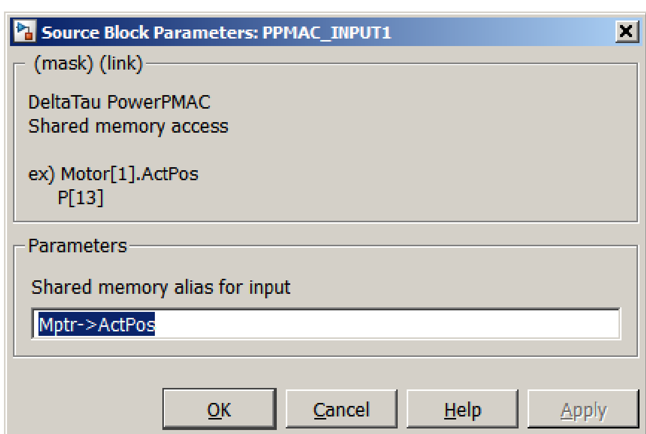
The second step includes using the Delta Tau Library blocks in Simulink as for the inputs and outputs of the algorithm. To do so, launch the Simulink Library browser. The following picture shows how the Delta Tau library looks after the Power PMAC Target been successfully installed on MATLAB.



The library includes 4 blocks:

- PPMAC_INPUT
- PPMAC_OUT
- PPMAC_Servo_RTN
- PPMAC_Traj

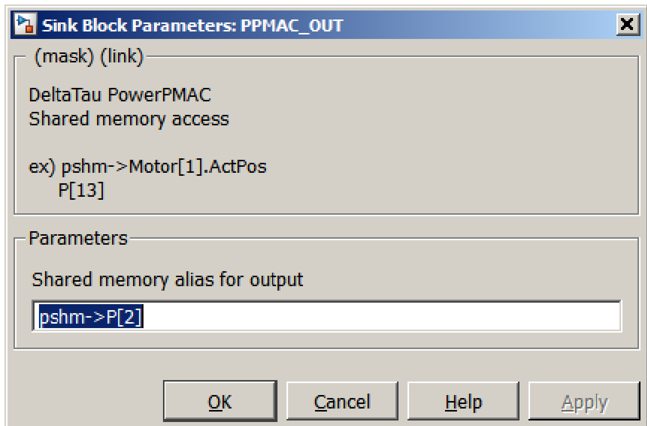
The PPMAC_INPUT and PPMAC_OUT blocks can be used anywhere the user needs to have access to a memory location in Power PMAC. Use PPMAC_INPUT to get data values from Power PMAC to use in the algorithm and PPMAC_OUT to set (write) data values to Power PMAC. After putting one of these blocks into the model double-click it to set the memory location with which it is associated. Double-clicking an input block will bring up the following screen:



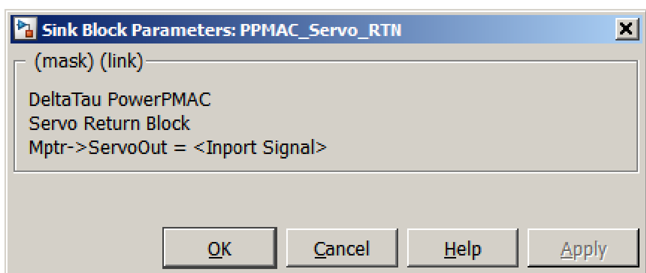
Here are some examples of memory locations:

Pshm->P[1]
Pshm->Ddata[0]
Mptr->ServoOut
Mptr->IqCmd
Pshm->Motor[3].Kp

This screen below is displayed when an output block is double clicked:

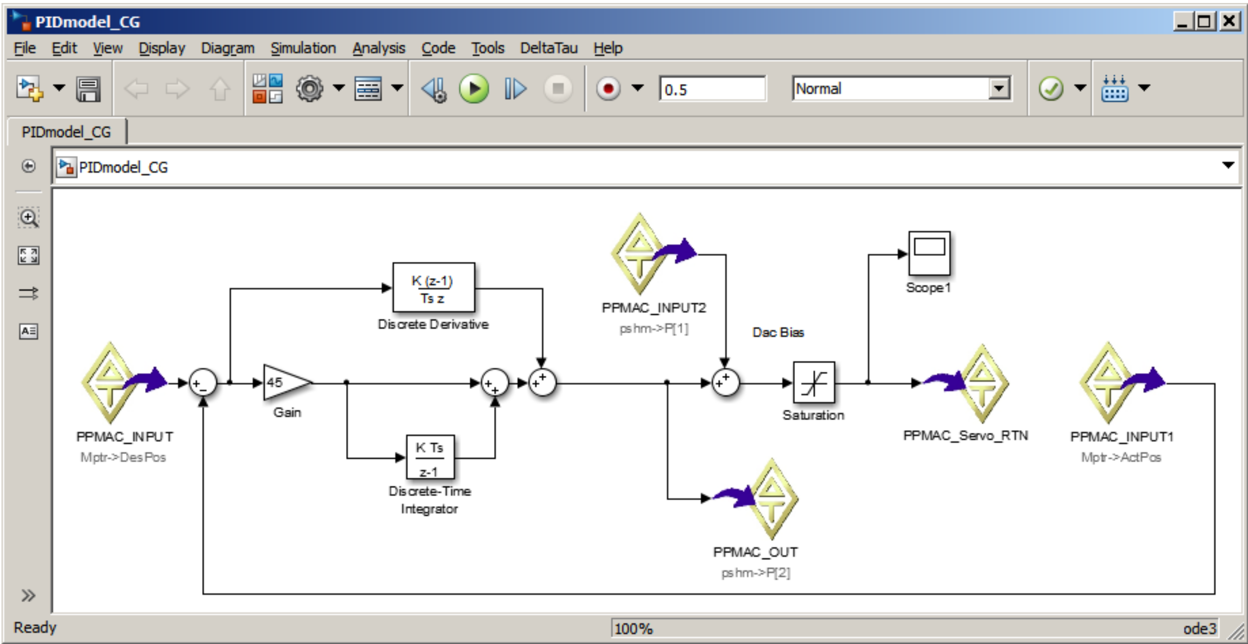


This screen below is displayed when an return block is double clicked:



At this point saving the test model as a different name and then working on the new model for code generation is recommended.

In this example there are three input blocks, one output block and one servo return block used and the parameters are set as shown below:

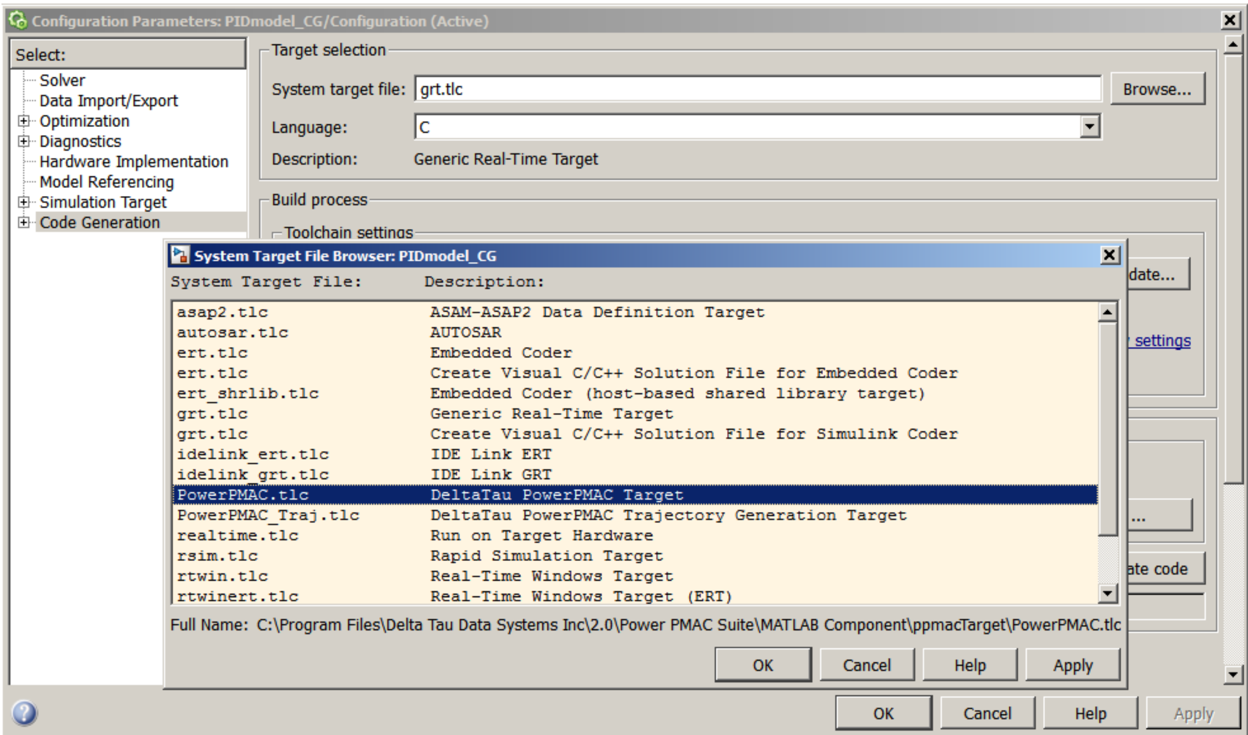


Mptr->DesPos and **Mptr->ActPos** are used to have access to the desired position and actual position of the motor that runs this servo algorithm, respectively. **Pshm->P[1]** is used as an input; its value will be added to the corresponding connected signal. **Pshm->P[2]** is also used in an output block to get the value of the connected signal and write it to **P[2]** for parameter monitoring or other purpuses.
The PPMAC_Servo_RTN block can only be used ***once*** in a model. The value of the connected signal to this block will be written to **Mptr->IqCmd**, which is the DAC output of the motor running the servo algorithm. If this block is used, the model needs to be built and the C code needs to be generated with the **PowerPMAC.tlc** target, which generates servo algorithms, and not **PowerPMAC_Traj.tlc**, which generates trajectories. PPMAC_Traj block can also be only used with the **PowerPMAC_Traj.tlc** target and not **PowerPMAC.tlc**.

The models that include Deltau Tau's Power PMAC Library blocks can only be used for the purpose of code generation and not simulation (i.e. they cannot be used in Simulink at runtime).

Step 3: C Code Generation

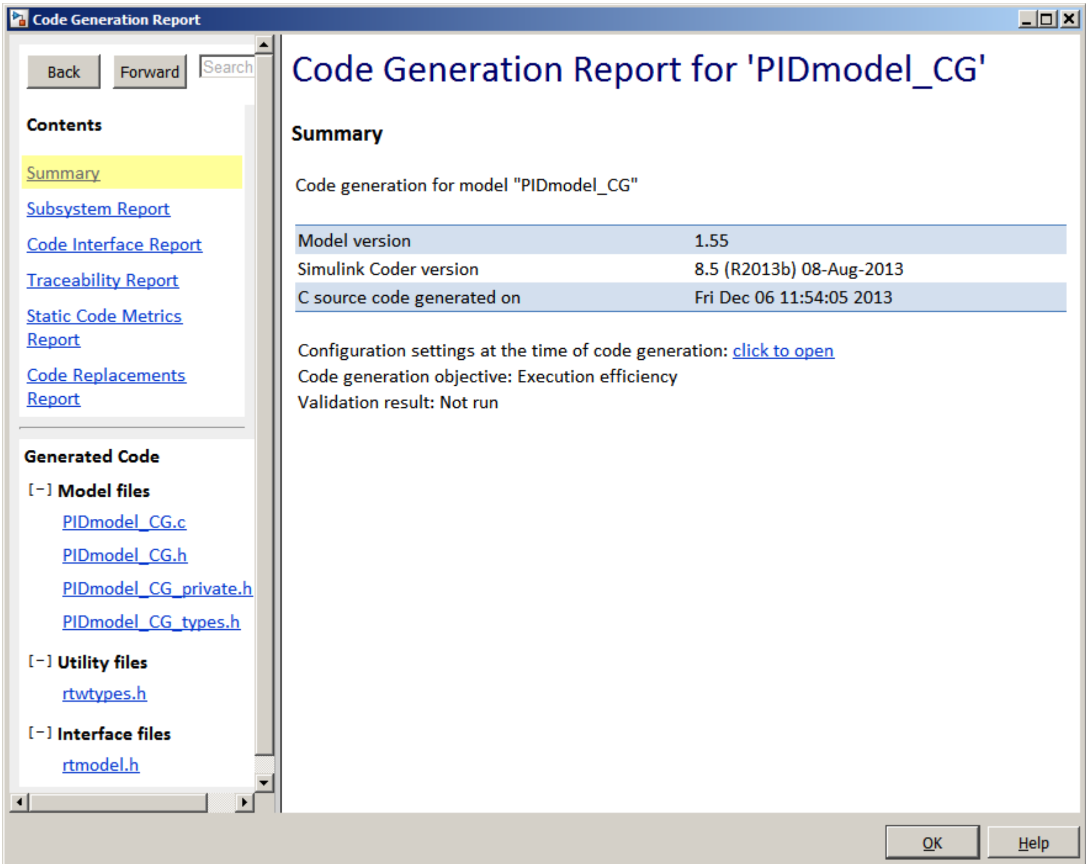
The third step is to generate the C code. Open the model's "Model Configuration Parameters" dialog box which can be found at the "Simulation" menu or by pressing Ctrl+E. Go to the Code Generation pane of the dialog box and choose **PowerPMAC.tlc** as the System Target File as shown below:



Setting **PowerPMAC.tlc** as the system target file forces the Simulink Coder and Embedded Coder to generate C code that is compatible with Power PMAC's memory structure and can be downloaded to Power PMAC. If the servo rate is different than default it needs to be set here at the Solver pane under Fixed-Step Size.

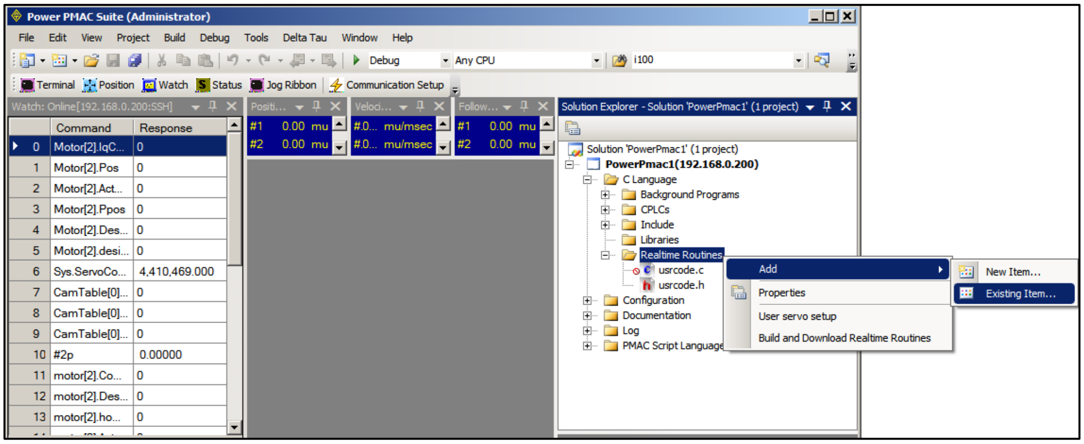
Apply the changes in the Model Configuration Parameters dialog box and save the model again. Return to the dialog box and press the Generate Code button in the "Code Generation" pane. The C code will be automatically generated and saved in MATLAB's current folder. A report including the C code (**.c** source and **.h** header files) will be automatically opened and saved in the same folder as well.

Click on the links on the left tab of the report, shown below, to see the generated C code:

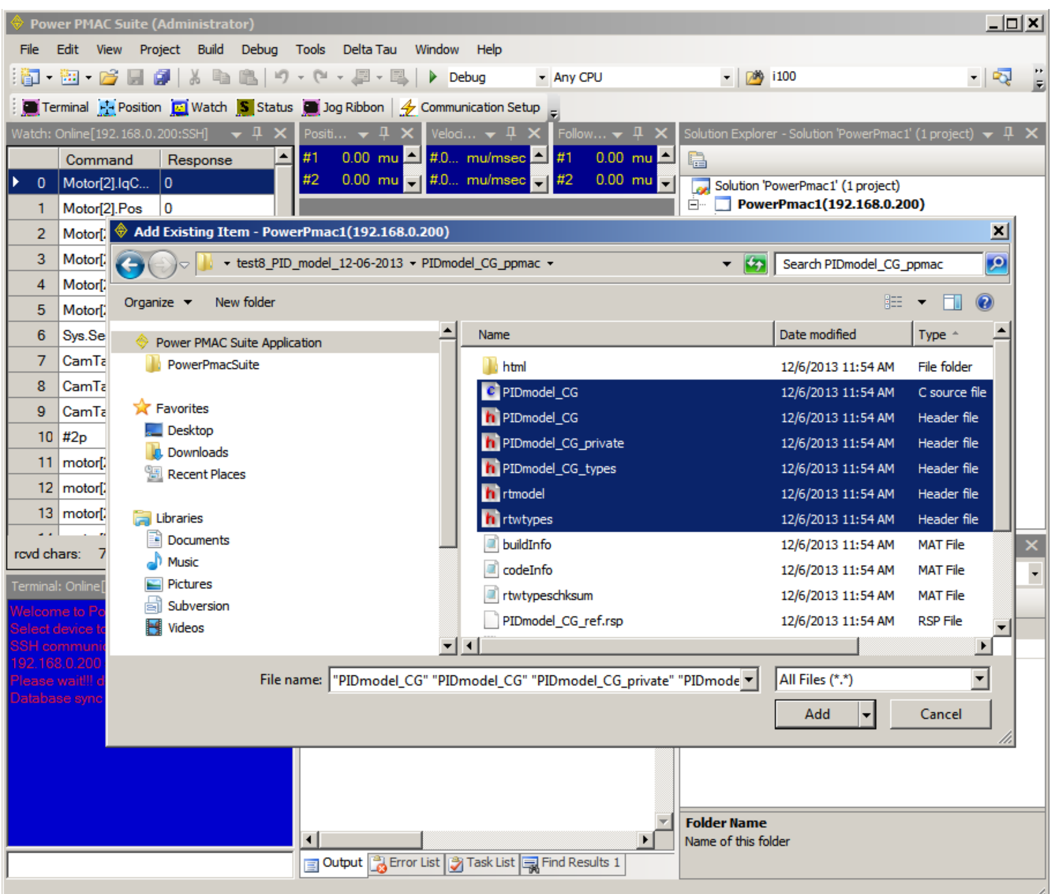


Step 4: Deploy the Model in the Power PMAC IDE

The fourth step is to deploy the model in the Power PMAC IDE. To do so create a new folder, preferably in MATLAB's Current Folder. Launch the Power PMAC IDE and create a new project in that folder. In the Power PMAC IDE, open the "Solution Explorer" window and then the C Language→Realtime Routines folder. Right click on the "Realtime Routines" folder, choose "Add Existing item…" (as shown below) and then go to the folder where the generated code was saved.

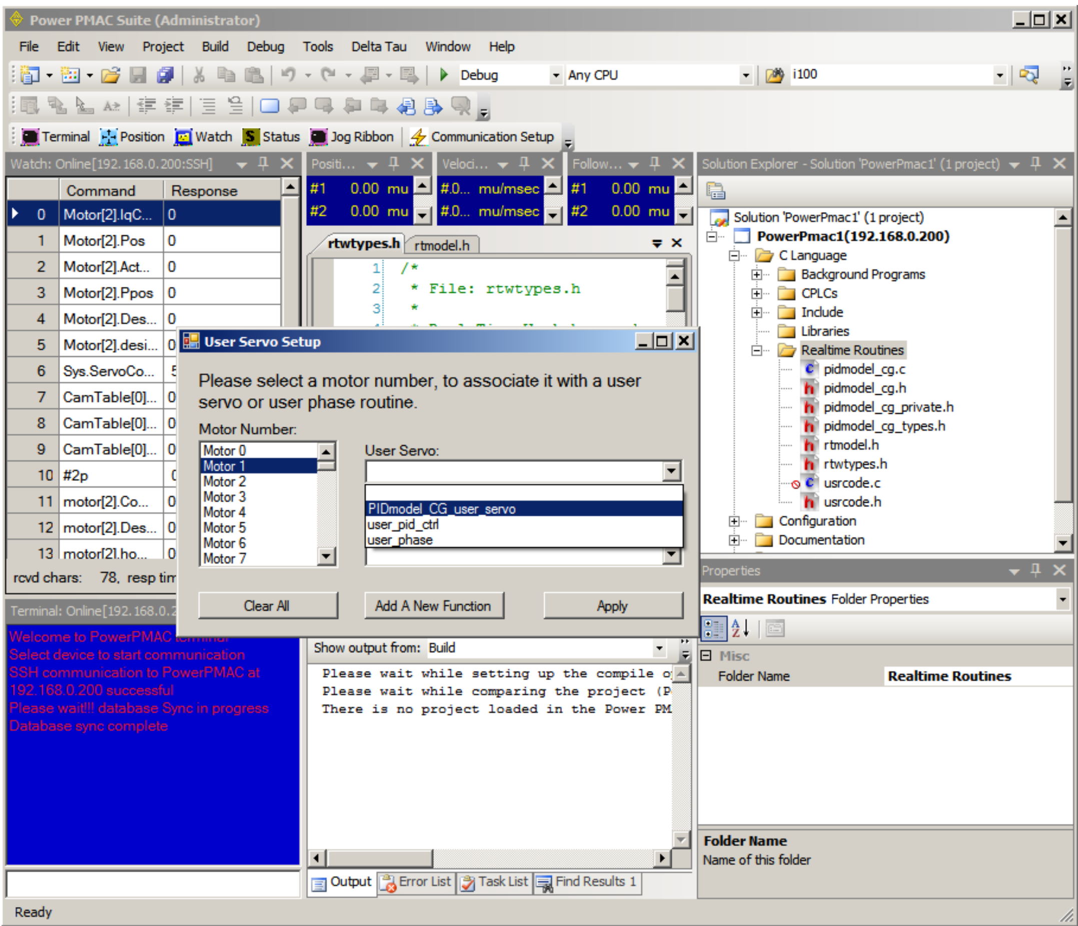


Add all of the generated .c and .h files as shown below:



The generated files are saved in a folder in MATLAB's Current Folder at the instant when the user-clicked on the "Generate Code" button on the Model Configuration Parameter's dialog box. The name of the folder is the same name as the model but with a **_ppmac** suffix.

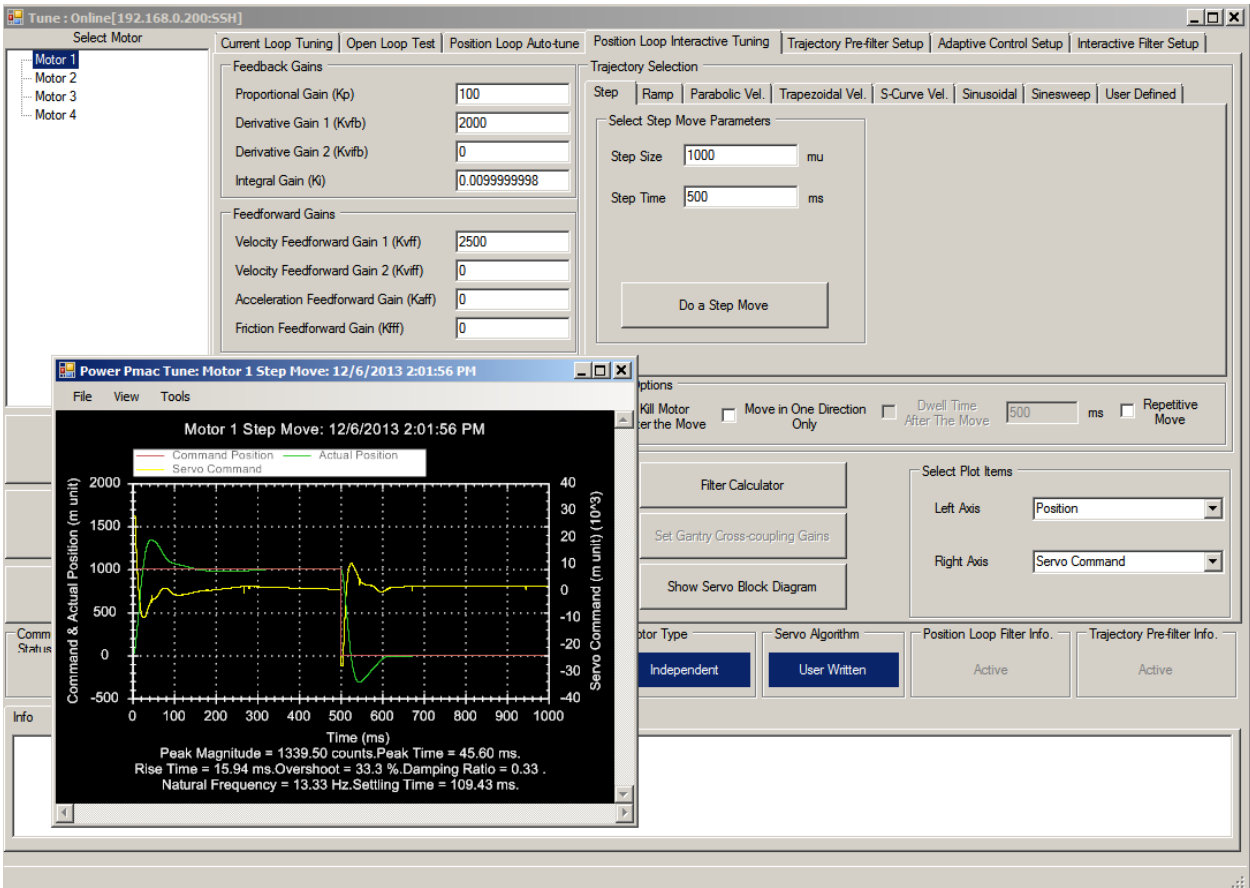
Right-click on "Realtime Routines" and choose "User Servo Setup,". Choose the number of the motor that will execute the servo algorithm, select the User Servo's name and then press Apply, as shown below:



Add any other necessary files to the project right-click the Project and then click "Build and Download". The selected motor's user servo algorithm will start running as soon as the motor is activated and enabled. To activate the motor issue a **Motor[1].ServoCtrl=1** command and to enable it, issue a **#1j/** command from the terminal Window. To verify that the motor is using the user servo algorithm, check the value of **Motor[1].Ctrl**. If it is set to **UserAlgo.ServoCtrlAddr[*i*]** then it is using the user servo algorithm.

Step 5: Verify the Result

To verify the result give the same desired position input to the motor that was commanded in Simulink (e.g. a Step input of 1000 cts for 0.5 sec). This could be done using the IDE's Tuning application (from within the IDE, click Tools→Tune). The following image shows a real result from Motor 1 on a UMAC Demo rack:



The result of the step response from the Tuning software should now be compared with the step response obtained from Simulink previously to see how closely the model matches the real response.

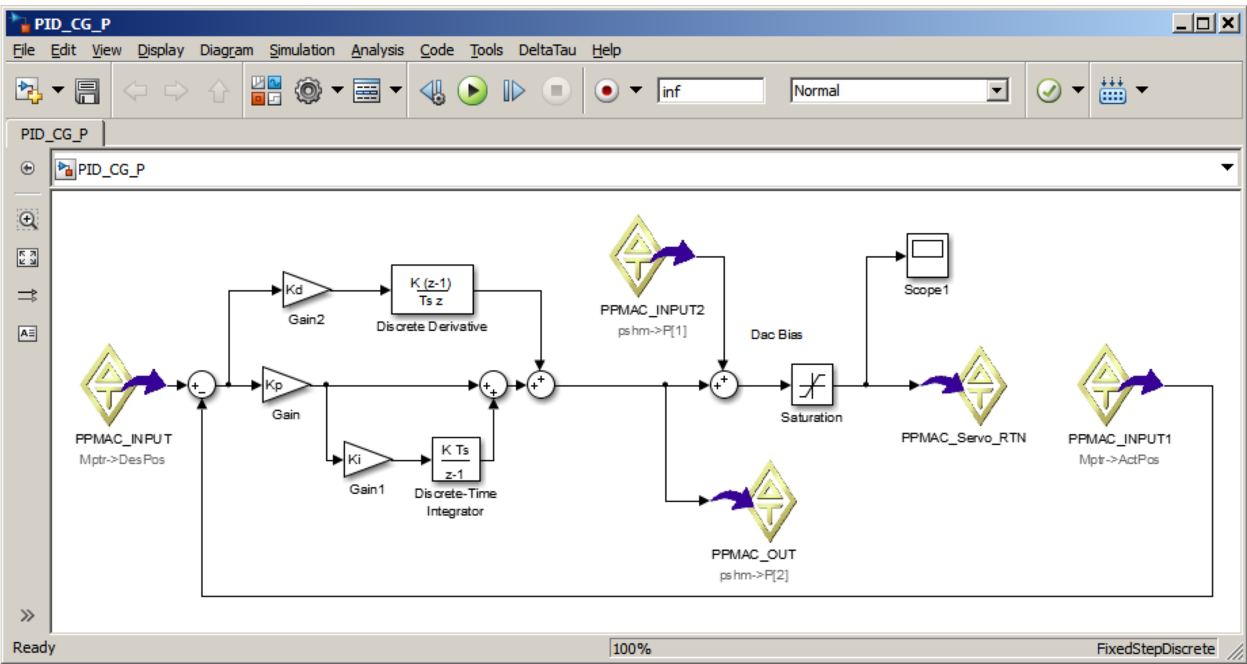
Using Tunable Parameters in Models and Code

The parameter values in the previous example model (e.g. Kp=45, Kd=1500*0.000442, Ki=0.01/0.000442) are hard-coded in the generated C code. In the last example, after a test run of the project in Power PMAC IDE, if the value for any of those parameters needs to be changed then the C code needs to be changed and the whole project be built again and downloaded again.

Here is how one can generate C code with tunable parameters. These parameters could then be changed dynamically as the program is running.

Example: Variable Kp, Kd, and Ki

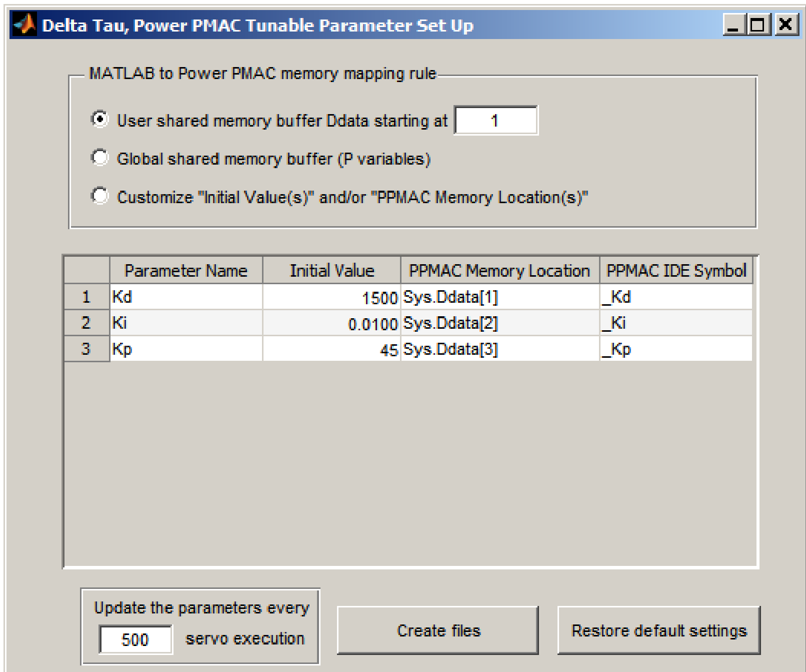
In the Simulink model, replace the model parameters with Kp, Ki and Kd. Do not specify these parameters inside the derivative or the integrator blocks themselves; the gains must be separated from the integration or differentiation blocks (see the following picture):



The parameters Ki and Kd can be put as gains before the integrator and derivative blocks, respectively. The values of 0.00044274211 and 1/ 0.00044274211 (i.e. the numerical values of the servo period and its inverse, respectively) need to be set for the gains of the Integrator and Derivative blocks, respectively. The numerical values of Kp=45, Ki=0.01 and Kd=1500 also need to be present in MATLAB's base workspace. To do so, type **Kp=45; Ki=0.01; Kd=1500;** in the MATLAB command window.

Before generating the C code in the Simulink Model, click on the "DeltaTau" menu, then "Parameters", and then "Model Validation". A message will ask if the parameter's numerical values need to be attached as a preload function to the model or not. If "Yes" is clicked the next time the model is opened the same numerical values for Kp, Kd, and Ki will appear in MATLAB's workspace. These numerical values will be used only as an initial value for the parameters; the parameters could be changed later when deployed in the Power PMAC IDE. In the "Model Validation" program check the model to make sure that only acceptable Simulink blocks are being used. MATLAB does not yet support some Simulink blocks like the "Discrete Derivative" and "Discrete Integration" for parameter tuning. Many other blocks like "Gain" and "Constant" are supported. Using these two blocks with tunable parameters is often enough for many models. Note how the Gain blocks are used in the above example in order to tune parameters affecting the "Discrete Derivative" and the "Discrete Integration" blocks.

Next, in the Simulink model window, click on the "Delta Tau" menu, then "Parameters" and then "Parameter Assignment". A dialog will open which asks the user about the memory location in Power PMAC to which the model parameters should be saved and from which to be updated when the project is running. There are three options: **Sys.DData[*i*]** user buffer memory, Global Variables (P-Variables), or a custom memory location as shown below:

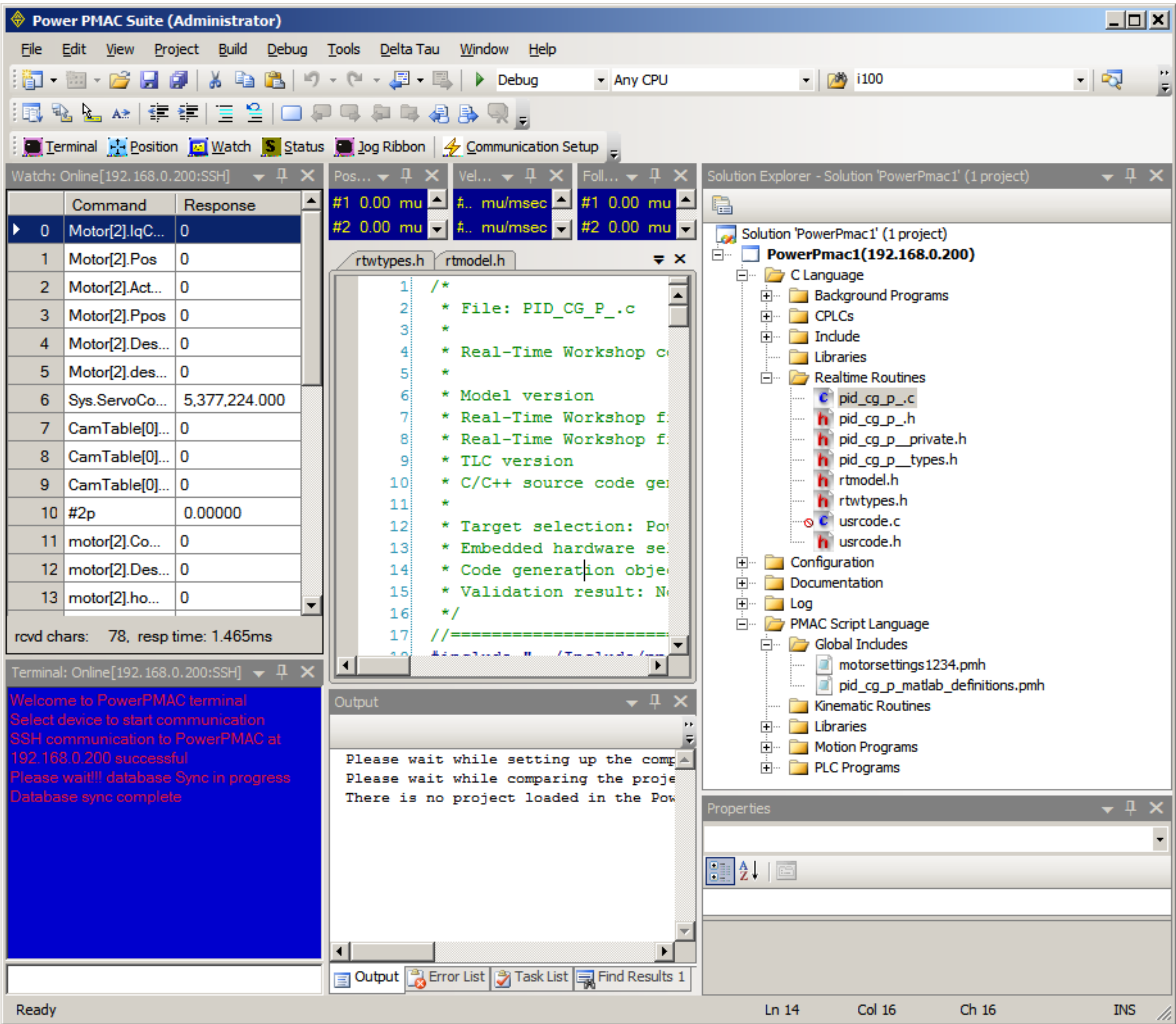


The parameters will be updated every 500 servo cycles by default. This number can be changed using the provided text box. The user can also change the initial values of the parameters here as well (in addition to in the MATLAB workspace). The Power PMAC memory location is also displayed. The Power PMAC IDE symbols correspond to the symbols that will be created in the Power PMAC IDE project. The user can write to these variables (e.g. _Kp, _Ki, _Kd) and change the values of the adjustable parameters on the fly through the Terminal Window, PLC, a motion program ,or in other C programs.

Click on the "Create files" button. When clicked, the model closes and a new model with the same name but with an additional "_" suffix is created and opened. This model will be used for code generation. This model has structured, tunable parameters. A new folder is also created in the same folder as the first model. This folder is named after the first model with "_param_ppmac" suffix. This folder includes 3 text files named "…_param_update.txt", "…_param_initial.txt" and "…_MATLAB_definitions.pmh", which will be used later.

Start the process of code generation for the new model whose name ends with "_". To do so, as explained in the last section, in the Simulink model, open the "Model Configuration Parameters" dialog box from the "Simulation" menu (or pressing Ctrl+E). In the "Code Generation" pane, go to "Target selection" and then "System Target File". Click "Browse" and choose the target. Choose "Power PMAC.tlc" for general math and/or control loop algorithm (user-servo) code or choose "Power PMAC_Traj.tlc" for the custom trajectory generation target. Tunable parameters are used in the same way for both these targets. After the target is chosen, save the model, then go back to the "Model Configuration Parameters" dialog box. Go to the "Code Generation" pane and click on "Generate code". The generated code will be saved in MATLAB's Current Folder and a report that includes the generated code will launch. The code is saved in a folder named after the original model with the "__ppmac" suffix.

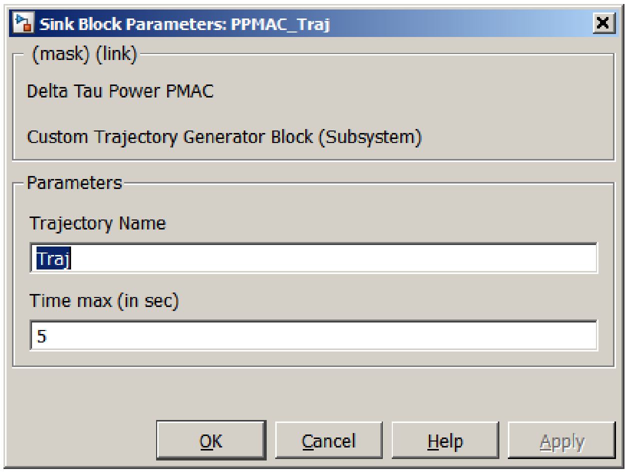
The Generated code can be deployed the same way as explained in the last section to the Power PMAC IDE project, with the only difference being that the file named "…_MATLAB_definitions.pmh" saved in the "…_param_ppmac" folder also needs to be put in the Power PMAC IDE project in the Script Language→Global Includes section. See the following picture for an example.



To tune the parameters using the variable names created in the .pmh file. For example, command "_Kp=44" in the terminal window to set the Kp parameter equal to 44.

How to Use Simulink to Create a Trajectory

The Power PMAC Target can be used for generating trajectories as well. The generated trajectory will command motors (not axes) individually. The trajectory can be made by using the PPMAC_traj block made specifically for this purpose. It can be found in the “Simulink Library Browser” in “Delta Tau PPMAC Library” within Simulink. The input signal to the PPMAC_Traj block will be the position signal commanded to the motor. More than one trajectory can be made in every Simulink model. Out of each of the PPMAC_Traj blocks, only one trajectory is made. For example, five PPMAC_Traj blocks in a Simulink model makes five trajectories, each of which could be run on any motor. These trajectories need to have distinct names. The names of these trajectories will be used to create a flag which can be used (in addition to a motor number) to start the trajectory for that specific motor. For example if the trajectory is named “Sintraj,” to start this trajectory for Motor 4, user needs to issue the command “SinTraj_flag(4)=1”. The trajectory will run as long as it is defined in the Simulink model. The total amount of time that the trajectory runs is fixed and cannot be changed; it must be defined in the Simulink model in the PPMAC_Traj block’s parameters. On the right is a screenshot from double-clicking the block:

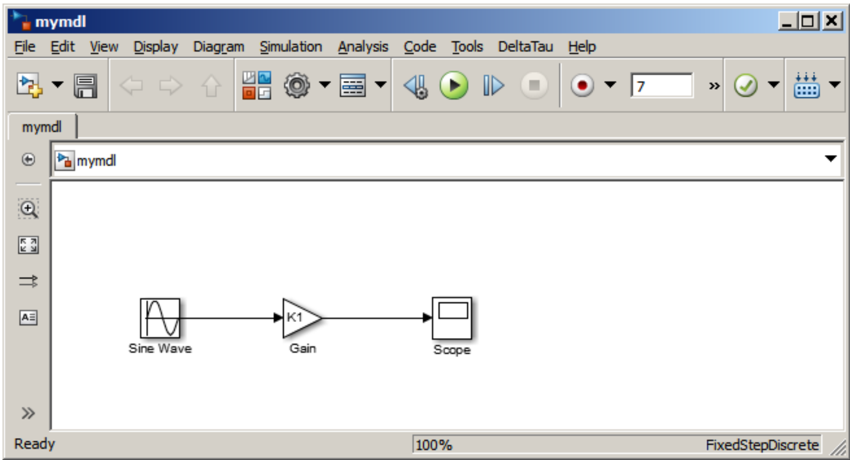


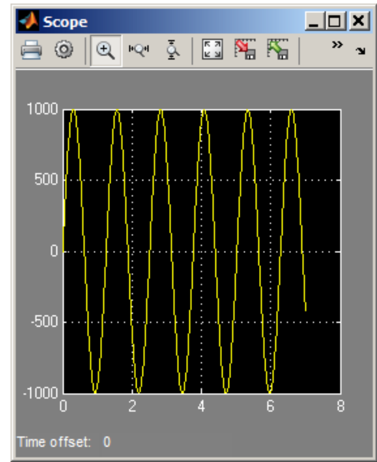
“Trajectory Name” and “Time max” (in seconds) are the two parameters that need to be set for every PPMAC_Traj block.

Trajectories are usually made using the Simulink source blocks. All the time parameters (if there are any that need to be set in blocks) must be set in units of seconds. Frequencies must be set in Hz. To access the “Source” blocks launch the “Simulink Library Browser”, then go to Simulink→Sources. Most Simulink blocks can also be used in the model. Delta Tau does not support the following “Source” blocks for trajectory code generation using the PowerPMAC_Traj.tlc target:“Counter Limited”, “Digital Clock”, “Enumerated Constant”, “Random Number”, “Unified Random Number” and “Band Limited White Noise”. Other “Source” blocks, however, can be used for this purpose. To check if other blocks in other libraries are supported, check the Simulink Coder’s list for “Supported Blocks for Code Generation” in MATLAB’s documentation.

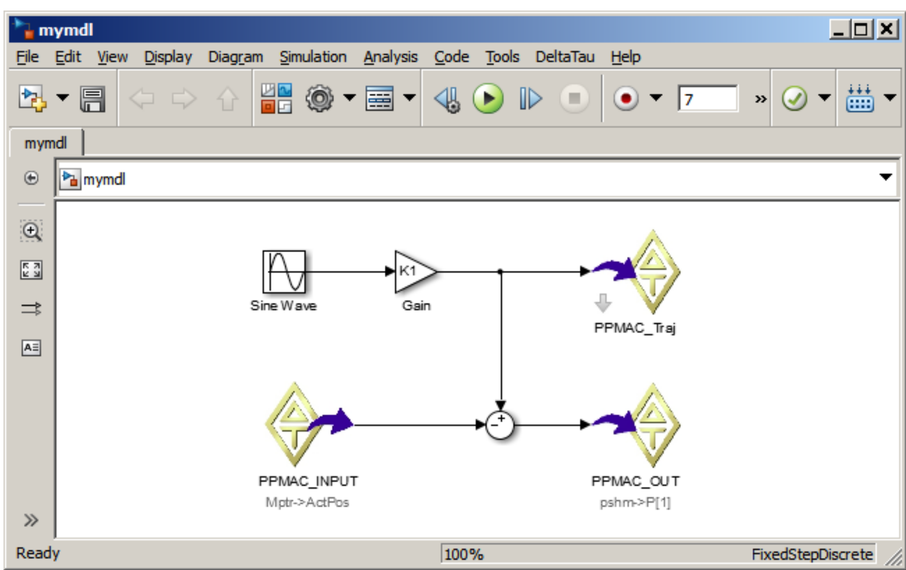
Example Trajectory Generation Model

The following is an example of how to create a model in Simulink. The following picture shows the model of a time-based sinusoidal input. The sine wave uses, Amplitude=1, Bias=0, Frequency=5 rad/sec, Phase=0 radians and Sample time=0 sec. The Gain K1 has the value 1000 in the workspace. The picture also shows the results in a Scope window for a 7 sec simulation. After the designer checks and accepts the result in the Scope, the scope block can be replaced with a PPMAC_Traj block, as shown in the image below:



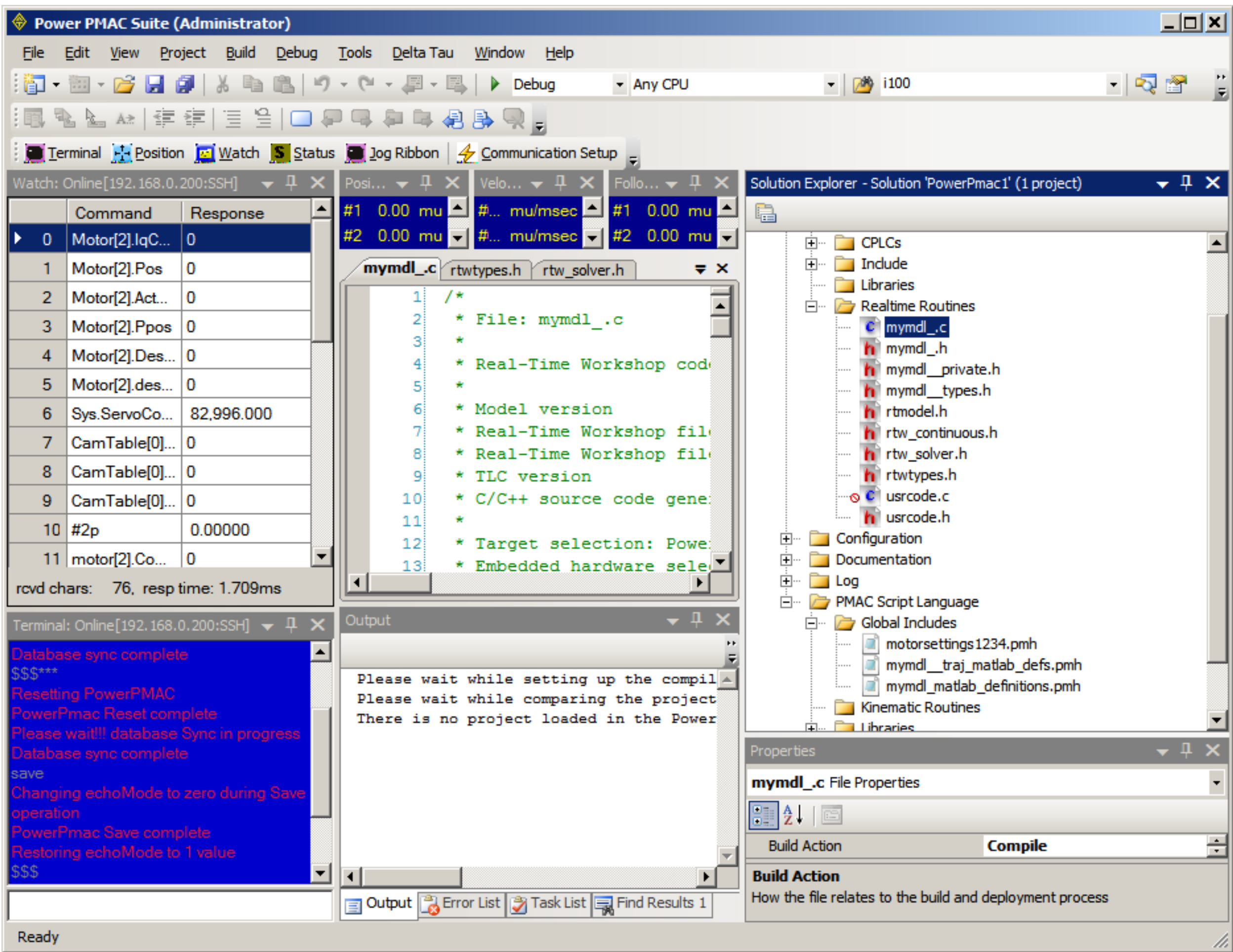


The following image shows a model which has the same Sine Wave and Gain blocks but the Scope has been replaced with a PPMAC_Traj block for which the trajectory name is set to SinTraj and the Max Time set to 7 sec. The PPMAC_INPUT and PPMAC_OUT blocks can also be used in trajectory generation models. In this example, a PPMAC_INPUT block with memory location address **Mptr->ActPos** and a PPMAC_OUT block with address **pshm->P[1]** are used. This way, the signal that is input to the PPMAC_Traj block, the desired position, is subtracted from the motor's actual position and thus the following error is written to the global variable P1, as shown below:



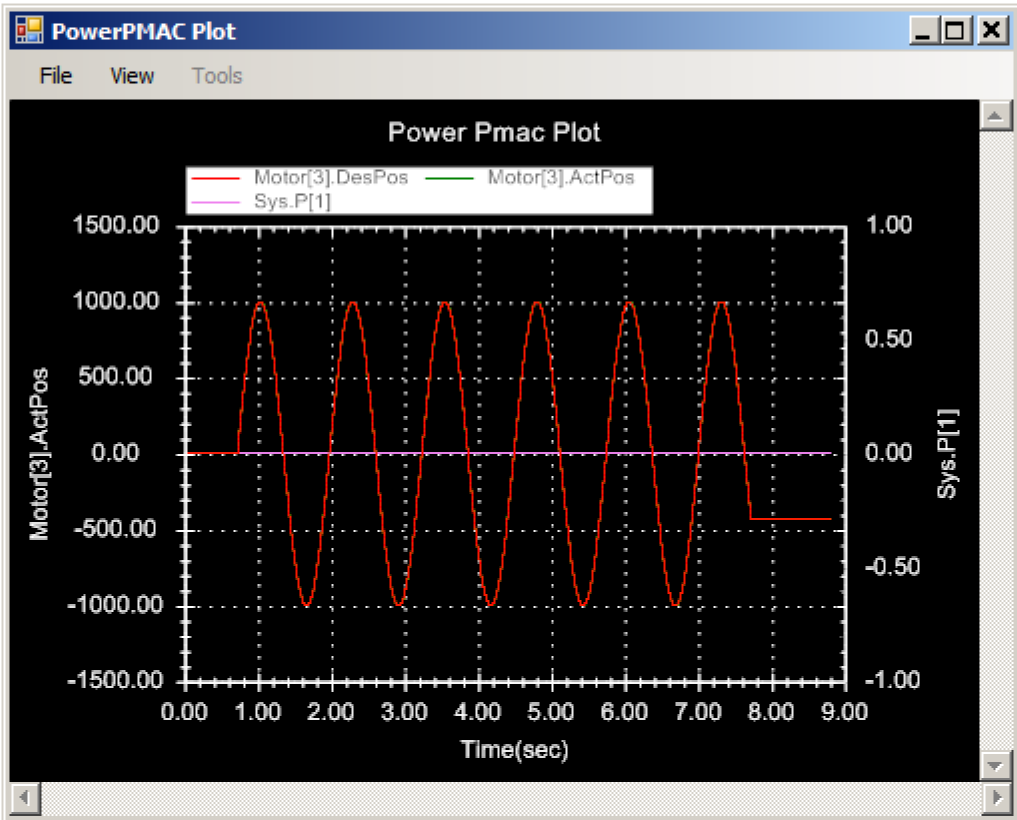
The PPMAC_Servo_RTN block cannot be used for trajectory generation models; it gives an error when compiled with the PowerPMAC_Traj.tlc system target file. Since there is a tunable parameter K1 in this

model, click on DeltaTau→Parameters→Model Validation, thereby attaching the parameter as a preload function to the model. Next, click DeltaTau→Parameters→Parameter Assignment is clicked. Leave the parameters at default and click "Create files." The new model is named as "…_param_ppmac". The three files named "mymdl_param_update.txt", "mymdl_param_initial.txt" and "mymdl_MATLAB_definitions.pmh" are created and saved in the "mymdl_param_ppmac" folder. Next, in the model named "mymdl_", which is automatically generated, select the menu item Simulation→Model Configuration Parameters→Code Generation pane, choose the system target file "PowerPMAC_Traj.tlc", and then save the model again. In the "Model Configuration" dialog box, in the "Code Generation" pane, press the "Generate code" button. The code generation report will open automatically and the generated code will be saved in a folder called "…_traj_ppmac" where "…" is the model'ss name (e.g. "mymdl__traj_ppmac" with model name of "mymdl_").
Next, create a project in the Power PMAC IDE and import the generated .c and .h files to the project. In addition to those, the two .pmh files named "…_Traj_MATLAB_Defs.pmh" and "…MATLAB_definitions.pmh," which can also be found in the "…_traj_ppmac" folder need to be added to the Power PMAC IDE's project in the Script Language→Global Includes section. The following picture shows the files added to the project:



Before initiating the Build and Download, a virtual motor (e.g. Motor 0) could be configured to run the user-servo algorithm named "Trajectories." This motor needs to be activated (**Motor[0].ServoCtrl=1**) to run the "Trajectories" servo routine. To start running the trajectory that was named SinTraj for Motor 3 for example, set SinTraj_flag(3)=1. The trajectory will finish after 7 seconds and will automatically stop.

The tunable parameter K1 can be tuned dynamically by modifying "_K1" in the Power PMAC, through the Terminal Window or PLC, for example. The following image shows **Motor[3].DesPos** and **Motor[3].ActPos**, the desired and actual positions of motor 3, respectively, on the left axis of the plot and **Sys.P[1]** on the right axis. Due to sufficient tuning, the plots of **Motor[3].DesPos** and **Motor[3].ActPos** are almost completely overlapping. The value of **P[1]** is also zero everywhere since the error is almost zero everywhere.



APPENDIX

Application Notes

1. How to use EtherCAT slave naming – OEI Application Team- Mike Esposito

Scope

Using new naming feature in IDE 4.5 to implement Slave Names and use these for mapping PDO variable names in your project.

Overview

Here demonstrate how to use ECAT Slave Names for your project.

By default, the IDE uses the Slave Address for creating the Slave name and then using this for mapping variable names for that slave. This works fine and makes each unique. However, if you modify the Slave position in the network by adding or moving its location in the network, the Address changes, Name changes, Variable names change. So now you must modify any code in the project using this Slave and its mapping variables since the names have changed. It is possible to control the Slave Address assigned, but using a real name is here suggested as a best practice.

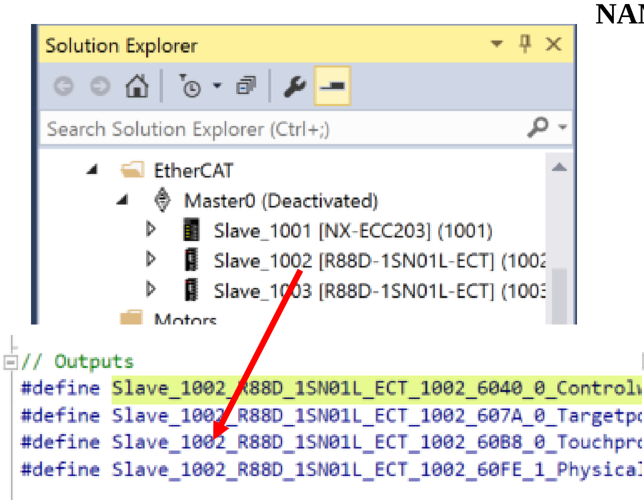
Instead of using the Slave Address for its name and variable names we can now create our own unique NAME.
Giving each slave in your ECAT network a unique slave NAME has these benefits:

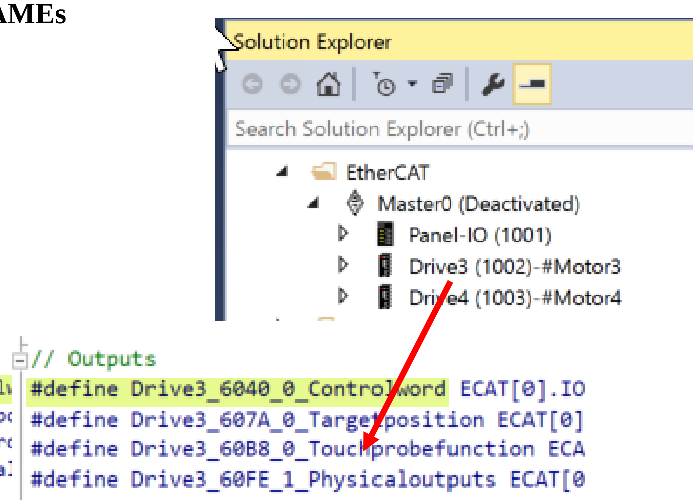
1. Creates variable mapping names that are more readable and useful in the project code.

2. If the same project later makes a change to the network (add/remove a slave) as long as the slave names are kept the same then the mapping of variables does not change. So, the project code using these names
also does NOT need to change, even though the slaves underlying ECAT registers have changed.
- this isolates your project code from the ECAT registers being assigned

Here using default ADDRESS **Here using custom**

NAMEs



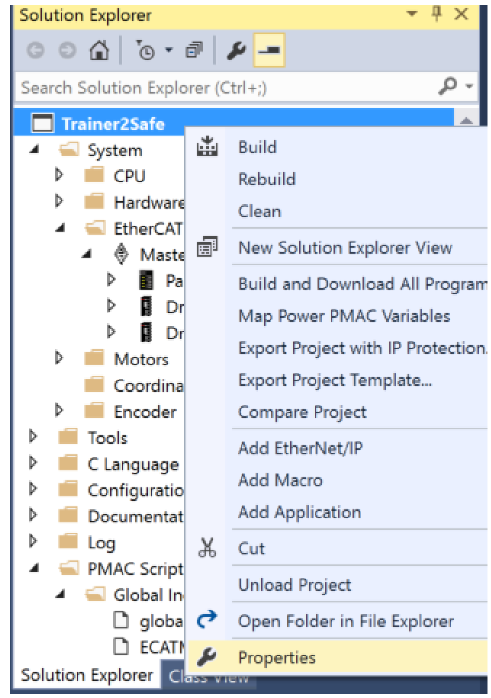


A. IDE Setup

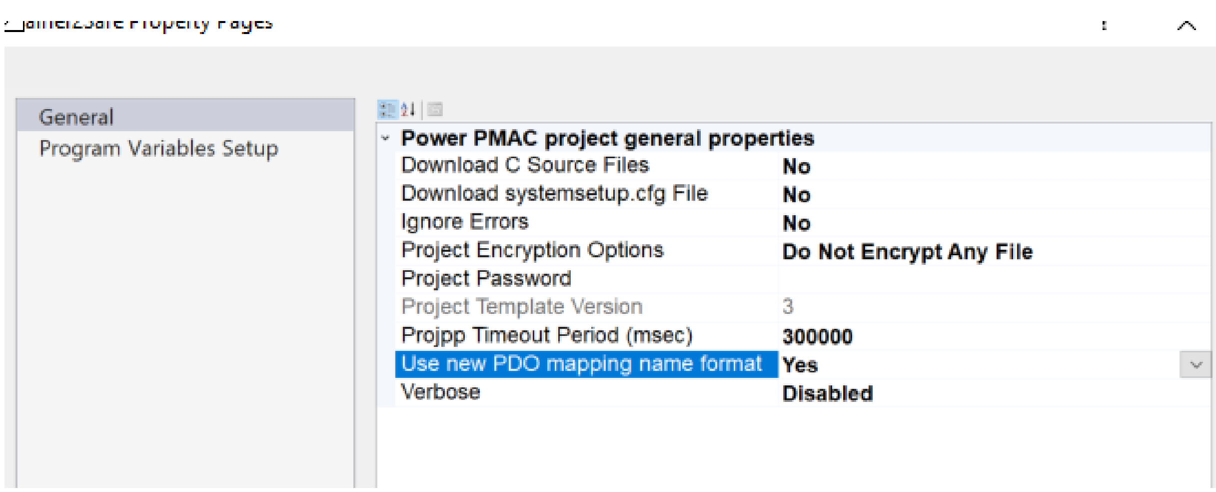
To use this new feature, you must enable 2 properties within the project. Do this before starting to setup the EtherCAT slaves.

1. Enable new PDO mapping using Names instead of Address.

a. Right-Click on the Solution Name (top of explorer tree)
- select Properties in the list (at bottom)



b. Set the Property "***Use new PDO mapping name format***" to **[YES]**
- then select **[OK]** button at bottom to save this setting

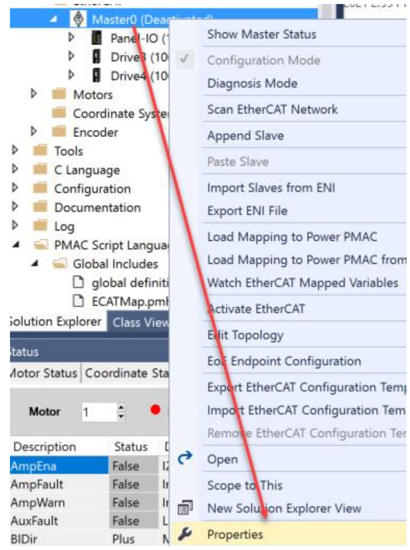


2. Remove Station Address from PDO Variable naming:

- this is done so we have only the slave NAME in PDO variables, no addresses

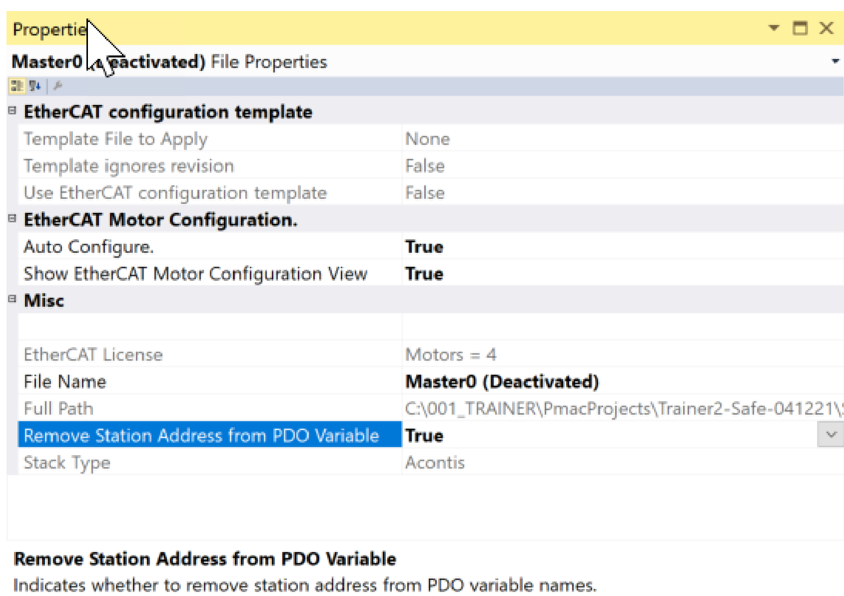
a. Right-Click on the EtherCAT Master in Tree

- select the Properties item in menu



b. In the Properties page set the "***Remove Station Address from PDO Variable***": [**True**]

- close the page with **X** in top right corner



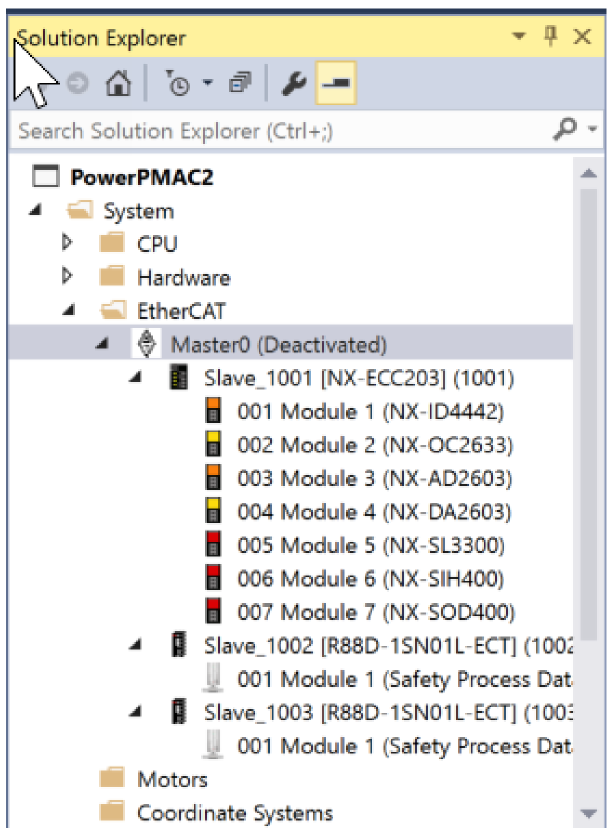
B. Example Usage

Here we show an example of using variable names instead of Address for ECAT slaves setup and PDO variable names.

1. IDE has been setup to use new Naming and Remove Address from names in PDO variables as above.

2. Scan the ECAT network and get results similar to below:

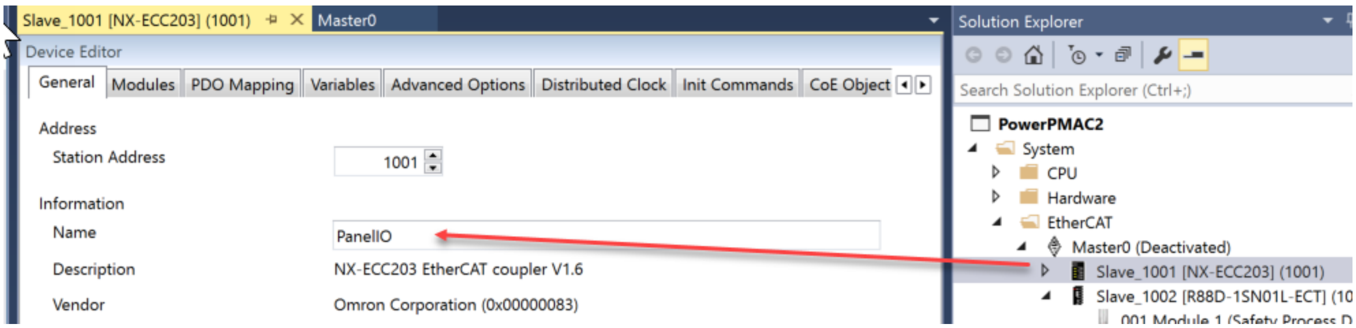
- notice the slaves are all named by default using the address based on where they are in network
- here we see Slave_1001, Module 1, 2,… Slave_1002, Slave_1003
- this is fine but has little meaning in our project
- later if the network is changed these names may change as well



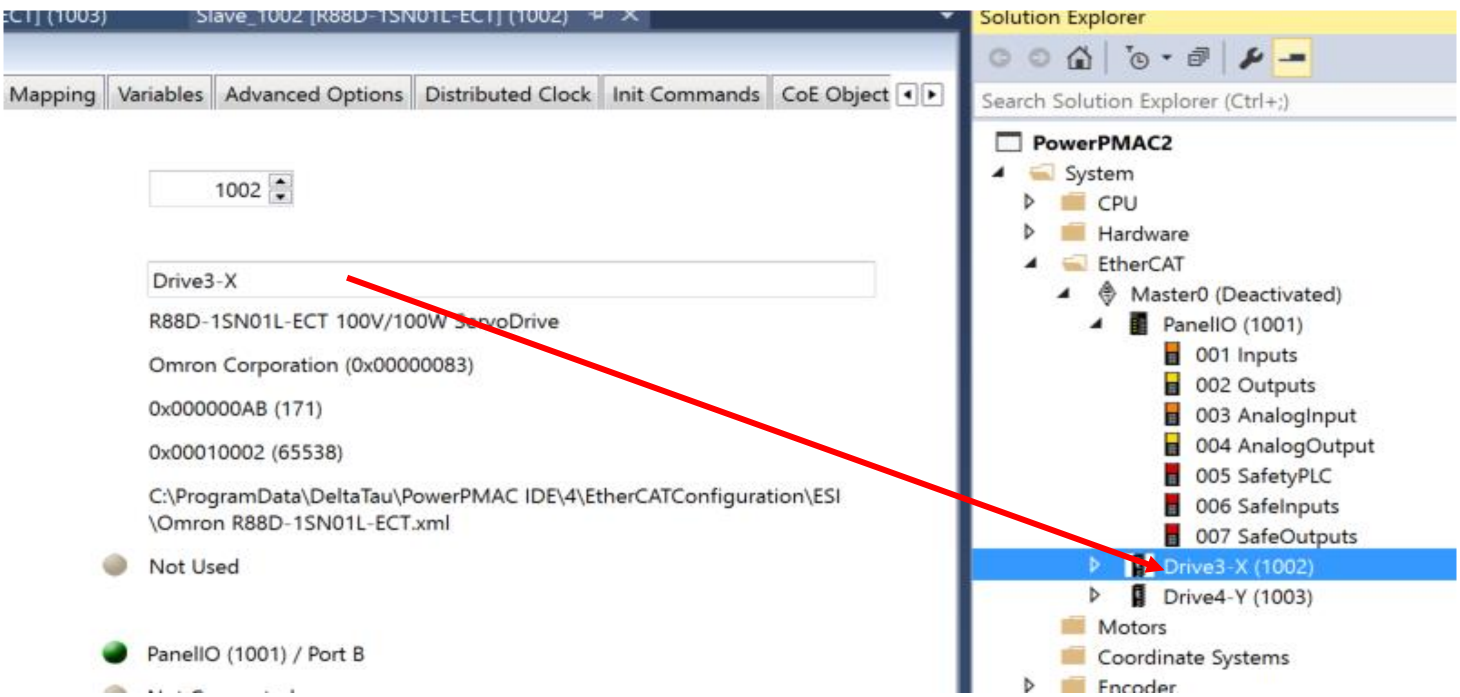
3. Select Each Slave, Double-Click to bring up Device Editor, Select the General Tab

- here in the Name field create and insert a custom name with meaning to your project
  - this is the name that will be used for PDO variable names later

- below we change the default name ***"Slave_1001 (NX-ECC203)***" to **PanelIO**



- after you finish and click out of the field you will also see the name change in the Tree



- here we can see setup names for all slaves including the ones on the coupler
- **now instead of**: Slave_1001, Module 1, 2,… Slave_1002, Slave_1003
- **we have names**: PanelIO, Inputs, Outputs,… Drive3-X, Drive4-Y

4. Now finish setup and mapping as usual

- Load Mapping to Power PMAC

- Now open the mapping file "ECATMap.pmh" auto generated by IDE
  - notice the PDO variables are all now using names instead of address:

- For example, the first Drive on the network had these PDO variable mappings by default:

```
// Outputs
#define Slave_1002_R88D_1SN01L_ECT_6040_0_Controlword ECAT[0].IO[16].Data
#define Slave_1002_R88D_1SN01L_ECT_607A_0_Targetposition ECAT[0].IO[17].Data
#define Slave_1002_R88D_1SN01L_ECT_60B8_0_Touchprobefunction ECAT[0].IO[18].Data
#define Slave_1002_R88D_1SN01L_ECT_60FE_1_Physicaloutputs ECAT[0].IO[19].Data
```

- Now the same PDO variables have these mappings:
  - these are much more meaningful and terser, for better use in your project

```
// Outputs
#define Drive3_X_6040_0_Controlword ECAT[0].IO[16].Data
#define Drive3_X_607A_0_Targetposition ECAT[0].IO[17].Data
#define Drive3_X_60B8_0_Touchprobefunction ECAT[0].IO[18].Data
#define Drive3_X_60FE_1_Physicaloutputs ECAT[0].IO[19].Data
```

5. If later the ECAT Network is changed:

- be sure to use the same names for the same slaves
- if so then the mapping will go to different ECAT registers BUT the names will be the same
- this will keep the project code that uses the variable names working as before the change

2. Commission Safety PLC (NX-SL3300 or NX-SL3500) Plus 1S servo drive with Power PMAC – OEI Application Team- Atanas Karaatanasov

Scope

How to commission Safety PLC (NX-SL3300 or NX-SL3500) with 1S servo drive under control of PMAC. This App Note does not explain how to commission 1S Servo drive, it is out of scope.

Note

The operator should have basic knowledge of Sysmac Studio and PMAC-IDE.

Note: The operator should have basic knowledge of Sysmac Studio and PMAC-IDE.
This note does not state that is complete.

Legal Note:

Limitation on Liability:

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.

SOFTWARE / hardware

PMAC-IDE ver: 4.5.x.x

Sysmac Studio ver: 1.45.x

Software / hardware

PMAC-IDE ver: 4.5.x.x

Sysmac Studio ver: 1.44.1

ITEM	NUMBER	DESCRIPTION	NOTES
1	CK3E-1310 / FW 2.6.0.0	PMAC	
2	NX-ECC203 / FW1.6	ECAT Coupler Unit	Use at least with FW1.6
3	SL3300	Safety PLC	
4	SID800	Safety Input Unit	
5	SOD400	Safety Output Unit	
6	R88D-1SN02L	1S Servo Drive / Motor	
7	R88D-1SN02L	1S Servo Drive / Motor	

Terms and Definitions

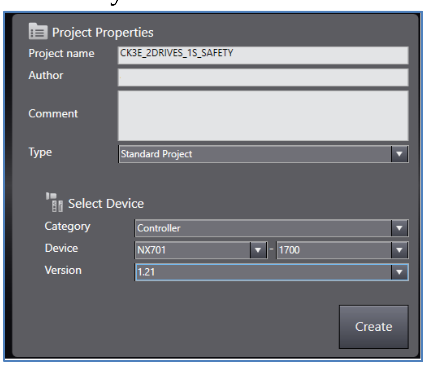
Term	Explanation and Definition
EtherCAT	Ethernet for Control Automation Technology
SLAVE	Slaves are devices connected to EtherCAT. There are various types of slaves such as servo drivers handling position data and I/O terminals handling the bit signals.

PDO Communications (Communications using Process Data Objects)	One type of EtherCAT communications in which Process Data Objects (PDOs) are used to exchange information cyclically and in real time. This is also called "process data communications".
PDO Mapping	The association of objects used for PDO communications.
ESI file (EtherCAT Slave Information file)	An ESI file contains information unique to the EtherCAT slaves in XML format. You can load ESI files into the Power PMAC IDE, to easily allocate slave process data and make other settings.
ENI file (EtherCAT Network information file)	An ENI file contains the network configuration information related to EtherCAT slaves.
PMAC IDE	This computer software is used to configure the PMAC Controller, create user programs, and monitor the programs. PMAC is an acronym for Programmable Multi-Axis Controller.

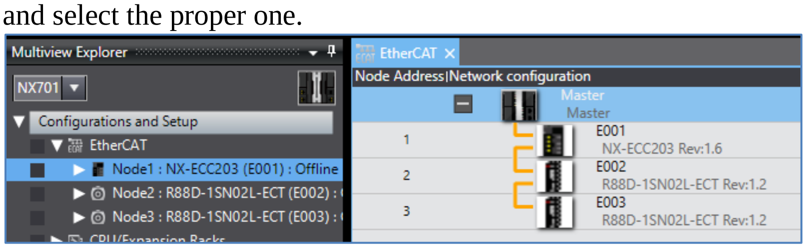
1. SYSMAC Configuration

1.1. Create new project.

Note: Any Controller can be selected.

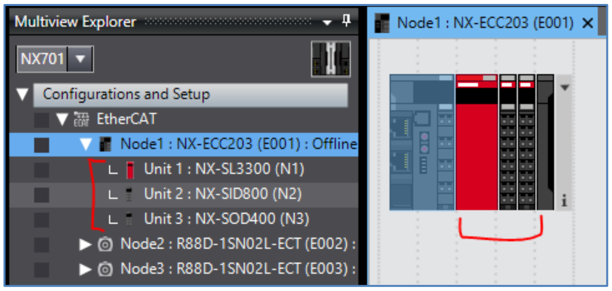


1.2. Configure ECAT devices as per your hardware. Note: Verify the firmware of the coupler (1.6 or 1.7) and select the proper one.

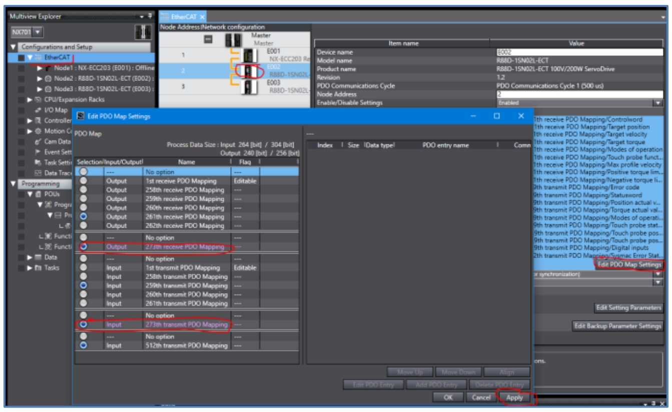


1.3. Deploy NX safety cards in EtherCAT Configuration and Setup.

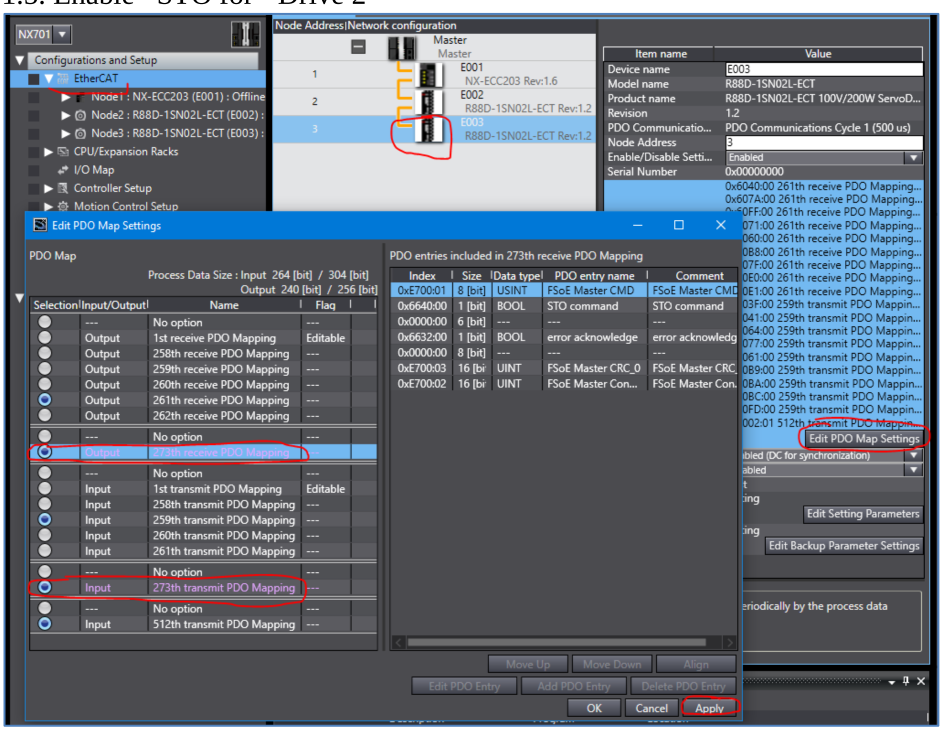
Note: Other option is to go online with the coupler and execute "Compare and Merge with Actual configuration".



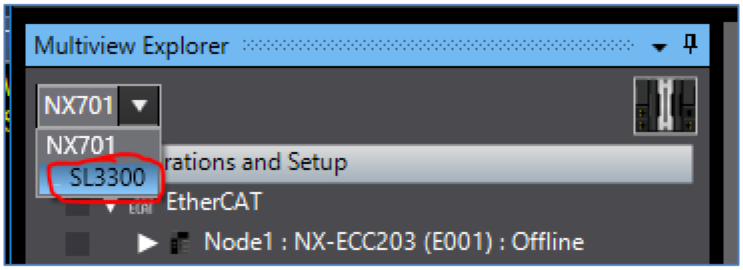
1.4. Enable STO for - Drive 1



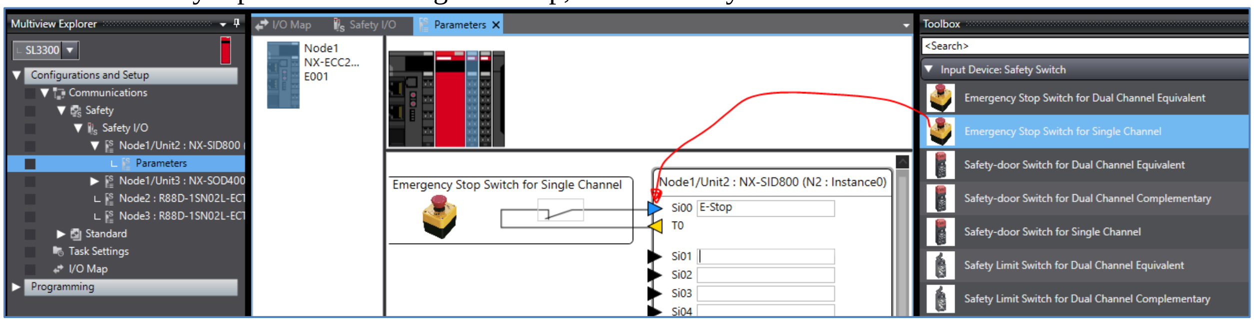
1.5. Enable STO for - Drive 2



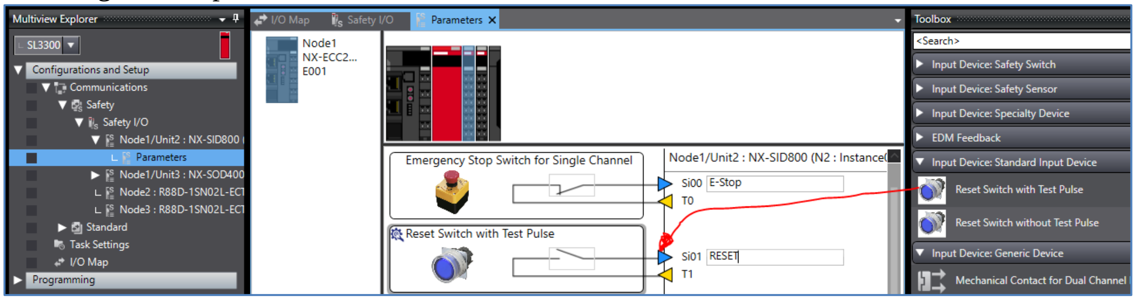
1.6. Select Safety controller



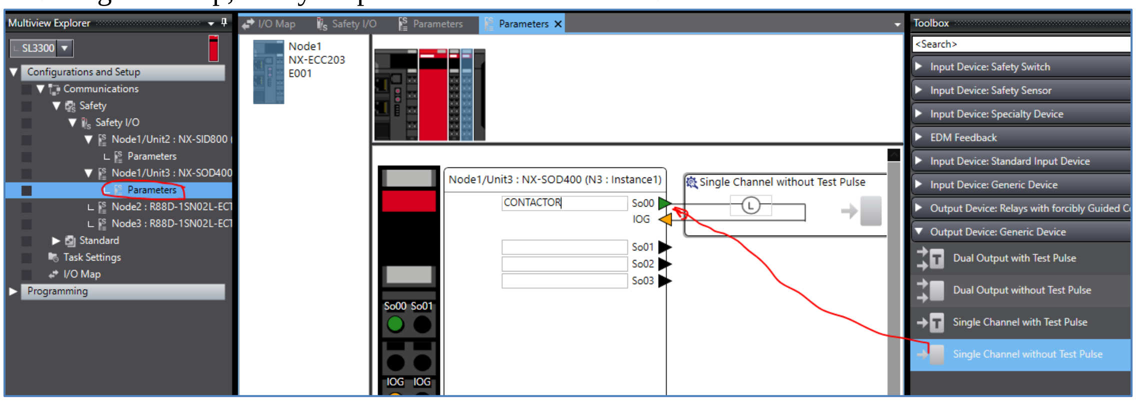
1.7. Select Safety input card and Drag-and-drop, desired safety feature:



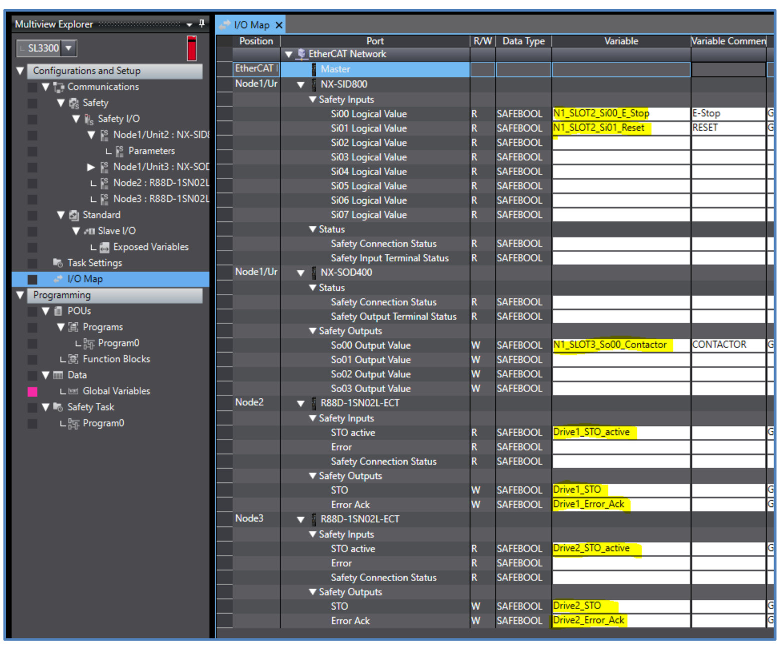
1.8. Drag-and-drop, reset button:



1.9. Drag-and-drop, safety output for external contactor:

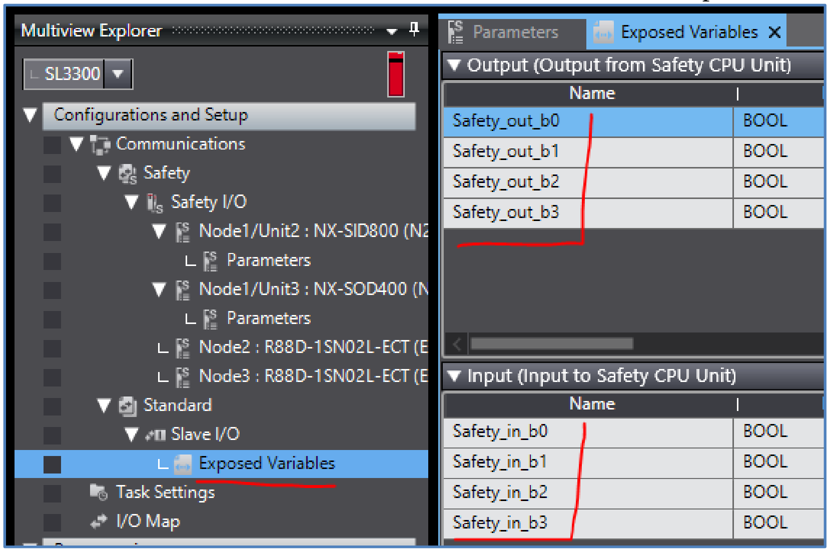


1.10. Define fallowing Variables for use in safety program

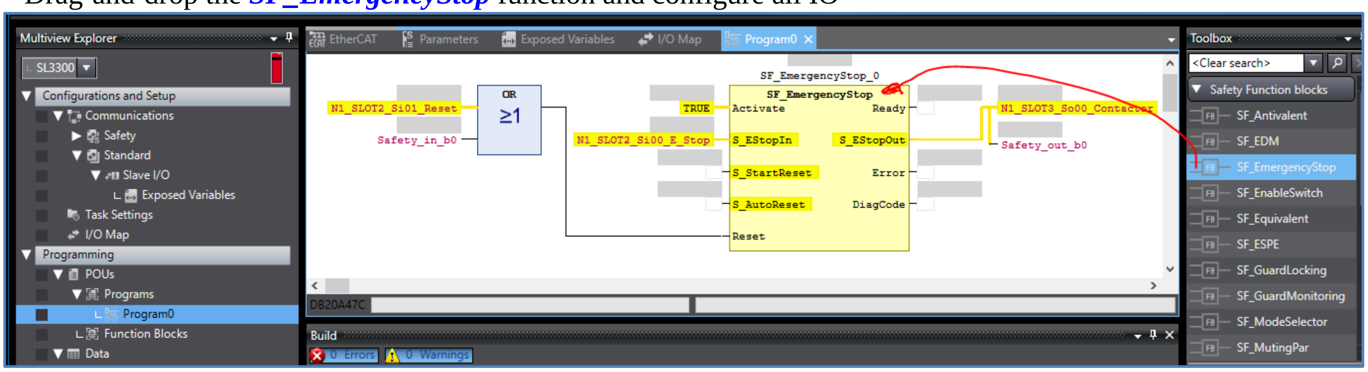


1.11. Define Exposed Variables

To exchange status and control variables between Safety PLC and PMAC. The easiest way is to use BOOL variables. Variables with other diminutions are also possible. Depend from application goals.

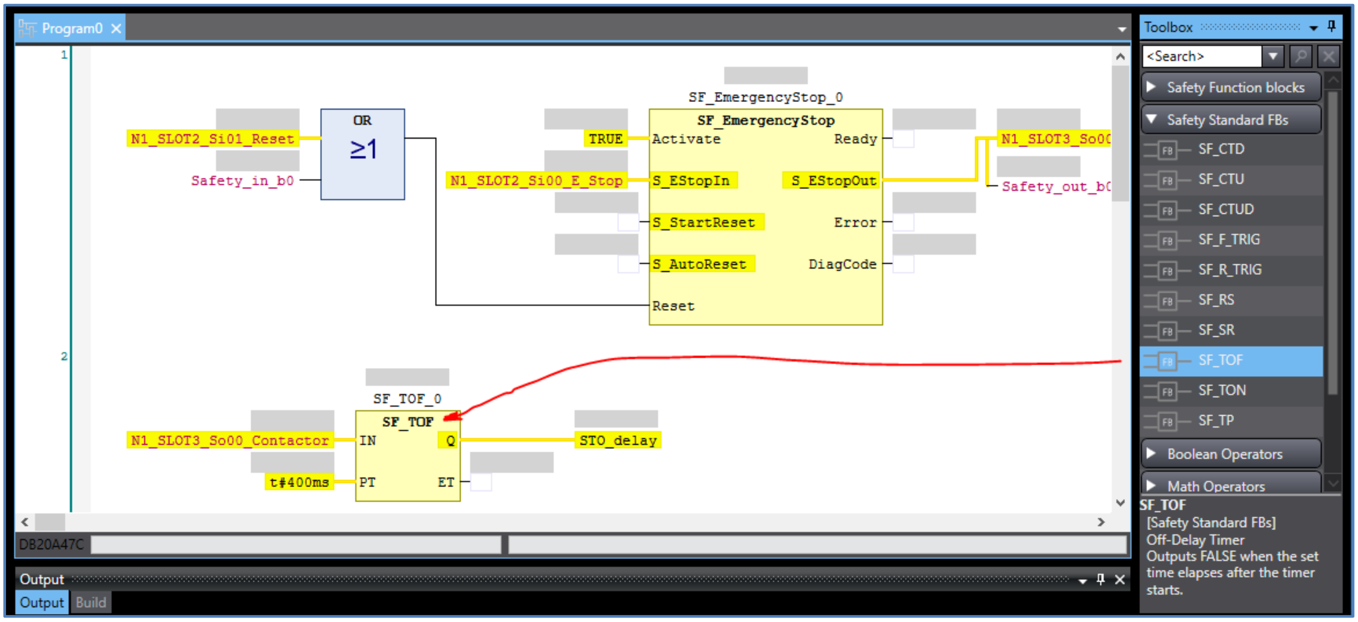


1.12. Developing the safety Program0.
Drag-and-drop the ***SF_EmergencyStop*** function and configure all IO

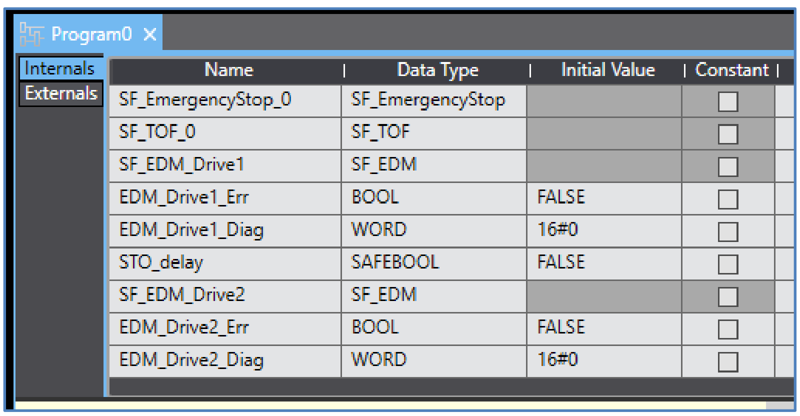


using ***Safety_in_b0***, from the PMAC, will reset the **SF_EmergencyStop** block, same as reset button.

1.13. Add 400mS delay.
If safety command is triggered in mid motion – the motion controller can stop the motor controllable and then execute STO.

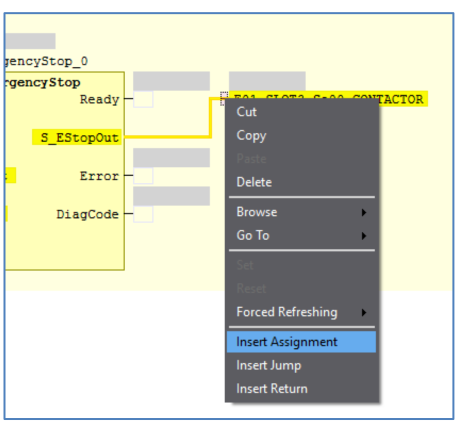


1.14. Internal variable:

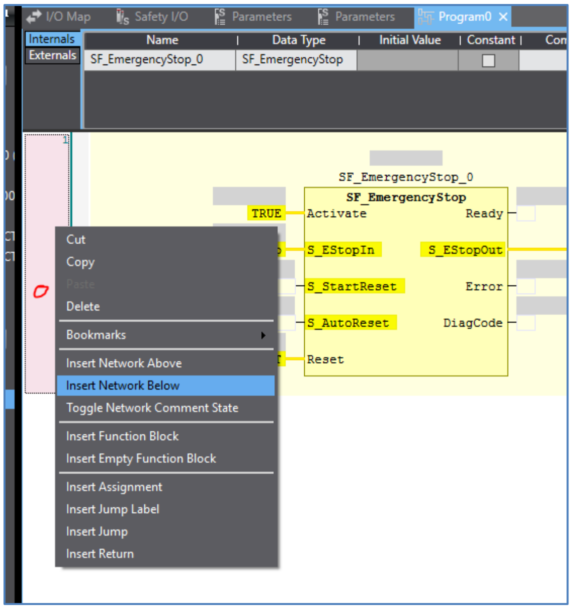


Name	Data Type	Initial Value	Constant
SF_EmergencyStop_0	SF_EmergencyStop		☐
SF_TOF_0	SF_TOF		☐
SF_EDM_Drive1	SF_EDM		☐
EDM_Drive1_Err	BOOL	FALSE	☐
EDM_Drive1_Diag	WORD	16#0	☐
STO_delay	SAFEBOOL	FALSE	☐
SF_EDM_Drive2	SF_EDM		☐
EDM_Drive2_Err	BOOL	FALSE	☐
EDM_Drive2_Diag	WORD	16#0	☐

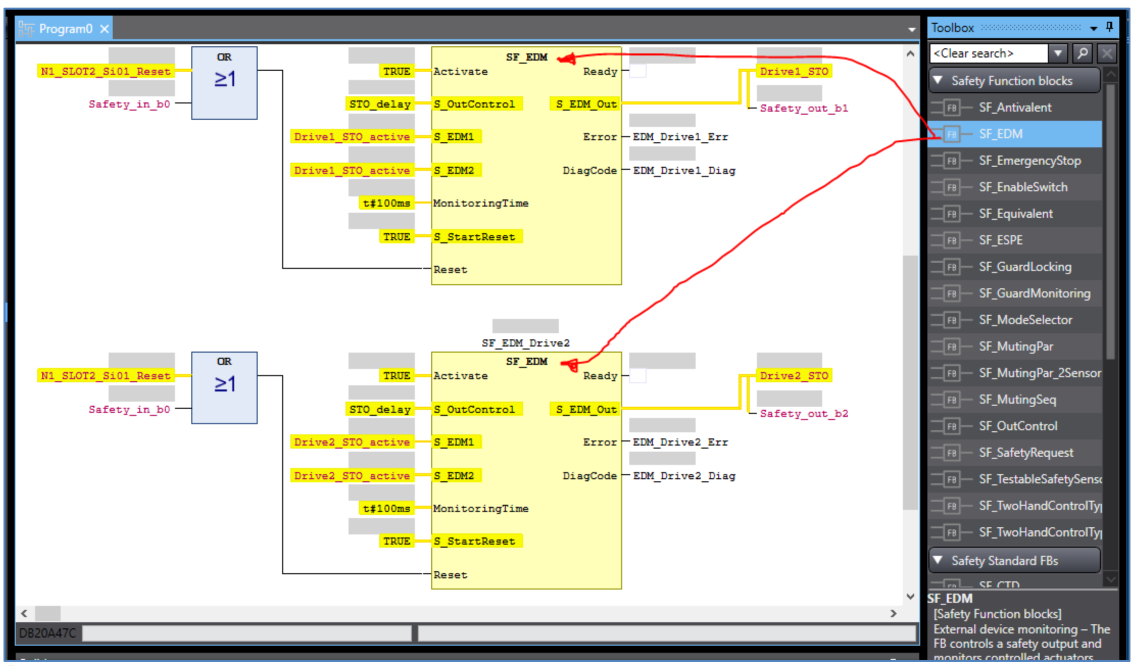
1.15. Right click to assign S_EStopOut signal - Insert Assignment



1.16. Right click and - Insert Network Below

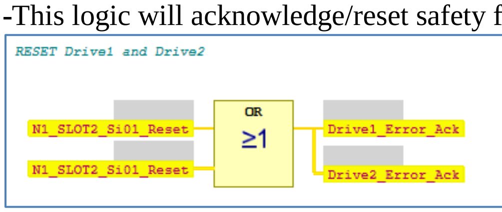


1.17. Define EDM for STO @Drive1 and @Drive2



Note: using ***Safety_in_b0***, from the PMAC, will reset the **SF_EDM** block same as reset button.

1.18. Apply additional logic if needed
-This logic will acknowledge/reset safety faults in drives.

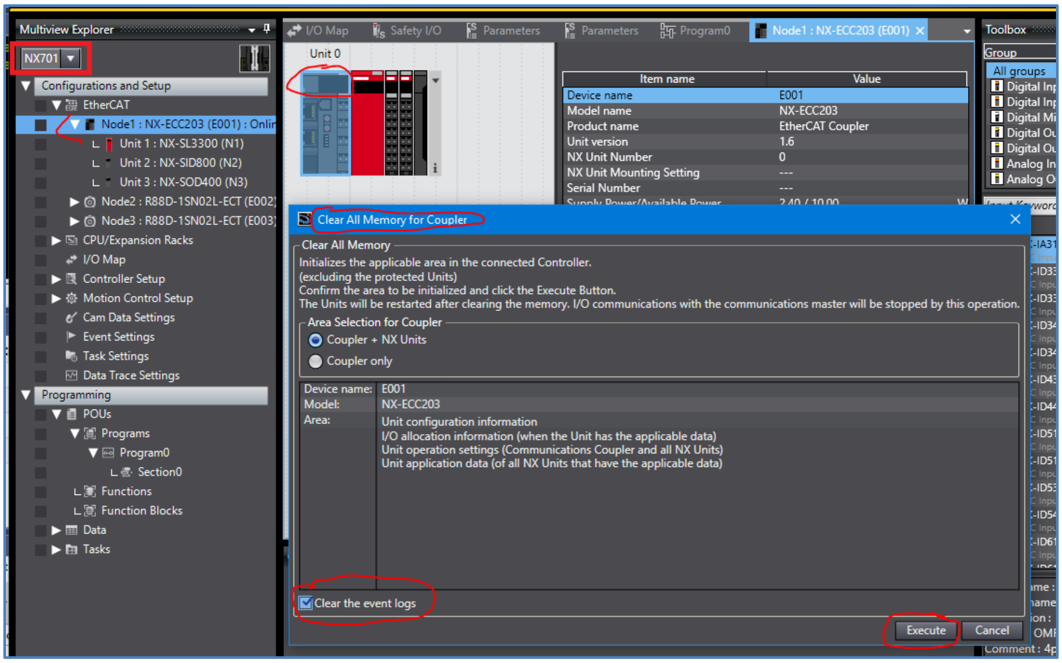


2. Download SYSMAC project to ECC203 and sl3300

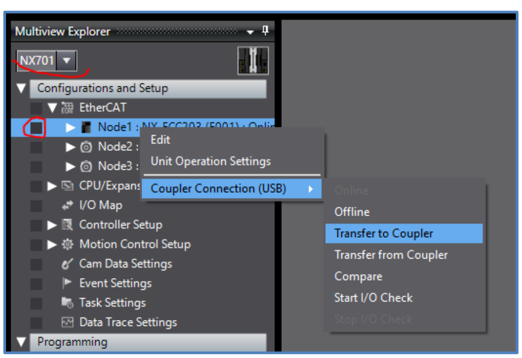
2.1. Online with ECC203

Through USB port. Right click on ECC203 to clear all memory

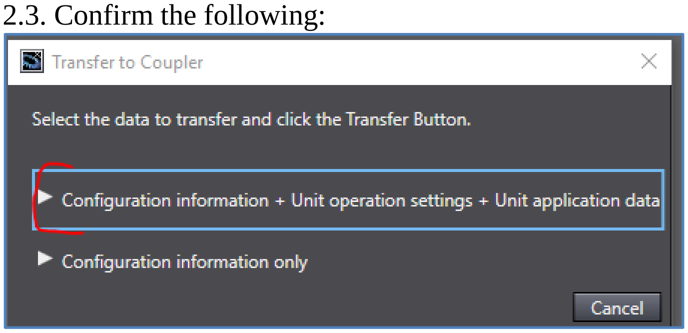
ECC203 use type B for USB connection



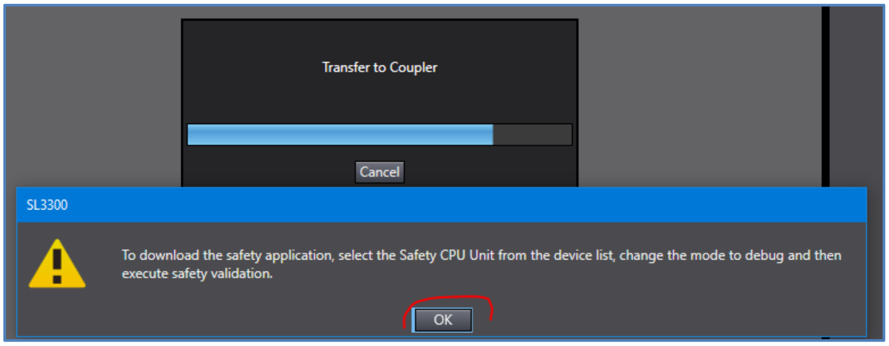
2.2. Transfer configuration to the ECC203 coupler



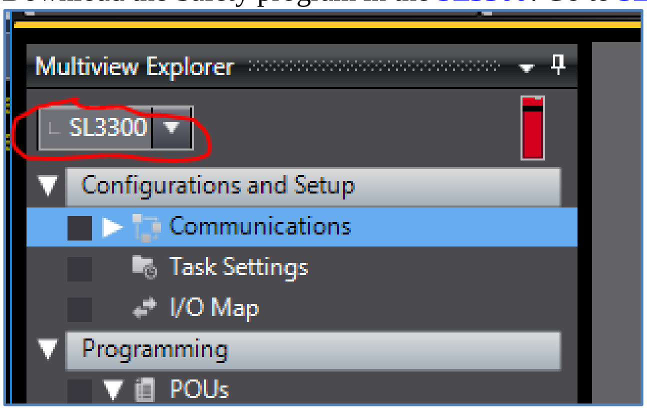
2.3. Confirm the following:



2.4. Confirm with OK

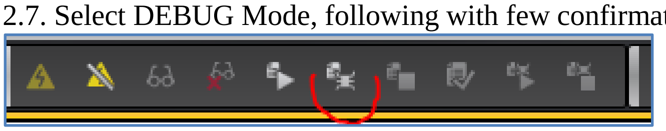


2.5. The final step
Download the Safety program in the ***SL3300***. Go to ***SL3300*** from the explorer while still online.

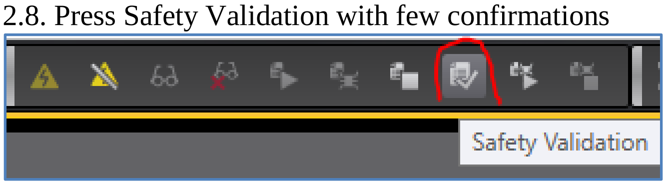


2.6. Stop Safety PLC following with few confirmations:

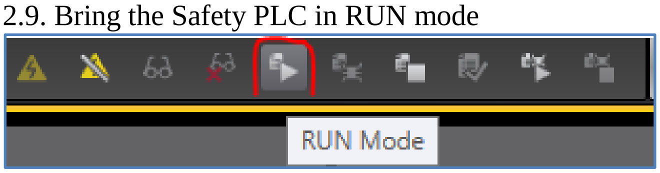
2.7. Select DEBUG Mode, following with few confirmations



2.8. Press Safety Validation with few confirmations



2.9. Bring the Safety PLC in RUN mode

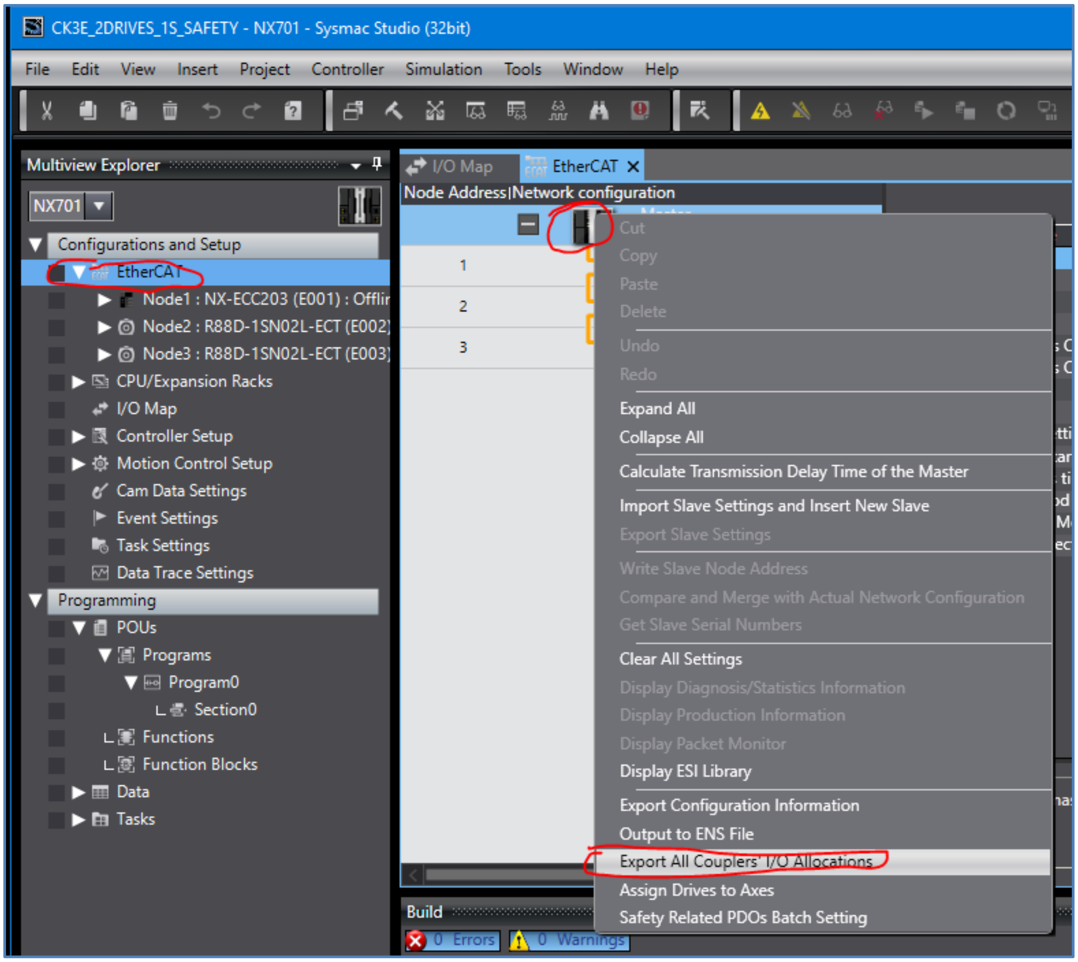


With this STEP, safety configuration in the SL3300 is complete.

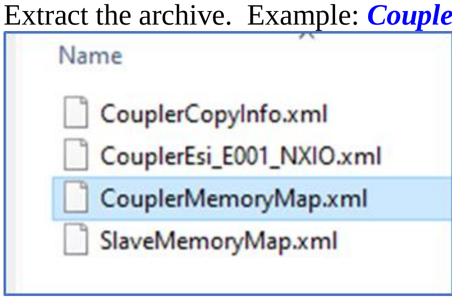
3. Export sysmac pdo configuration

3.1. Sysmac PDO configuration need to be migrate to the PMAC-IDE.
To do that the first step is to extract PDO configuration for the coupler from Sysmac Studio. Right click on the Master and Export All Couplers I/O Allocations

To execute that command ECC203 need to be in offline mode. (not seeing the orange “online” bar on top of Sysmac)



3.2. Save the ZIP file in desire directory.
Extract the archive. Example: ***Coupler_20210406_105541.zip***



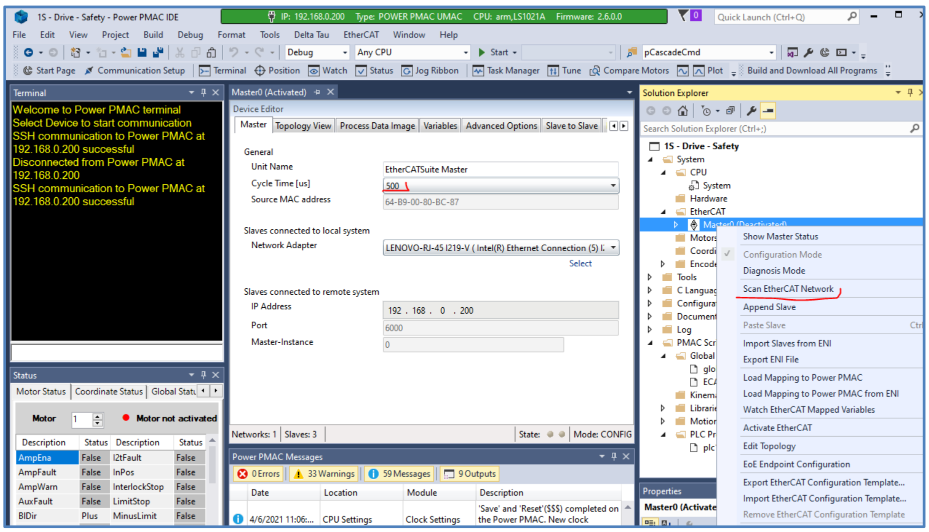
3.3. Open CouplerMemoryMap.xml
File with Internet Explorer (IE11), and not with Google Chrome - just to visualize the telegrams. This page shows the important PDOs related with Safety.

Content should look like this:
yellow – safety
blue – Non-safety – Status\Control communication with PMAC
red – PDO section ***Input Data set 1,2*** and ***Output Data set 1,2***

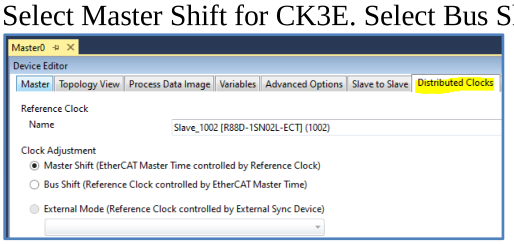
4. Power PMAC IDE configuration

4.1. Reset & Re-Initialize PMAC.

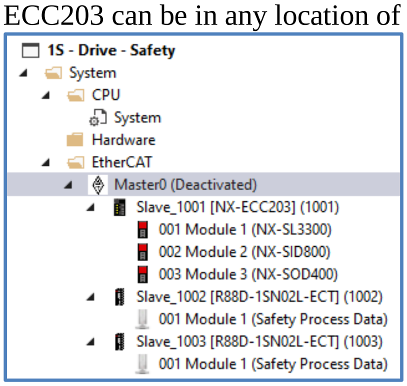
"$$$***". Scan EtherCAT Network.



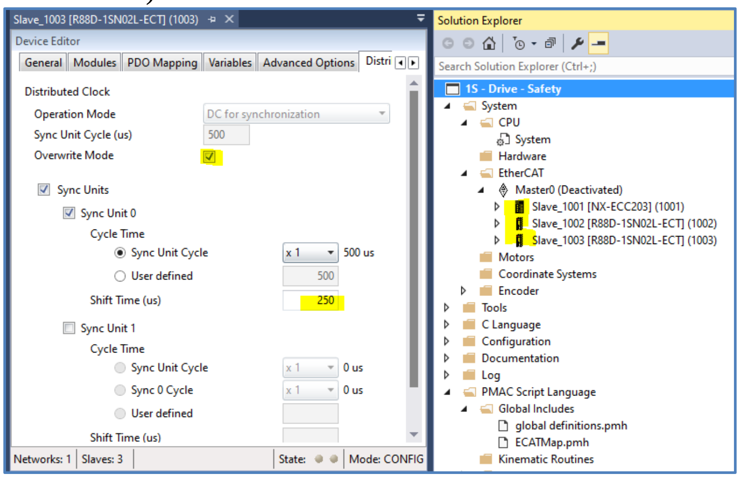
Select Master Shift for CK3E. Select Bus Shift for CK3M with Gate3



4.2. Result:
ECC203 can be in any location of the ECAT network: beginning, middle, end.

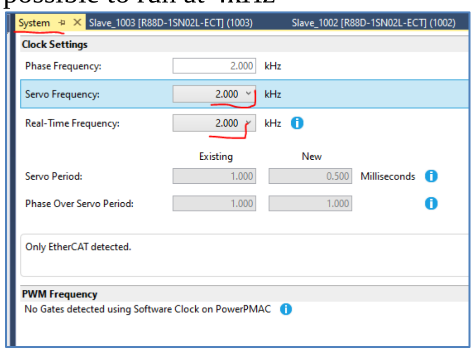


4.3. Select DC for synchronization. Set “Shift Time” 125uS -250uS for all 3 ECAT devices (ECC203 and 2 drives)

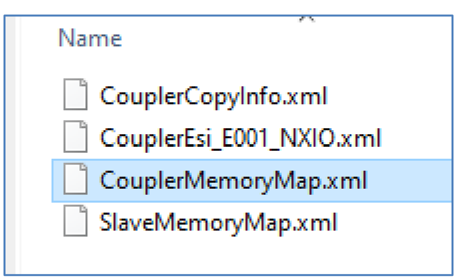


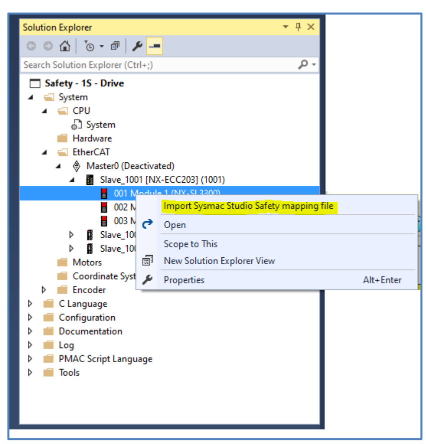
4.4. Set CPU speed @ 2kHz

Safety program with PMAC was tested up to 2kHz with Dual Core ARM. With Quad Core is possible to run at 4kHz

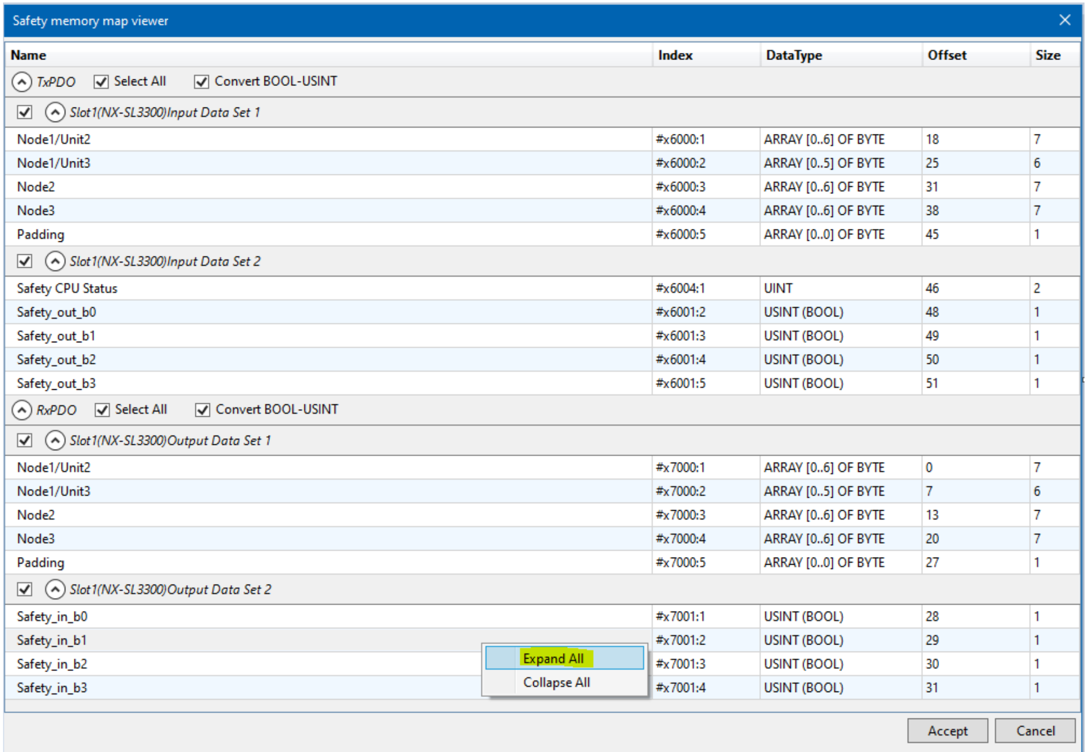


4.5. Import Sysmac safety PDO map file: (Right Click SL3300) CouplerMemoryMap.xml





4.6. Safety memory map viewer should look like this. Leave ***Select All*** selected and click ***Accept***

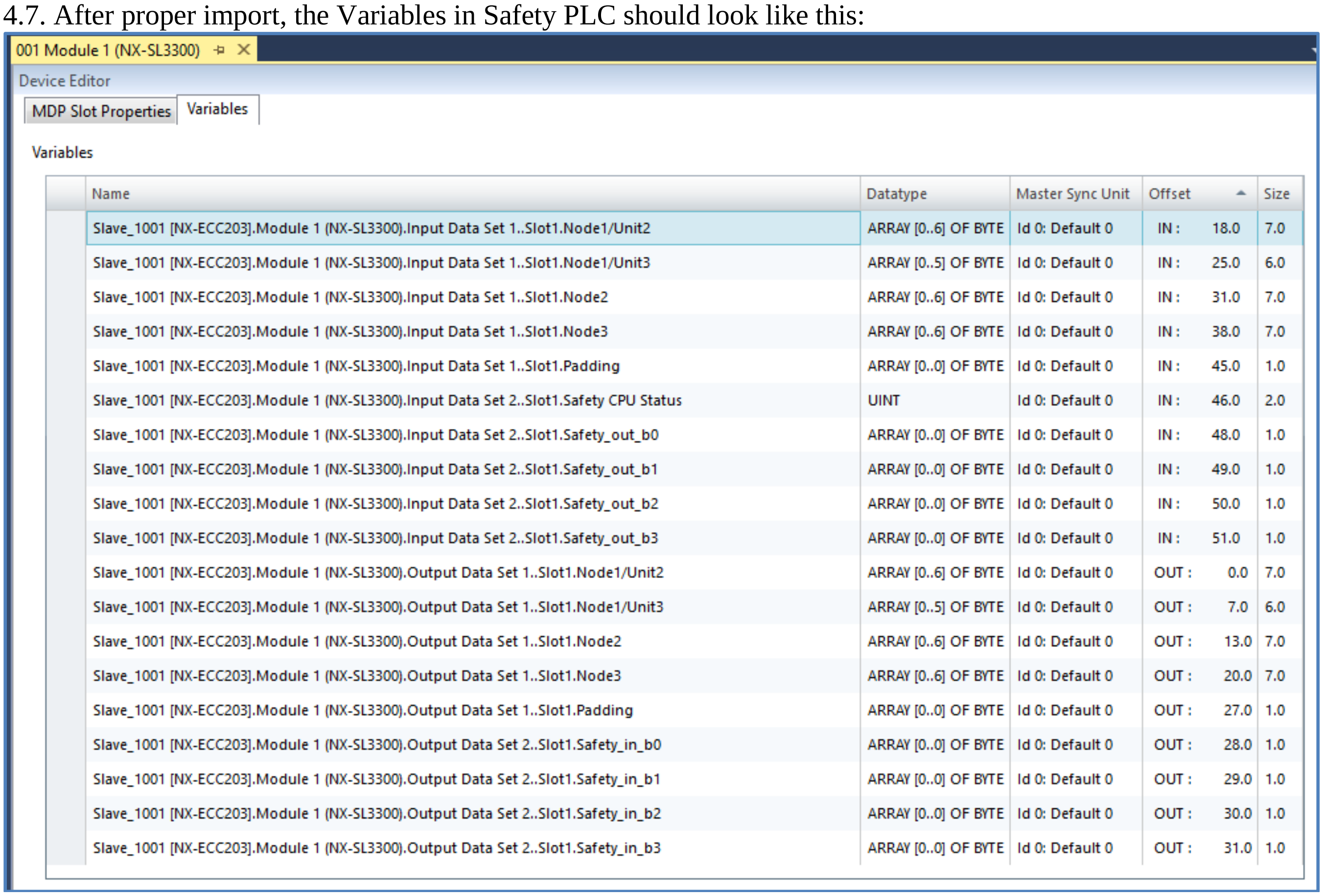


Safety memory map viewer

Name	Index	DataType	Offset	Size
TxPDO Select All Convert BOOL-USINT				
Slot1(NX-SL3300)Input Data Set 1				
Node1/Unit2	#x6000:1	ARRAY [0..6] OF BYTE	18	7
Node1/Unit3	#x6000:2	ARRAY [0..5] OF BYTE	25	6
Node2	#x6000:3	ARRAY [0..6] OF BYTE	31	7
Node3	#x6000:4	ARRAY [0..6] OF BYTE	38	7
Padding	#x6000:5	ARRAY [0..0] OF BYTE	45	1
Slot1(NX-SL3300)Input Data Set 2				
Safety CPU Status	#x6004:1	UINT	46	2
Safety_out_b0	#x6001:2	USINT (BOOL)	48	1
Safety_out_b1	#x6001:3	USINT (BOOL)	49	1
Safety_out_b2	#x6001:4	USINT (BOOL)	50	1
Safety_out_b3	#x6001:5	USINT (BOOL)	51	1
RxPDO Select All Convert BOOL-USINT				
Slot1(NX-SL3300)Output Data Set 1				
Node1/Unit2	#x7000:1	ARRAY [0..6] OF BYTE	0	7
Node1/Unit3	#x7000:2	ARRAY [0..5] OF BYTE	7	6
Node2	#x7000:3	ARRAY [0..6] OF BYTE	13	7
Node3	#x7000:4	ARRAY [0..6] OF BYTE	20	7
Padding	#x7000:5	ARRAY [0..0] OF BYTE	27	1
Slot1(NX-SL3300)Output Data Set 2				
Safety_in_b0	#x7001:1	USINT (BOOL)	28	1
Safety_in_b1	#x7001:2	USINT (BOOL)	29	1
Safety_in_b2	#x7001:3	USINT (BOOL)	30	1
Safety_in_b3	#x7001:4	USINT (BOOL)	31	1

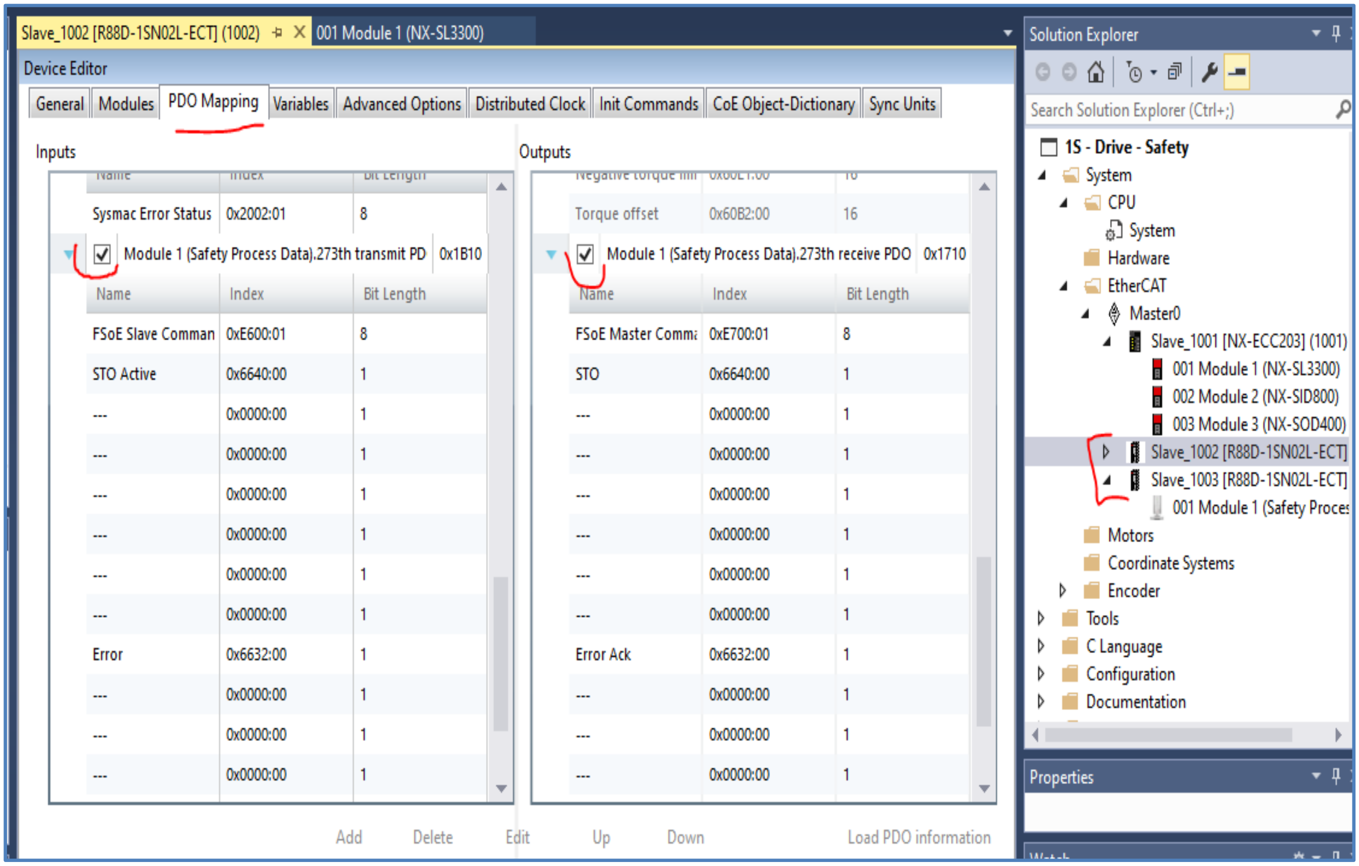
Leave ***Convert BOOL-USINT*** selected.

4.7. After proper import, the Variables in Safety PLC should look like this:



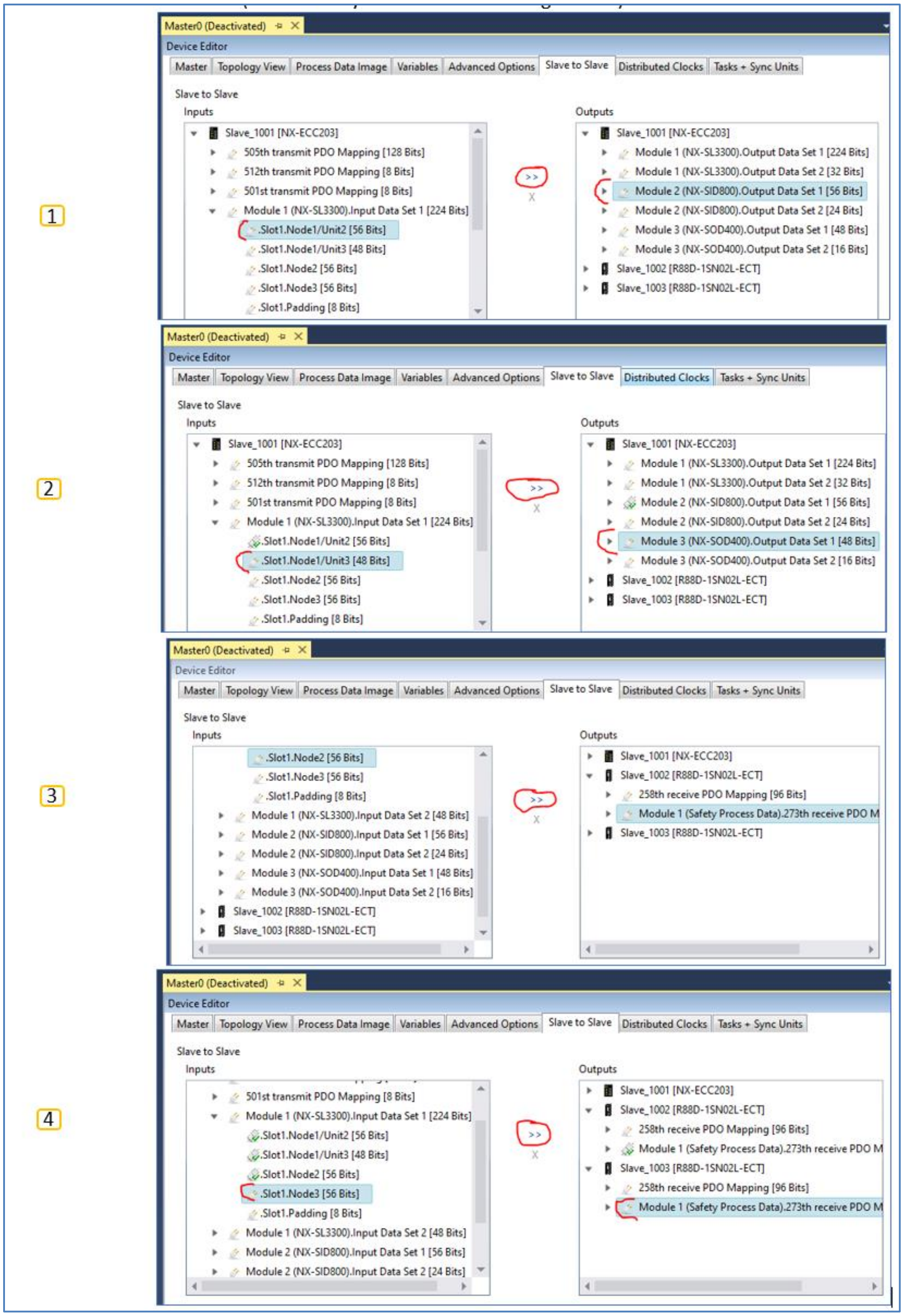
Name	Datatype	Master Sync Unit	Offset	Size
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node1/Unit2	ARRAY [0..6] OF BYTE	Id 0: Default 0	IN : 18.0	7.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node1/Unit3	ARRAY [0..5] OF BYTE	Id 0: Default 0	IN : 25.0	6.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node2	ARRAY [0..6] OF BYTE	Id 0: Default 0	IN : 31.0	7.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node3	ARRAY [0..6] OF BYTE	Id 0: Default 0	IN : 38.0	7.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Padding	ARRAY [0..0] OF BYTE	Id 0: Default 0	IN : 45.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 2..Slot1.Safety CPU Status	UINT	Id 0: Default 0	IN : 46.0	2.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 2..Slot1.Safety_out_b0	ARRAY [0..0] OF BYTE	Id 0: Default 0	IN : 48.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 2..Slot1.Safety_out_b1	ARRAY [0..0] OF BYTE	Id 0: Default 0	IN : 49.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 2..Slot1.Safety_out_b2	ARRAY [0..0] OF BYTE	Id 0: Default 0	IN : 50.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 2..Slot1.Safety_out_b3	ARRAY [0..0] OF BYTE	Id 0: Default 0	IN : 51.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node1/Unit2	ARRAY [0..6] OF BYTE	Id 0: Default 0	OUT : 0.0	7.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node1/Unit3	ARRAY [0..5] OF BYTE	Id 0: Default 0	OUT : 7.0	6.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node2	ARRAY [0..6] OF BYTE	Id 0: Default 0	OUT : 13.0	7.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node3	ARRAY [0..6] OF BYTE	Id 0: Default 0	OUT : 20.0	7.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Padding	ARRAY [0..0] OF BYTE	Id 0: Default 0	OUT : 27.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 2..Slot1.Safety_in_b0	ARRAY [0..0] OF BYTE	Id 0: Default 0	OUT : 28.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 2..Slot1.Safety_in_b1	ARRAY [0..0] OF BYTE	Id 0: Default 0	OUT : 29.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 2..Slot1.Safety_in_b2	ARRAY [0..0] OF BYTE	Id 0: Default 0	OUT : 30.0	1.0
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 2..Slot1.Safety_in_b3	ARRAY [0..0] OF BYTE	Id 0: Default 0	OUT : 31.0	1.0

4.8. On each drive (Inputs / Outputs) Safety Process Data with telegram ***273th*** need to be selected.



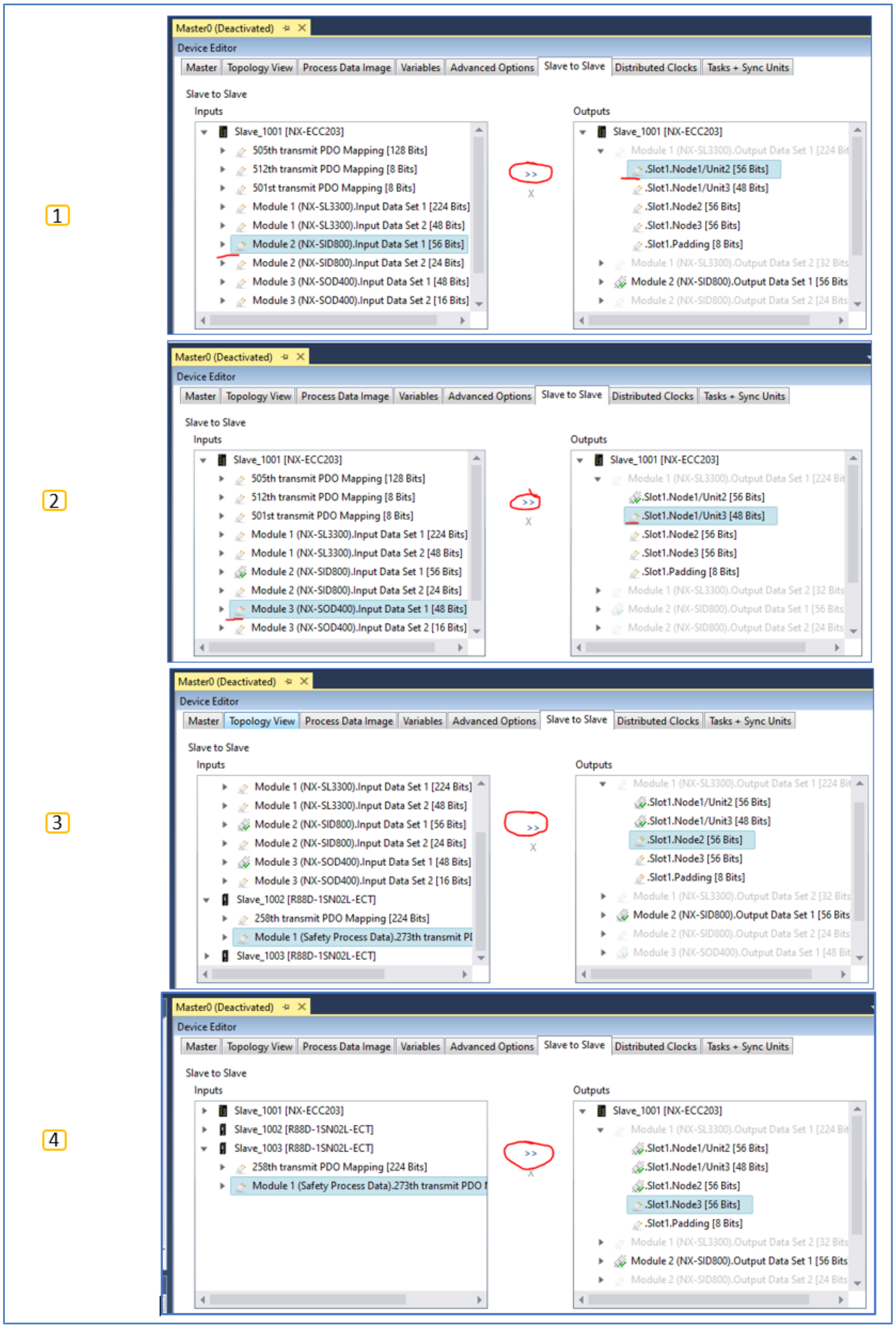
In this example PDO 258th for position close loop control is used.

4.9. When PDO is complete, Slave to Slave communication need to be establish:- 4 Connection for INPUTs - (this will vary with different configuration)



Note: Every time when modifying ECAT network ***Slave to Slave*** need to be ***Disconnected*** and ***Connected*** again.

4.10. / 4 Connections for OUTPUTs - (this will vary if configuration is different)

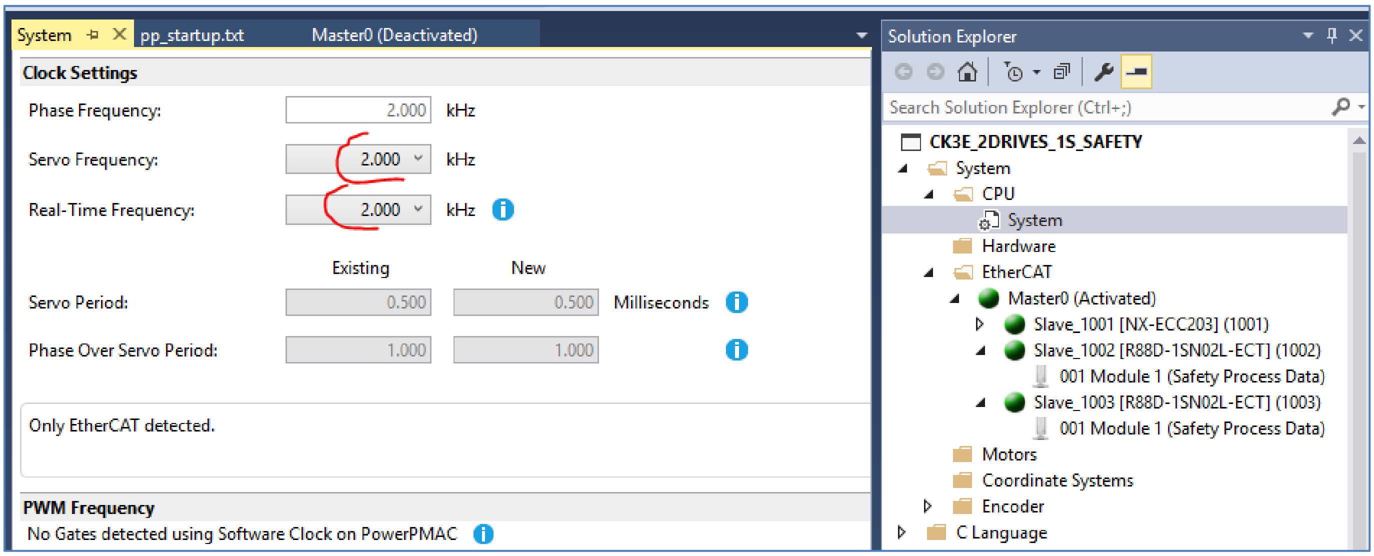


Note: Every time when modifying ECAT network ***Slave to Slave*** need to be ***Disconnected*** and ***Connected*** again.

4.11. When completed, Connections menu should look like this:

Input	Offset		Output	Offset	BitSize
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node1/Unit2	18.0	>>	Slave_1001 [NX-ECC203].Module 2 (NX-SID800).Output Data Set 1	32.0	56
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node1/Unit3	25.0	>>	Slave_1001 [NX-ECC203].Module 3 (NX-SOD400).Output Data Set 1	42.0	48
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node2	31.0	>>	Slave_1002 [R88D-1SN02L-ECT].Module 1 (Safety Process Data).273th receive PDO Mapping	82.0	56
Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Input Data Set 1..Slot1.Node3	38.0	>>	Slave_1003 [R88D-1SN02L-ECT].Module 1 (Safety Process Data).273th receive PDO Mapping	117.0	56
Slave_1001 [NX-ECC203].Module 2 (NX-SID800).Input Data Set 1	52.0	>>	Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node1/Unit2	0.0	56
Slave_1001 [NX-ECC203].Module 3 (NX-SOD400).Input Data Set 1	62.0	>>	Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node1/Unit3	7.0	48
Slave_1002 [R88D-1SN02L-ECT].Module 1 (Safety Process Data).273th transmit PDO Mapping	98.0	>>	Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node2	13.0	56
Slave_1003 [R88D-1SN02L-ECT].Module 1 (Safety Process Data).273th transmit PDO Mapping	133.0	>>	Slave_1001 [NX-ECC203].Module 1 (NX-SL3300).Output Data Set 1..Slot1.Node3	20.0	56

4.12. Check the CPU clock to match the selected 2kHz for ECAT master

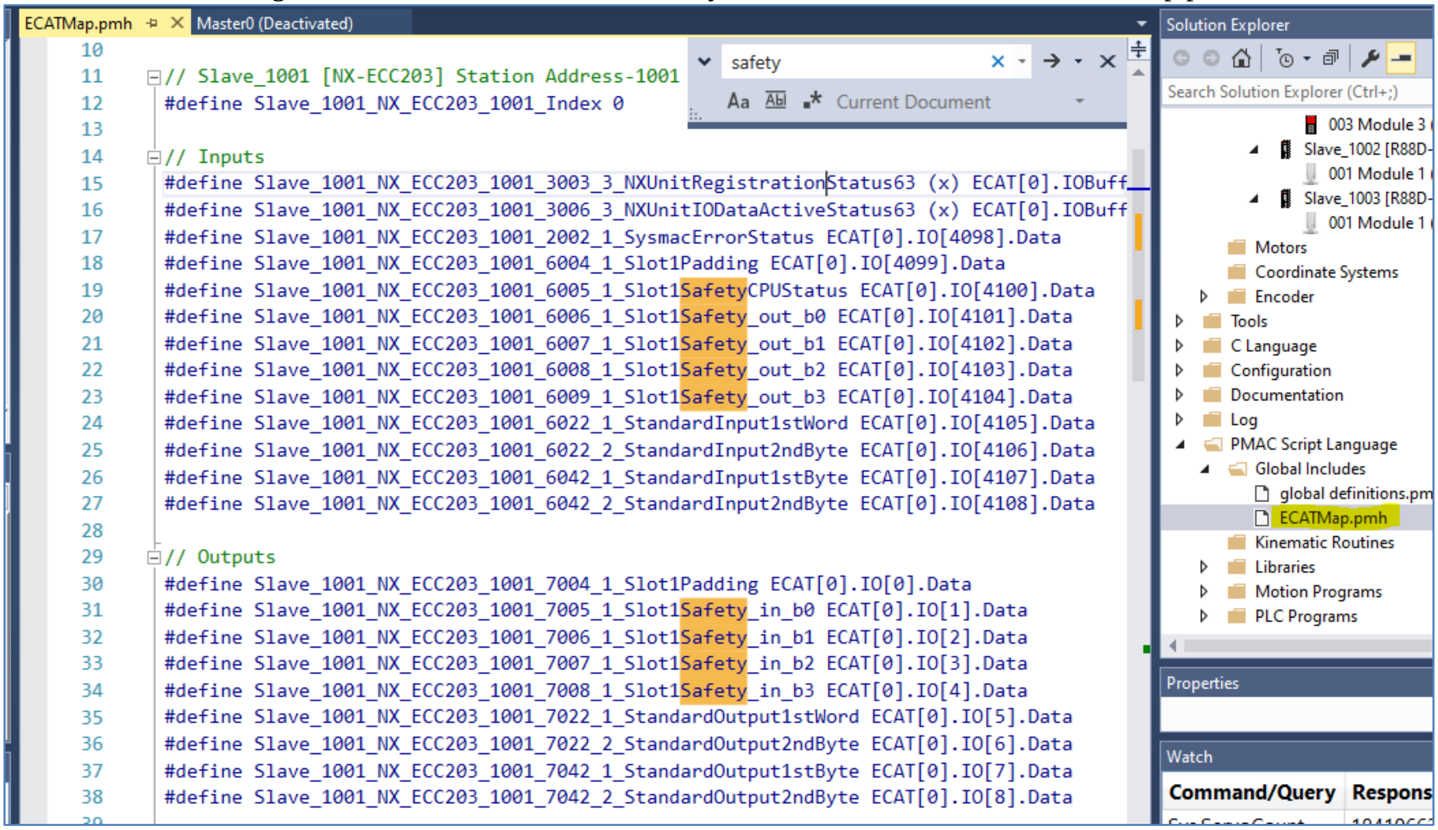


4.13. Load Mapping to PowerPMAC .
SAVE(terminal) $$$ (terminal)

Enable the ECAT with command: “ECAT[0].enable=1”

When RESET button is pressed, the CONTACTOR should enable and drives should remove STO (“St” on LED display) and go to normal operation (“—“ on LED display).

4.14. If status bits - diagnostic data, is needed in the PMAC you can find the variables in ECATMap.pmh



Upgrading project from IDEV3.x to IDEV4.x

Use Case 1

This case assumes that the project in IDE3.x is created with the complete Power PMAC setting stored as **.cfg** file in the Project configuration folder. For example, let us call this file 'MyGoodConfigFile.cfg.'

This file includes the following:

- Motor structure element
- Coordinate system structure element
- Gate structure element
- Custom initialize element

1. Open IDE4.x and select Open project.
2. Choose the project that is created using V3.x IDE.
3. On opening the project, a message will be displayed that the project is a 'One-way upgrade' process. A success message will display that the project has been upgraded successfully.
4. On successful downloading of the V3.x project, download the 'MyGoodConfigFile.cfg' from configuration folder. Once this is complete the device is now ready.

As explained earlier V4.x will add System, Hardware, Motor, Coordinate and EtherCAT nodes to the V3.x project. Other than the hardware node, most of the nodes are empty as this is an upgrade project from v3.x.

Note	IDE V4.0.x will always download the SystemSetup.cfg file on Build and Download. The Automatic management property 'Download Systemsetup.cfg file' is set to ***Yes*** by default.
Note	IDE V4.1.x will automatically set the Automatic management property 'Download Systemsetup.cfg file' to ***No.*** Automatic management of this file is OFF and user will need to set the Project property to **Yes**

This process is the recommended way of upgrading the V3.x project to V4.0 and above.

Use Case 2

This case assumes that the user is not using conventional recommended way of creating Power PMAC settings, saved in a .cfg file, as explained in the Use Case 1 but instead the settings are in the .pmh and .cfg files.

The .cfg file is created using Create Config file option from Configuration node.

1. Open IDE 4.1.x and select Open project.
2. Choose the project that is created using V3.x IDE.
3. On opening the project, a message will be displayed that the project is a 'One-way upgrade' process. A success message will display that the project has been upgraded successfully.

It is recommended that with this style of project the User should make sure the 'Download Systemsetup.cfg file' property is set to ***No***. Note: **-** This property is set to ***No*** if IDE V4.1.x is used.

In case of IDE V4.0.x, it is recommended that after Build and Download the Power PMAC settings are download again. It is also recommended to create a configuration file as detailed in Use Case 1.

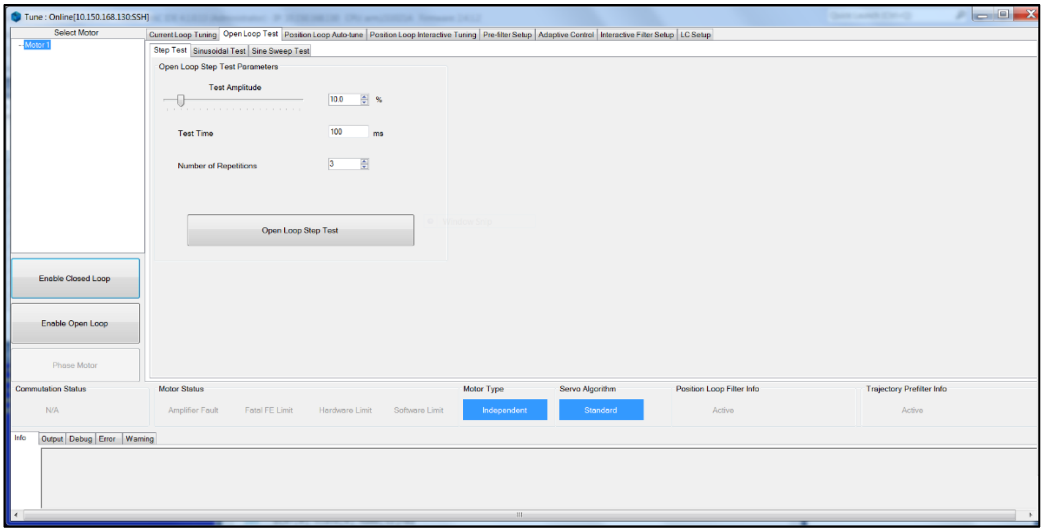
How to Tune 1S and G5 drive using Advance Tune tool

This section describes Tuning 1S and G5 using the Advance Tune tool option from the IDE when used in Cyclic Synchronous Torque mode (CST) or Cyclic Synchronous Velocity mode (CSV) mode.

A prerequisite is that the EtherCAT network is configured in either CST or CSV mode and Motor is setup correctly for EtherCAT.

Due to the transport delays over the network, the servo update frequency for the drive has to be set up to at least 2 kHz, preferably 4 KHz. The User can set this as described in Step 1 of EtherCAT setup.

It is good practice to check the open loop response to verify whether the drive is properly set before starting the Auto Tune move.



Enable open loop first, this step is necessary if the drive is not activated and perform an open loop test.

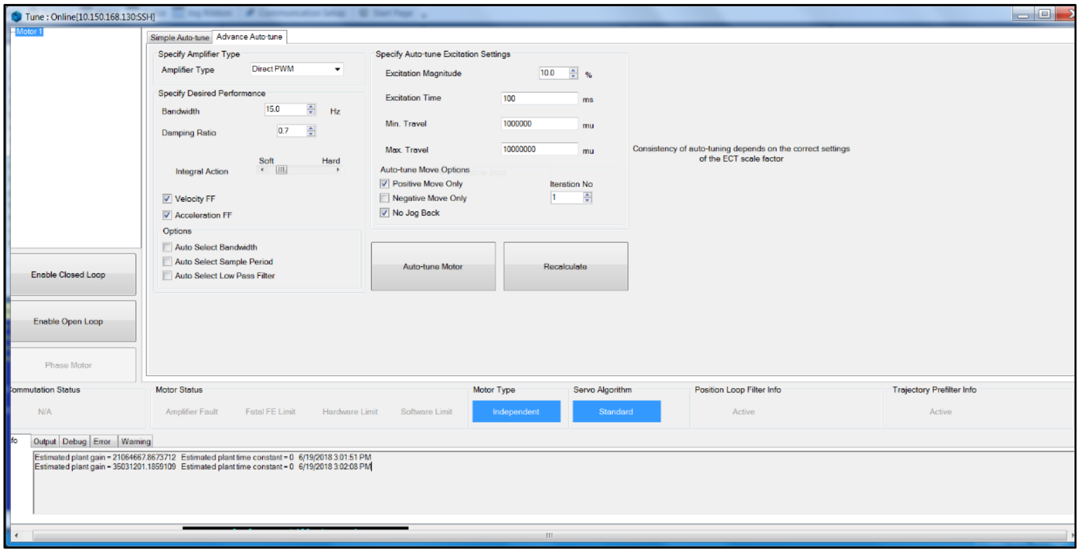
The response should look like the graph below. i.e. a linear relationship between the torque command and motor acceleration

If the open loop response does not show this linear relationship check the Maximum profile velocity, positive and negative torque limits for the drive.

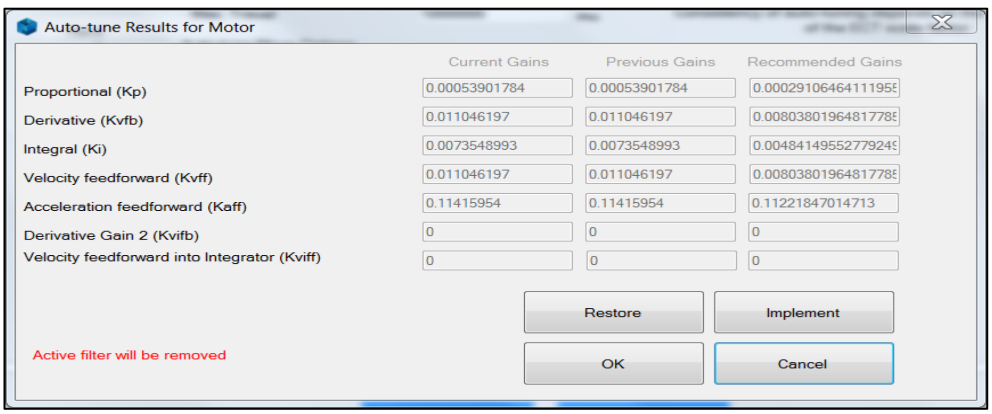
Go to Advance auto-tune tab under Position Loop Auto-tune, set the excitation magnitude to 10% and excitation time to 100, similar to the open loop step test values. Set the maximum travel to 1 or 2 motor revolution in motor units and set the minimum travel 1/10 motor revolution in motor units. e.g. for a motor with 23-bit encoder Maximum travel is ^{23} = 8388608.

Check the positive or negative move option. A bandwidth between 5 to 25 Hz can be selected and varying damping ratios or integral action.

Before performing an auto-tune move, verify the drive is active. Issue a #1out0 command from the terminal or press enable open loop button.

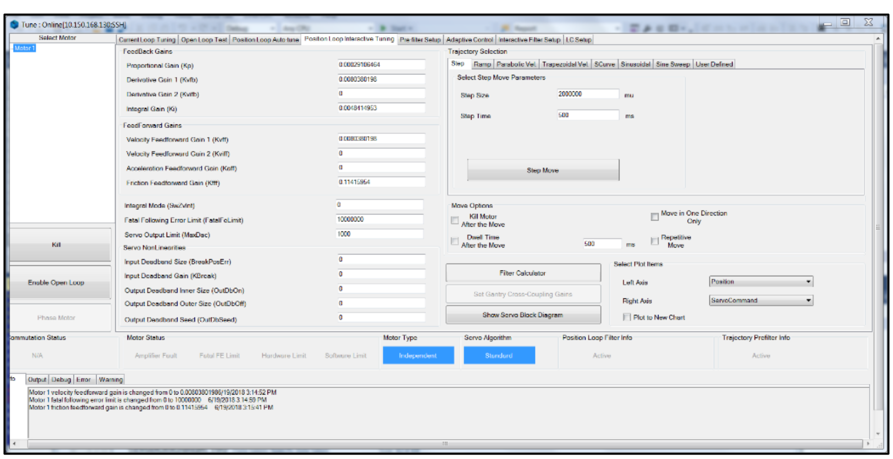


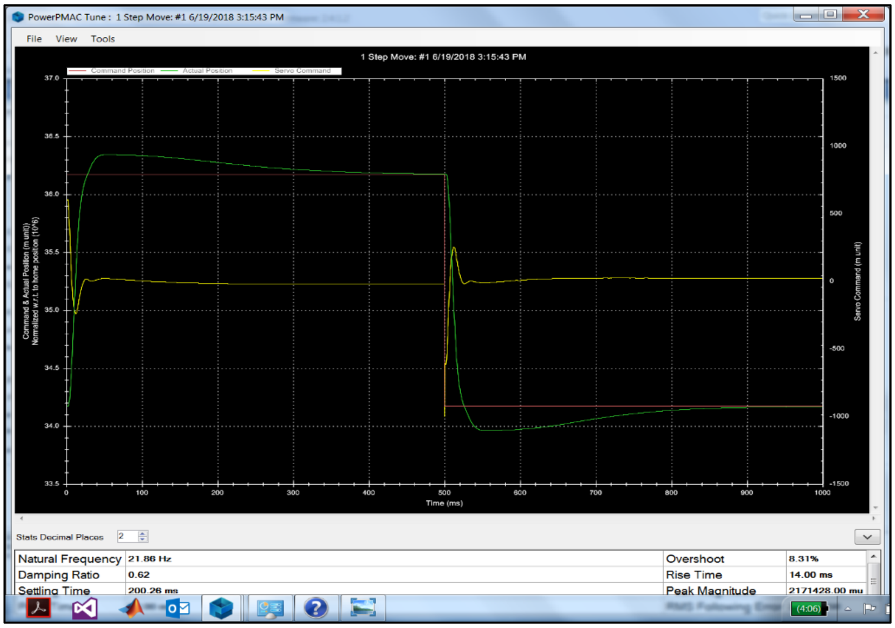
After the autotune move is completed, recommended gains are displayed as shown below.



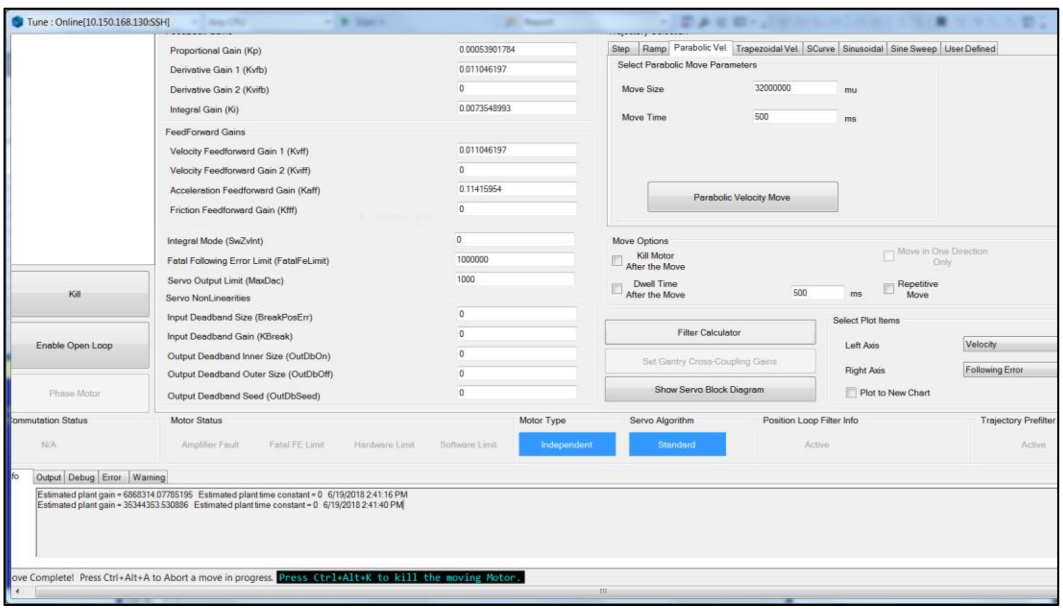
After implementing the servo gains, verify the tuning via a step move or a parabolic move. Typical responses are shown below:

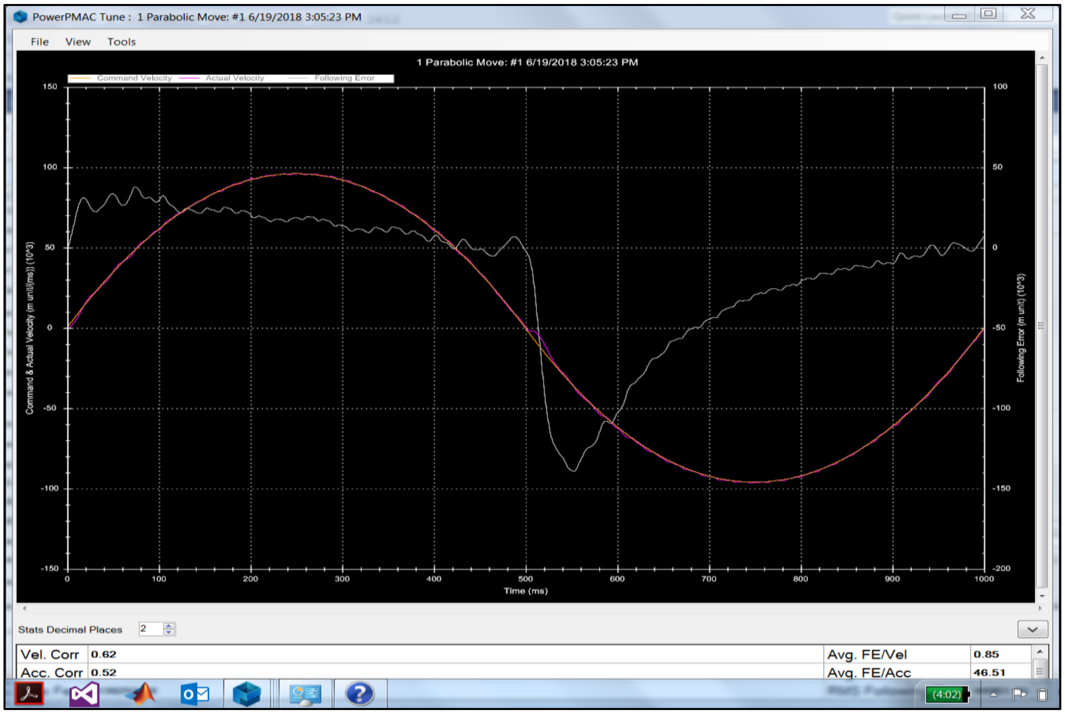
Step move





Parabolic Move




PowerPMAC Tune : 1 Parabolic Move: #1 6/19/2018 3:05:23 PM
File View Tools
1 Parabolic Move: #1 6/19/2018 3:05:23 PM
Command Velocity
Actual Velocity
Following Error
Command & Actual Velocity (m unit/(ms)) (10^3)
Following Error (m unit) (10^3)
Time (ms)
Stats Decimal Places 2
Vel. Corr 0.62
Acc. Corr 0.52
Avg. FE/Vel 0.85
Avg. FE/Acc 46.51

Motor-Encoder combination chart supported by System Setup

Power PMAC IDE future version will keep adding more motor-encoder combination as they are available.

Encoder vs. Motor	No Feedback	Quadrature	Analog Sinusoidal	Gate 3 (ACC-24E3) Serial*: Endat 2.2, SSI, Panasonic, Kawasaki, Mititoya, Tamagawa	(ACC-84) Serial: BiSS	(Ck3W-ECS) Serial* : BiSS, EnDat	(Ck3W-GC) Serial: XY2-100, SL2-100, TCR	Halls 60° and 120°
Virtual Motor	✓	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Galvonometer	N/A	N/A	N/A	N/A	N/A	N/A	✓	N/A
Stepper, PFM	✓	N/A	N/A	N/A	N/A	N/A	N/A	N/A
ECAT CSP	✓	N/A	N/A	N/A	N/A	N/A	N/A	N/A
ECAT CST	N/A	✓	✓	✓	✓	✓	N/A	✓
Rotary Brushed (Analog)	N/A	✓	✓	✓	✓	✓	N/A	N/A
Linear Brushless PWM	N/A	✓	✓	✓	✓	✓	N/A	✓
Rotary Brushless, PWM	N/A	✓	✓	✓	✓	✓	N/A	✓

* Absolute phasing working for all, Absolute position (homing) is working for all except Panasonic.

✓	Tested Working
X	Tested not working
N/A	Not applicable

ACONTIS Error Codes

(Ecat[n].Error ; n=Master Number)

4.2.1 Generic Error Codes **Code / Define**	**Text**	**Group**	**Possible error cause**
0x00000000 EC_E_NOERROR	No Error	n. a.	Function call successful.
0x98110001 EC_E_NOTSUPPORTED	Feature not supported	APP	Function or property not available
0x98110002 EC_E_INVALIDINDEX	Invalid Index	APP	CoE: invalid SDO index.
0x98110003 EC_E_INVALIDOFFSET	Invalid Offset	ISW	Invalid offset, while accessing Process Data Image
0x98110004 EC_E_CANCEL	Cancel	APP	master should abort current mbx transfer
0x98110005 EC_E_INVALIDSIZE	Invalid Size	APP	Invalid size - while accessing Process Data Image - while storing data
0x98110006 EC_E_INVALIDDATA	Invalid Data	ISW	Multiple error sources
0x98110007 EC_E_NOTREADY	Not ready	ISW	Multiple error sources
0x98110008 EC_E_BUSY	Busy	APP	Stack is busy currently and not available to process the API request. The function may be called again later.
0x98110009 EC_E_ACYC_FRM_FREEQ_EMPTY	Cannot queue acyclic ecat command	ISW	Acyclic command queue is full. Possible solution: Increase of configuration value *dwMaxQueuedEthFrames*
0x9811000A EC_E_NOMEMORY	No Memory left	CFG	Not enough allocatable memory available (memory full / corrupted).
0x9811000B EC_E_INVALIDPARM	Invalid Parameter	APP	API function called with erroneous parameter set.
0x9811000C EC_E_NOTFOUND	Not Found	APP	Network Information File not found or API called with invalid SlaveID.
0x9811000D EC_E_DUPLICATE	Duplicated fixed address detected	ISW	Internally handled.
0x9811000E EC_E_INVALIDSTATE	Invalid State	ISW	Multiple error sources

Code / Define	Text	Group	Possible error cause
0x9811000F EC_E_TIMER_LIST_FULL	Cannot add slave to timer list	ISW	Slave timer list full.
0x98110010 EC_E_TIMEOUT	Timeout		Multiple error sources.
0x98110011 EC_E_OPENFAILED	Open Failed	ISW	Multiple error sources.
0x98110012 EC_E_SENDFAILED	Send Failed	LLA	Transmit of frame failed.
0x98110013 EC_E_INSERTMAILBOX	Insert Mailbox error	CFG	Mailbox command couldn't be stored to internal command queue. Possible solution: Increase of configuration value *dwMaxQueuedCoeCmds*
0x98110014 EC_E_INVALIDCMD	Invalid Command	ISW	Unknown mailbox command code.
0x98110015 EC_E_UNKNOWN_MBX_PROTOCOL	Unknown Mailbox Protocol Command	ISW	Unknown Mailbox protocol or mailbox command with unknown protocol association.
0x98110016 EC_E_ACCESSDENIED	Access Denied	ISW	Master internal software error:
0x9811001A EC_E_PRODKEY_INVALID	Product Key Invalid	CFG	Application is using evaluation version of stack, which stops operation after 30 minutes.
0x9811001B EC_E_WRONG_FORMAT	Wrong configuration format	ENI	Network information file is empty or malformed.
0x9811001C EC_E_FEATURE_DISABLED	Feature disabled	APP	Application tried to perform a missing or disabled API function.
0x9811001E EC_E_BUSCONFIG_MISMATCH	Bus Config Mismatch	ENI	Network information file and currently connected bus topology does not match.
0x9811001F EC_E_CONFIGDATAREAD	Error reading config file	ENI	Network information file could not be read.
0x98110021 EC_E_XML_CYCCMDS_MISSING	Cyclic commands are missing	ENI	Network information file does not contain cyclic commands.
0x98110022 EC_E_XML_ALSTATUS_READ_MISSING	AL_STATUS register read missing in XML file for at least one state	ENI	Read of AL Status register is missing in cyclic part of given network information file.
0x98110023 EC_E_MCSM_FATAL_ERROR	Fatal internal McSm	ISW	Master control state machine is in an undefined state.
0x98110024 EC_E_SLAVE_ERROR	Slave error	SLV	Cannot address slave (no station address configured or slave absent)

Code / Define	Text	Group	Possible error cause
0x98110025 EC_E_FRAME_LOST	Frame lost, IDX mismatch	SLV	An EtherCAT frame was lost on bus segment, means the response was not received. In case this error shows frequently a problem with the wiring could be the cause.
0x98110026 EC_E_CMD_MISSING	At least one EtherCAT command is missing in the received frame	SLV	Received EtherCAT frame incomplete.
0x98110028 EC_E_INVALID_DCL_MODE	IOCTL EC_IOCTL_DC_LATCH_REQ_LTIMVALS invalid in DCL auto read mode	APP	This function cannot be used if DC Latching is running in mode „Auto Read".
0x98110029 EC_E_AI_ADDRESS	Auto increment address increment mismatch	SLV	Network information file and bus topology doesn't match any more. Error shows only, if a already recognized slave isn't present any more.
0x9811002A EC_E_INVALID_SLAVE_STATE	Slave in invalid state, e.g. not in OP (API not callable in this state)	APP	Mailbox commands are not allowed in current slave state.
0x9811002B EC_E_SLAVE_NOT_ADDRESSABLE	Station address lost (or slave missing) - FPRD to AL_STATUS failed	SLV	Slave had a power cycle.
0x9811002C EC_E_CYC_CMDS_OVERFLOW	Too many cyclic commands in XML configuration file	ENI	Error while creating network information file within configuration utility.
0x9811002D EC_E_LINK_DISCONNECTED	Ethernet link cable disconnected	SLV	EtherCAT bus segment not connected to network interface.
0x9811002E EC_E_MASTERCORE_INACCESSIBLE	Master core not accessible	RAS	Connection to remote server was terminated or master instance has been stopped on remote side.
0x9811002F EC_E_COE_MBXSND_WKC_ERROR	COE mbox send: working counter	SLV	CoE mailbox couldn't be read on slave, slave didn't read out mailbox since last write.
0x98110030 EC_E_COE_MBXRCV_WKC_ERROR	COE mbox receive: working counter	SLV	CoE Mailbox couldn't be read from slave.
0x98110031 EC_E_NO_MBX_SUPPORT	No mailbox support	APP	Slave does not support mailbox access.
0x98110032 EC_E_NO_COE_SUPPORT	CoE protocol not supported	ENI	Configuration error or slave information file doesn't match slave firmware.
0x98110033 EC_E_NO_EOE_SUPPORT	EoE protocol not supported	ENI	Configuration error or slave information file doesn't match slave firmware.
0x98110034 EC_E_NO_FOE_SUPPORT	FoE protocol not supported	ENI	Configuration error or slave information file doesn't match slave firmware.
0x98110035 EC_E_NO_SOE_SUPPORT	SoE protocol not supported	ENI	Configuration error or slave information file doesn't match slave firmware.

Code / Define	Text	Group	Possible error cause
0x98110036 EC_E_NO_VOE_SUPPORT	VoE protocol not supported	ENI	Configuration error or slave information file doesn't match slave firmware.
0x98110037 EC_E_EVAL_VIOLATION	Configuration violates Evaluation limits	ENI	Network information file contains configuration data for more slaves than allowed, while using evaluation version of stack.
0x98110038 EC_E_EVAL_EXPIRED	Evaluation Time limit reached	CFG	Time limit (30minutes) of evaluation version is reached, hence master stack is stopped.
0x98110070 EC_E_CFGFILENOTFOUND	Master configuration not found	CFG	The path to the master configuration file (XML) was wrong or the file is not available.
0x98110071 EC_E_EEPROMREADERRO R	Command error while EEPROM upload	SLV	Could not read from slave EEPROM.
0x98110072 EC_E_EEPROMWRITEERR OR	Command error while EEPROM download	SLV	Could not write to slave EEPROM.
0x98110073 EC_E_XML_CYCCMDS_SIZ EMISMATCH	Cyclic command wrong size (too long)	ENI	Error while creating a new cyclic command. The size which was defined in the master configuration xml does not match to the size of the process data.
0x98110075 EC_E_XML_INVALID_OUT_ OFF	Invalid output offset in cyc cmd, please check OutputOffs	ENI	*Obsolete*
0x9811010e EC_E_SLAVE_NOT_PRESE NT	Command not executed (slave not present on bus)	APP / SLV	Slave disappeared or was never present.
0x98110112 EC_E_SYSDRIVERMISSING	Cannot open system driver	SYS	System driver was not loaded.
0x9811011E EC_E_BUSCONFIG_TOPOC HANGE	Bus configuration not detected, Topology changed		Topology changed while scanning bus
0x98110123 EC_E_VOE_MBX_WKC_ER ROR	VoE mailbox send: working counter	SLV	VoE mailbox couldn't be written.
0x98110124 EC_E_EEPROMASSIGNERR OR	EEPROM assignment failed	SLV	Assignment of the EEPROM to the slave went wrong.
0x98110125 EC_E_MBX_ERROR_TYPE	Error mailbox received	SLV	Unknown mailbox error code received in mailbox
0x98110126 EC_E_REDLINEBREAK	Redundancy line break	SLV	Cable break between slaves or between master and first slave
0x98110127 EC_E_XML_INVALID_CMD_ WITH_RED	Invalid EtherCAT cmd in cyclic frame with redundancy	ENI	BRW commands are not allowed with redundancy. LRW commands with an expected WKC>3 are not allowed with redundancy (Workaround: Use LRD/LWR instead of

Code / Define	Text	Group	Possible error cause
LRW)			
0x98110128 EC_E_XML_PREV_PORT_MISSING	<PreviousPort>-tag is missing	ENI	If the auto increment address is not the first slave on the bus we check if a previous port tag OR a hot connect tag is available
0x98110129 EC_E_XML_DC_CYCCMDS_MISSING	DC is enabled and DC cyclic commands are missing (e.g. access to 0x900)	ENI	Error in Configuration Tool.
0x98110130 EC_E_DLSTATUS_IRQ_TOPOCHANGED	Data link (DL) status interrupt because of changed topology	SLV	Handled inside the master
0x98110131 EC_E_PTS_IS_NOT_RUNNING	Pass Through Server is not running	PTS	The Pass-Through-Server was tried to be enabled/disabled or stopped without being started.
0x98110132 EC_E_PTS_IS_RUNNING	Pass Through Server is running	PTS	*Obsolete*. Replaced by EC_E_ADS_IS_RUNNING
0x98110132 EC_E_ADS_IS_RUNNING	ADS adapter (Pass Through Server) is running	PTS	API call conflicts with ADS state (running).
0x98110133 EC_E_PTS_THREAD_CREATE_FAILED	Could not start the Pass Through Server	PTS	The Pass-Through-Server could not be started.
0x98110134 EC_E_PTS_SOCK_BIND_FAILED	The Pass Through Server could not bind the IP address with a socket	PTS	Possibly because the IPaddress (and Port) is already in use or the IP-address does not exist.
0x98110135 EC_E_PTS_NOT_ENABLED	The Pass Through Server is running but not enabled	PTS	-
0x98110136 EC_E_PTS_LL_MODE_NOT_SUPPORTED	The Link Layer mode is not supported by the Pass Through Server	PTS	The Master is running in interrupt mode but the Pass-Through-Server only supports polling mode.
0x98110137 EC_E_VOE_NO_MBX_RECEIVED	No VoE mailbox received	SLV	The master has not yet received a VoE mailbox from a specific slave.
0x98110138 EC_E_DC_REF_CLOCK_SYNC_OUT_UNIT_DISABLED	SYNC out unit of reference clock is disabled	ENI	Slave is selected as Reference clock in ENI file, but slave doesn't have a SYNC unit. Possible a ESI file bug.
0x98110139 EC_E_DC_REF_CLOCK_NOT_FOUND	Reference clock not found!	SLV	May happen if reference clock is removed from network.
0x9811013B EC_E_MBX_CMD_WKC_ERROR	Mailbox command working counter error	SLV	Mbx Init Cmd Retry Count exceeded.
0x9811013C EC_E_NO_AOE_SUPPORT	AoE: Protocol not supported	APP / SLV	Application calls AoE-API although not implemented at slave.

Code / Define	Text	Group	Possible error cause
0x9811016E EC_E_XML_AOE_NETID_INVALID	AoE: Invalid NetID	ENI	Error from Configuration Tool.
0x9811016F EC_E_MAX_BUS_SLAVES_EXCEEDED	Error: Maximum number of bus slave has been exceeded	CFG	The maximum number of pre-allocated bus slave objects are to small. The maximum number can be adjusted by the master initialization parameter EC_T_INITMASTERPARMS.wMaxBusSlaves.
0x98110170 EC_E_MBXERR_SYNTAX	Mailbox error: Syntax of 6 octet Mailbox header is wrong	SLV	Slave error mailbox return value: 0x01
0x98110171 EC_E_MBXERR_UNSUPPORTEDPROTOCOL	Mailbox error: The Mailbox protocol is not supported	SLV	Slave error mailbox return value: 0x02
0x98110172 EC_E_MBXERR_INVALIDCHANNEL	Mailbox error: Field contains wrong value	SLV	Slave error mailbox return value: 0x03
0x98110173 EC_E_MBXERR_SERVICENOTSUPPORTED	Mailbox error: The mailbox protocol header of the mailbox protocol is wrong	SLV	Slave error mailbox return value: 0x04
0x98110174 EC_E_MBXERR_INVALIDHEADER	Mailbox error: The mailbox protocol header of the mailbox protocol is wrong	SLV	Slave error mailbox return value: 0x05
0x98110175 EC_E_MBXERR_SIZETOOSHORT	Mailbox error: Length of received mailbox data is too short	SLV	Slave error mailbox return value: 0x06
0x98110176 EC_E_MBXERR_NOMOREMEMORY	Mailbox error: Mailbox protocol can not be processed because of limited resources	SLV	Slave error mailbox return value: 0x07
0x98110177 EC_E_MBXERR_INVALIDSIZE	Mailbox error: The length of data is inconsistent	SLV	Slave error mailbox return value: 0x08
0x98110178 EC_E_DC_SLAVES_BEFORE_REF_CLOCK	Slaves with DC configured present on bus before reference clock	ENI	The first DC Slave was not configured as potential reference clock.
0x9811017B EC_E_LINE_CROSSED	Line crossed	Cabling wrong.	
0x9811017C EC_E_LINE_CROSSED_SLAVE_INFO	Line crossed at slave …	*Obsolete*	