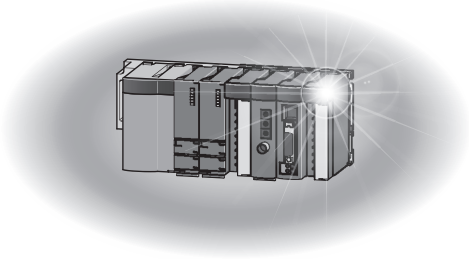
MITSUBISHI ELECTRIC

Mitsubishi Programmable Controller

MELSEC Q series

AD51H-BASIC Programming Manual (Debug and Compile)

-QD51
-QD51-R24
-A1SD51S
-AD51H-S3



• SAFETY PRECAUTIONS •

(Always read these instructions before using this equipment.)

Before using this product, please read this manual and the relevant manuals introduced in this manual carefully and pay full attention to safety to handle the product correctly.
The instructions given in this manual are concerned with this product. For the safety instructions of the programmable controller system, please read the CPU module user's manual.
In this manual, the safety instructions are ranked as "DANGER" and "CAUTION".

⚠ DANGER	Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury.
⚠ CAUTION	Indicates that incorrect handling may cause hazardous conditions, resulting in medium or slight personal injury or physical damage.

Note that the ⚠CAUTION level may lead to a serious consequence according to the circumstances. Always follow the instructions of both levels because they are important to personal safety.

Please save this manual to make it accessible when required and always forward it to the end user.

[Design Precautions]

⚠ DANGER

- Make sure to configure the interlock line outside the PLC system so that the system always operates normally when changing the data and control status of the PLC being operated from a peripheral device.
  Moreover, determine in advance how the system handles with communication errors by poor cable connection, etc. that may occur when performing online operations on the PLC CPU from a peripheral device.

⚠ CAUTION

- Please read this manual thoroughly and confirm the safety before starting online operations (especially forced outputs and operating status modifications) performed by connecting a peripheral device to the operating CPU module.
  Incorrect online operations may cause damage to the machinery or result in accidents.

A - 1

REVISIONS

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Apr., 2000	SH(NA)-080091-A	First printing
Sep., 2000	SH(NA)-080091-B	Correction Section 7.3
Oct., 2003	SH(NA)-080091-C	Addition WARRANTY Correction Chapter 1, Section 7.3
Oct., 2004	SH(NA)-080091-D	Correction Chapter 1, Section 7.7
Dec., 2006	SH(NA)-080091-E	Addition Appendix 3 Correction About Manuals, Section 2.1.1, 2.1.2, 2.2, 4.1, 4.2.1 to 4.2.3, 4.3.2, 4.4, 4.4.1, 4.4.2, 4.5.1, 4.5.2, 4.7, 4.8, 5.1, 5.2.1, 5.2.3 to 5.2.6, 5.3, 5.3.5, 5.3.6, 5.4.2, 5.4.3, 5.5.1, 5.5.2, 5.6, 5.7, 6.5.1, 6.5.2, 7.2, 7.3, 7.5.1, 7.7, 7.8.1, 7.8.2, Appendix 2 Deletion Section 7.5.2

Japanese Manual Version SH-080093-E

INTRODUCTION

Thank you for purchasing the MELSEC-Q/A series PLC.
Before using the equipment, please read this manual carefully to develop full familiarity with the functions and performance of the Q, A series PLC you have purchased, so as to ensure correct use.
Please forward a copy of this manual to the end user.

CONTENTS

About Manuals

The following manuals are also related to this product.
If necessary, order them by quoting the details in the tables below.

Related Manuals

Manual Name	Manual Number (Model Code)
Type AD51H-S3 Intelligent Communication Module User's Manual This manual contains information on the system configuration when using the module, module specifications, name and setting for each part, description of each function, and external dimensions of the module. (Provided with the module)	IB-66401 (13JE16)
Type A1SD51S Intelligent Communication Module User's Manual This manual contains information on the system configuration when using the module, module specifications, name and setting for each part, description of each function, and external dimensions of the module. (Sold separately)	IB-66551 (13JE90)
Type QD51/QD51-R24 Q Corresponding Intelligent Communication Module User's Manual This manual contains information on the system configuration when using the module, module specifications, name and setting for each part, description of each function, and external dimensions of the module. (Sold separately)	SH-080089 (13JR16)
AD51H-BASIC Programming Manual (Command) (Corresponds to the QD51, QD51-R24, A1SD51S, AD51H-S3.) Explains the commands, how to use the functions and the specifications of AD51H-BASIC. (Sold separately)	SH-080090 (13JF63)
AD51H-BASIC Package Type SW1IVD-AD51HP-E (For QD51(-R24), A1SD51S, AD51H-S3) Operating Manual This manual contains information on how to operate the software packages for IBM PCs/AT compatible PCs. (Provided with the software package)	IB-66698 (13J910)

A - 6

1 OVERVIEW

This programming manual explains system and debug commands as well as compilation methods used with the communication module.
When applying the following program examples to the actual system, make sure to examine the applicability and confirm that it will not cause system control problems.

(1) System and debug commands

The following operations can be performed by entering commands from the console or debugger:

- Edit and debug a BASIC program.
- Load and save a BASIC program from/to a memory card, floppy disk, or hard disk.
- Execute, stop, and display the status of a BASIC program.
- Read and write from/to general-purpose input/output and internal devices.
- Change and read multitask settings.

(2) Creation of BASIC programs using a general-purpose editor

It is possible to create BASIC programs in online, using any general-purpose editor that is available in the market.
Line numbers can furthermore be added to a program created with a general-purpose editor by using a line numbering tool.

(3) Compiling BASIC programs

It is possible to use a compiler to compile BASIC programs created by interpreter BASIC.
The execution speed of compiler BASIC is 3 to 4 times faster as compared with interpreter BASIC.

1

MEMO

1 - 2

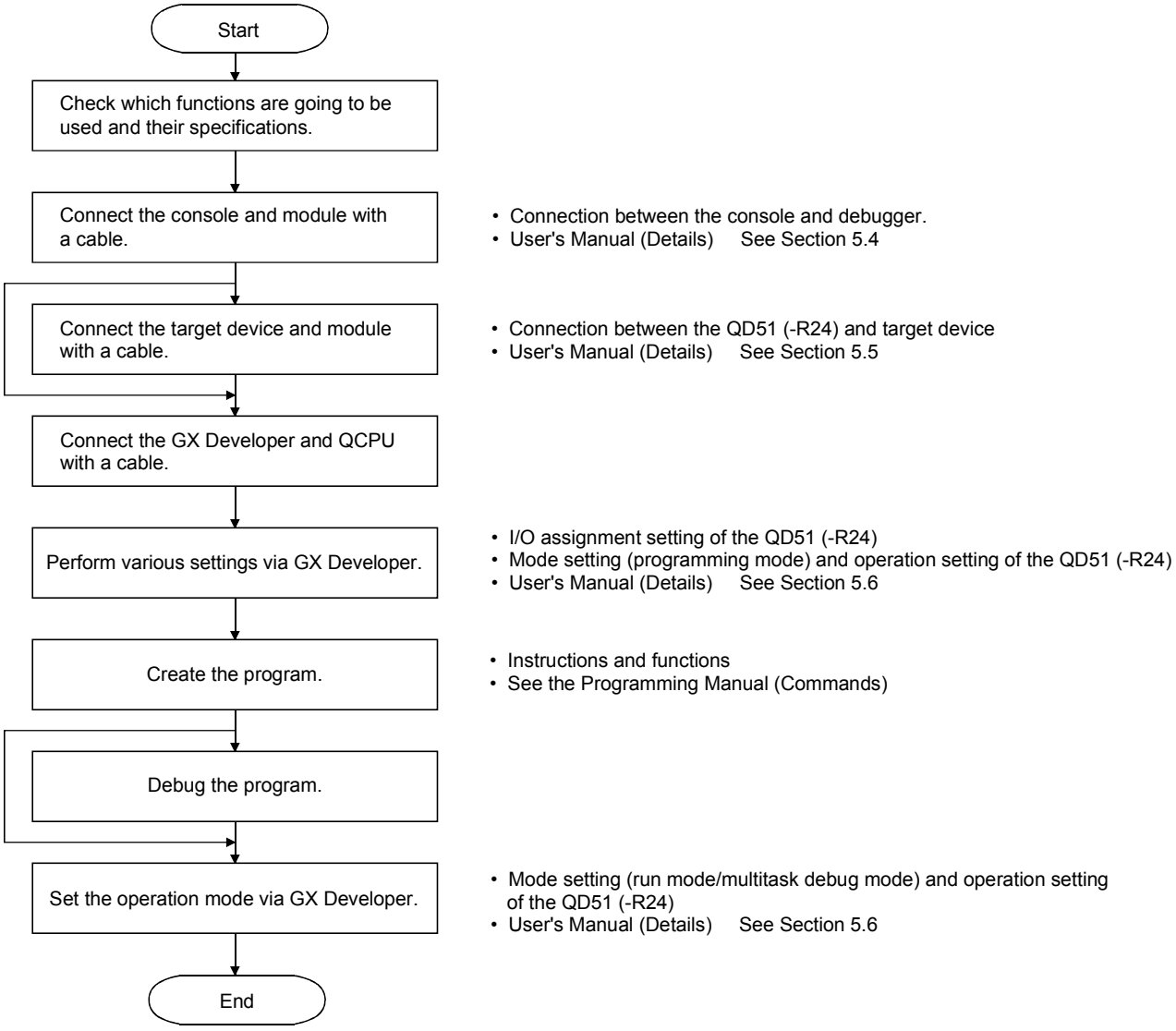
2 COMMUNICATION MODULE STARTUP AND MODE CHANGE

This chapter explains how to start up the communication module and how to change modes after the startup, when performing the online programming operations described in Chapter 4 and multitask debugging operations described in Chapter 5.

2.1 Outline of the Startup Procedure

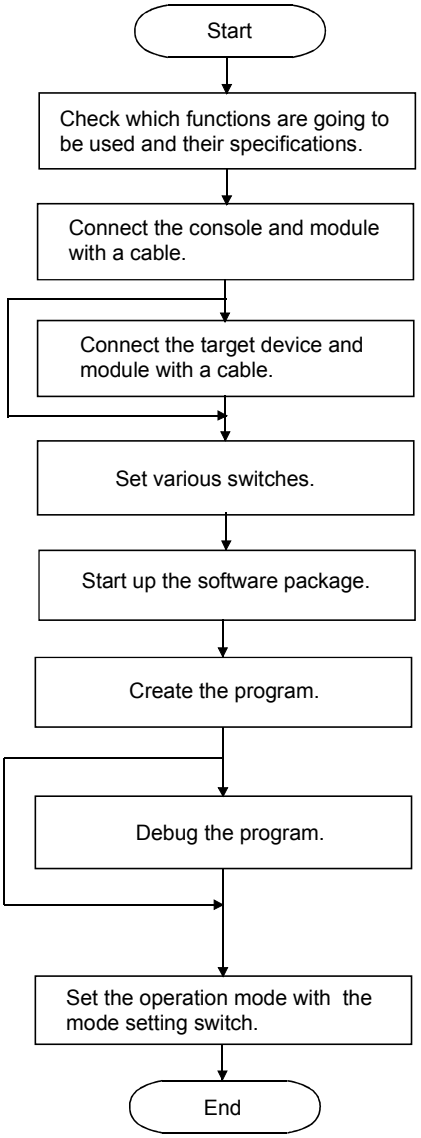
2.1.1 Starting up the QD51 (-R24)

The following flow chart shows an outline of the QD51 (-R24) startup procedure.



2.1.2 Starting up the A1SD51S/AD51H-S3

The following flow chart shows the outline of the A1SD51S/AD51H-S3 startup procedure.



- Connection between the console and debugger
- User's Manual

- Connection between the A1SD51S or AD51H-S3 and target device
- User's Manual

- Setting of the mode setting switch
- Setting of the console and debugger
- User's Manual

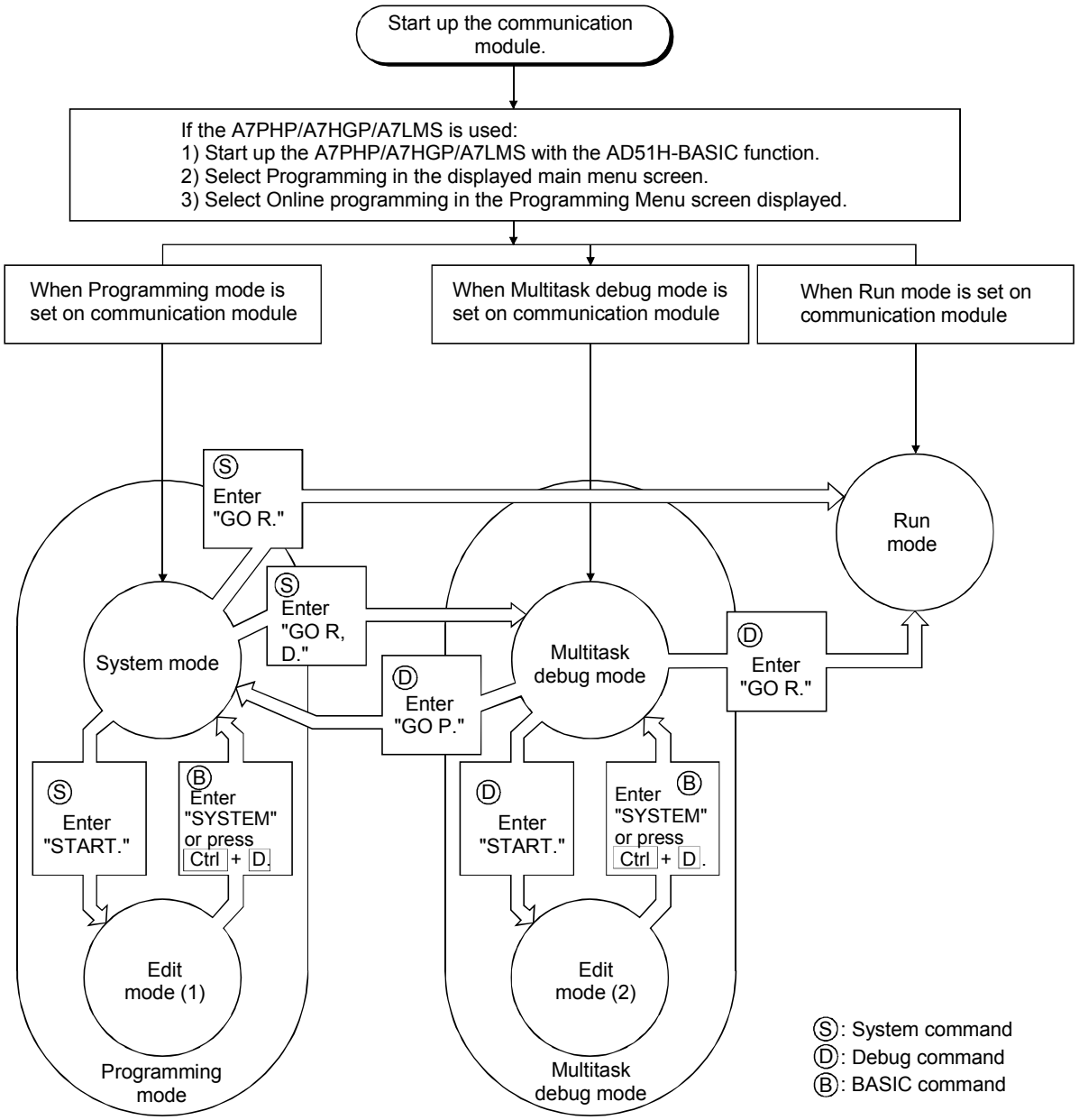
- Instructions and functions
- See the Programming Manual (Commands)

- Setting of the mode setting switch
- User's Manual

2 - 2

2.2 About Changing Between the Modes of the Communication Module

After startup of the communication module, the mode can be changed between multiple modes by entering a system command from the console or entering a debug command from the debugger.
This chapter explains how to change between the modes of the communication module by entering system commands and debug commands, and provides a brief description of each mode.



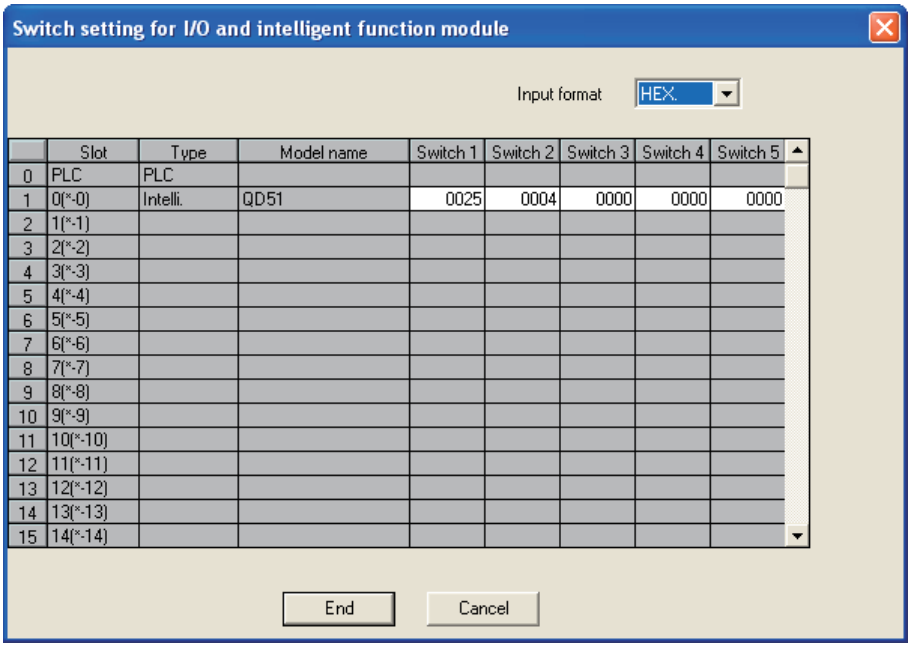
(1) Setting the mode activated at startup

How to set the mode activated at startup differs between module models.

(a) QD51 (-R24)

Make the setting in the Intelligent function module switch setting (Switch 2) of GX Developer.

[Setting screen]



Switch No.	Set value	Operation mode
Switch 2	0000_H	Run mode
	0001_H	
	0002_H	Multitask debug mode
	0003_H	
	0004_H	Programming mode
	0005_H to $FFFF_H$	Not used

For the switch setting of the QD51 (-R24), refer to the User's Manual.

(b) Setting of A1SD51S and AD51H-S3

Use the mode setting switch 1 of the communication module

Mode setting switch	Setting No.	Operation mode
	0	Run mode
	1	
	2	Multitask debug mode
	3	
	4	Programming mode
	5 to F	Not used

For the switch setting of the A1SD51S and AD51H-S3, refer to the User's Manual.

(2) Respective modes

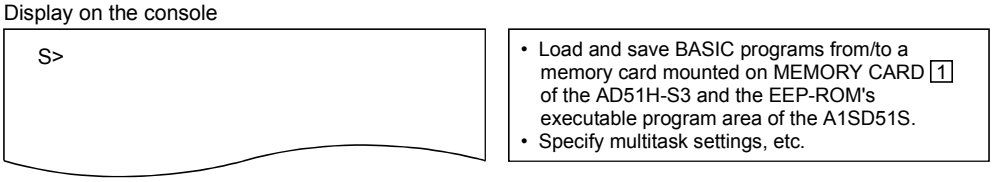
(a) Programming mode

1) The user can edit, debug, load/save from/to a memory card, and specify multitask settings for each BASIC program.
2) There are two modes in programming mode for performing the operations above: system mode and edit mode (1).

(b) System mode

1) This mode is activated when the communication module is started with Programming mode setting, or when the G0 command (G0 P) is entered with the debugger in Multitask debug mode.
2) The console is controlled by the operating system (OS) of the communication module.
3) Entering a system command from the console allows the following operations for each BASIC program.

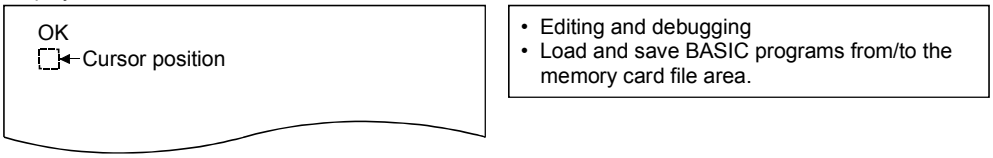
Display on the console



(c) Edit mode (1)

1) This is the mode that is changed to when the START command is entered on the console in system mode.
2) The console input is used by the interpreter (an OS that analyzes and executes BASIC commands).
3) It is possible to perform the following operations for each BASIC program, by entering instructions/functions of AD51H-BASIC from the console.

Display on the console



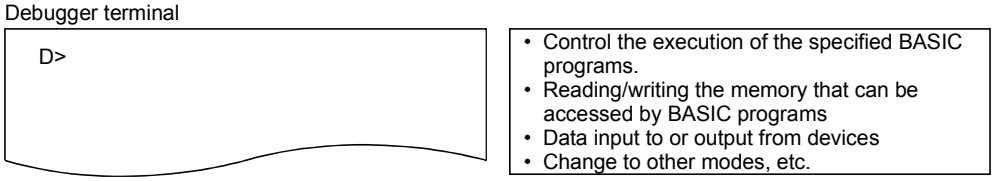
(d) Run mode

1) This mode is activated when the communication module is started up with Run mode setting, or when the GO command is entered from the console or debugger.
   On the A1SD51S/AD511H-S3, tasks are started with the RUN switch or RUN key switch set to “RUN”.
2) The multitask setting allows original system controls using multiple BASIC programs.

(e) Multitask debug mode

1) This mode is activated when the communication module is started up with Multitask debug mode setting, or when the GO command is entered from the console.
   On the A1SD51S/AD51H-S3, tasks are started with the RUN switch or RUN key switch set to “RUN”.
2) In Multitask debug mode, debugging is performed using the debuger (OS that analyzes and executes debug commands) of the communication module.
3) Entering a debug command from the debugger allows debugging of each BASIC program during multitask execution.

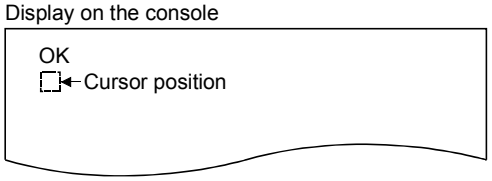
Debugger terminal



(f) Edit mode (2)

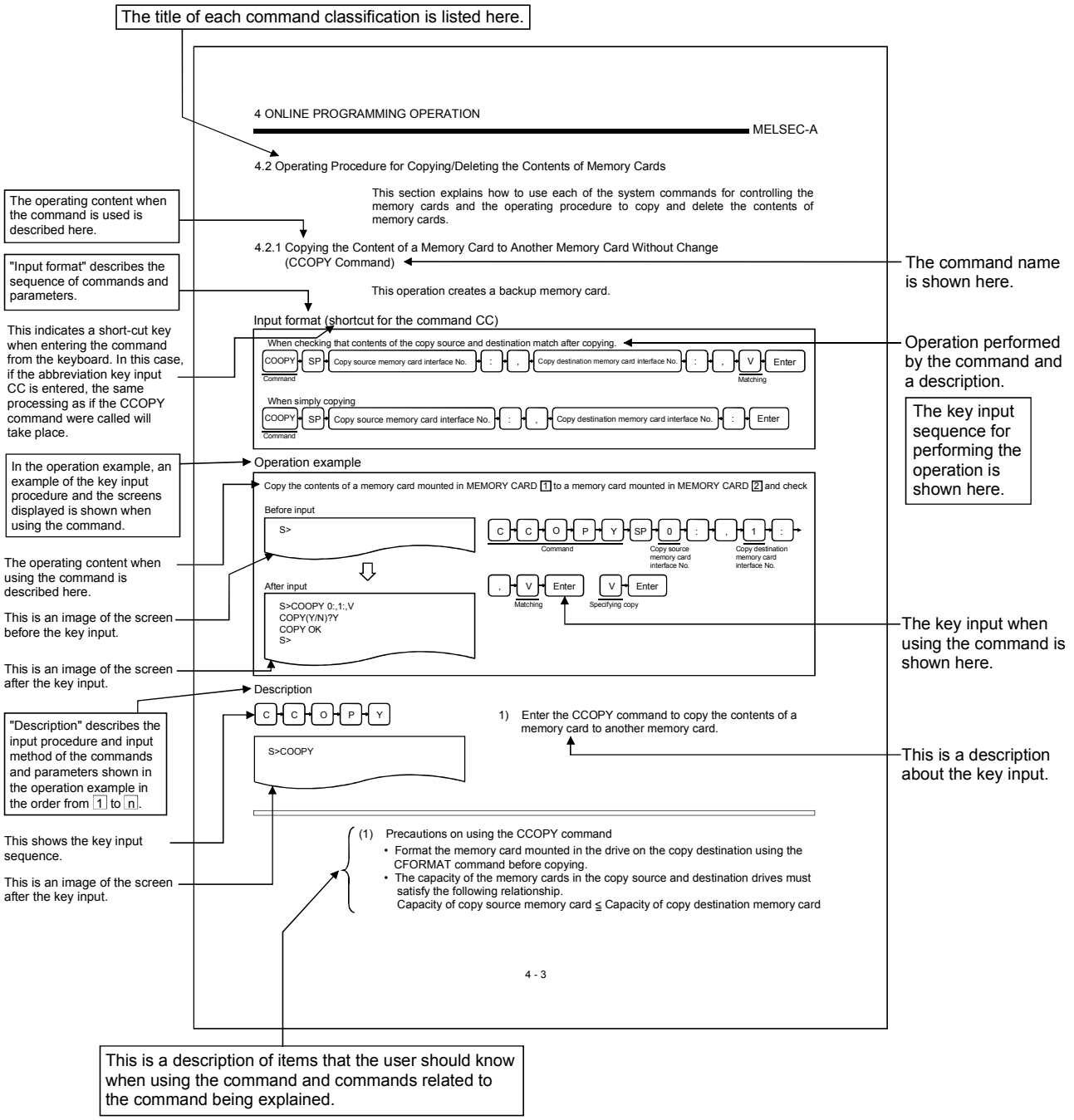
1) This is the mode that is changed to when the START command is entered from the debugger in multitask debug mode.
   (Tasks other than the task specified by the START command continue their multitask processing.)
2) The terminal for debugging is controlled by the interpreter.
3) It is possible to modify any BASIC program while executing other BASIC programs by entering instructions/functions of AD51H-BASIC from the debugger.

Display on the console



3 COMMAND EXPLANATION FORMAT

The following format is used to explain each command.



MEMO

3 - 2

4 ONLINE PROGRAMMING OPERATION

Online programming refers to editing and debugging BASIC programs, as well as loading and saving BASIC programs from/to memory cards, user-made floppy disks, and hard disks using the console connected to the communication module.
(Only one BASIC program in one task can be debugged at a time in online programming.)

This chapter explains how to use system commands for editing and debugging BASIC programs, as well as loading and saving BASIC programs from/to memory cards, user-made floppy disks, and hard disks using the console in system mode.

(1) This chapter mainly explains the key inputs and displays on the console side.
It is therefore generally omitted to state this fact explicitly for most key inputs and displays.
When necessary, it is pointed out explicitly that key inputs and displays are on the debugger side.

(2) It is necessary to perform the following tasks in advance in order to perform the online programming described in this chapter.
Perform each operation beforehand according to the explanation in the reference chapters below.
- In order to establish communication for performing online programming, the user should:
  Set the switches of the module : See Chapter 2.
- Connect the console : See Chapter 2.

4 - 1

4.1 System Command List

Table 4.1 lists system commands entered on the console from the keyboard during online programming.

Table 4.1 System Command List

Classification	System command	Function overview	Availability for module			Reference section
			AD51H-S3	A1SD51S	QD51 (-R24)	
Memory card control	CCOPY	Copies the contents of a memory card to another memory card without change. (Creation of a memory card for backup)	○	×	×	Section 4.2.1
	CFORMAT	Formats (physical format) a memory card.				Section 4.2.2
	CFORMAT?	Displays formatting information of a memory card.				Section 4.2.3
	CRECOVER	Recovers a file area in the unusable status to the usable status.				Section 4.9
	FFORMAT	Formats (logical format) the file area of a memory card.				Section 4.10
Executable program information control	MLOAD ＊1	Loads the contents of the specified BASIC task area in a memory card/EEP-ROM to the target BASIC task area of the communication module.	○	○	○	Section 4.3.1
	MSAVE	Saves the contents of the specified BASIC task area of the communication module to the target BASIC task area of a memory card/EEP-ROM. (The multitask settings are automatically specified)				Section 4.3.2
Multitask setting control	SET	Changes the multitask settings.	○	○	○	Section 4.4.1
	SET?	Displays the specified data of the multitask settings.				Section 4.4.2
Mode control	START ＊1	Changes the mode of the communication module from system mode to edit mode (1). (For editing and debugging each program)	○	○	○	Section 4.5.1
	GO	Changes the mode of the communication module from system mode to run mode or multitask debug mode.	○	○	○	Section 4.5.2
Interpreter operation control	TKILL ＊1	Ends the operation of the interpreter in the specified BASIC task area of the communication module.	○	○	○	Section 4.6
Others	EXIT	Displays the main menu screen on the console.	○	○	○	Section 4.7
	HELP	Displays the system command list, function overview, and command input format.				Section 4.8

＊1 These commands cannot be executed on tasks in which compiled BASIC programs are stored.

4.2 Operating Procedure for Copying/Deleting the Contents of Memory Cards For AD51H-S3 Only

This section explains how to use each of the system commands for controlling the memory cards and the operating procedure to copy and delete the contents of memory cards.

4.2.1 Copying the Content of a Memory Card to Another Memory Card Without Change (CCOPY Command) For AD51H-S3 Only

This operation creates a backup memory card.

Input format (shortcut for the command CC)

When checking that the contents of the copy source and destination match after copying.

CCOPY (Command) → SP → Copy source memory card interface No. → : → , → Copy destination memory card interface No. → : → , → V (Matching) → Enter

When simply copying

CCOPY (Command) → SP → Copy source memory card interface No. → : → , → Copy destination memory card interface No. → : → Enter

Operation example

Copy the contents of the memory card mounted in MEMORY CARD 1 to the memory card mounted in MEMORY CARD 2 and check that they match.

Before input

```
S>
```

C → C → O → P → Y (Command) → SP → 0 (Copy source memory card interface No.) → : → , → 1 (Copy destination memory card interface No.) → :

, → V (Matching) → Enter

After input

```
S>CCOPY 0:,1:,V
COPY(Y/N)?Y
COPY OK
S>
```

→ Y (Specifying copy) → Enter

Description

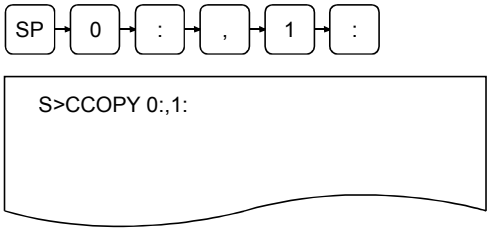
C → C → O → P → Y

```
S>CCOPY
```

1) Enter the CCOPY command to copy the content of a memory card to another memory card.

(1) Precautions on using the CCOPY command

- Format the memory card mounted in the drive on the copy destination using the CFORMAT command before copying.
- The capacity of the memory cards in the copy source and destination drives must satisfy the following relationship.
  Capacity of copy source memory card ≤ Capacity of copy destination memory card

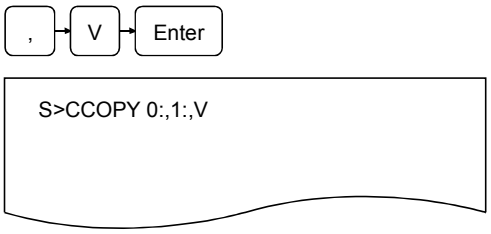


2) Enter the memory card interface number followed by a colon (:) for both the copy source and destination. Enter the copy source first, then the copy destination. Only 0 or 1 can be specified.

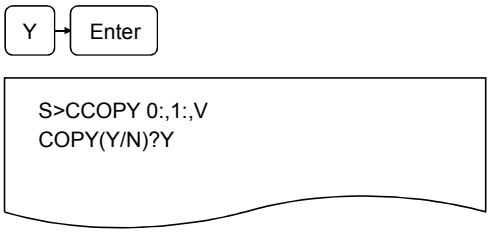
0 : The MEMORY CARD [1] drive on the AD51H-S3.

1 : The MEMORY CARD [2] drive on the AD51H-S3.

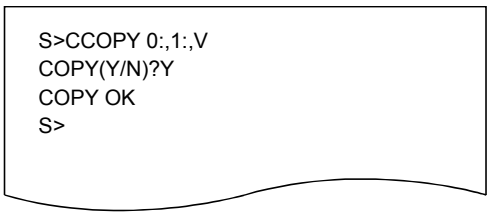
In the example figure to the left, the contents of a memory card in MEMORY CARD [1] are copied to a memory card in MEMORY CARD [2].



3) Specify "V" if it should be checked that the contents of the copy source and destination match after the copying.
Press [Enter] if it is not required to check that the contents match.
In the example figure to the left, the contents are checked after copying.



4) The screen displays "COPY (Y/N)? "
Enter [Y] to copy.
Enter [N] to stop copying. (The console returns to waiting for a system command entry.)
In the example figure to the left, copying is specified.



5) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "COPY OK" is displayed.
If the command ends abnormally, an error message or similar is displayed.
In the example figure to the left, a display where the command ends normally is shown.

6) "S>" is displayed in the line following the command execution result.
Enter the next command.

(2) Reference

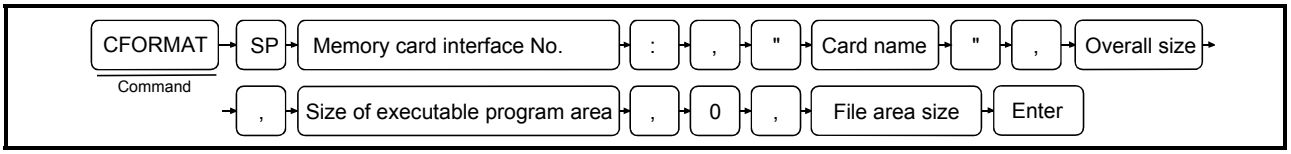
- Operation for formatting a memory card : CFORMAT command (Section 4.2.2)
- Operation for displaying formatting information of a memory card : CFORMAT? command (Section 4.2.3)

4 - 4

4.2.2 Formatting a Memory Card (CFORMAT Command) For AD51H-S3 Only

This operation formats a memory card (physically) mounted in MEMORY CARD 1 or 2 on the AD51H-S3.

Input format (shortcut for the command CF)



Operation example

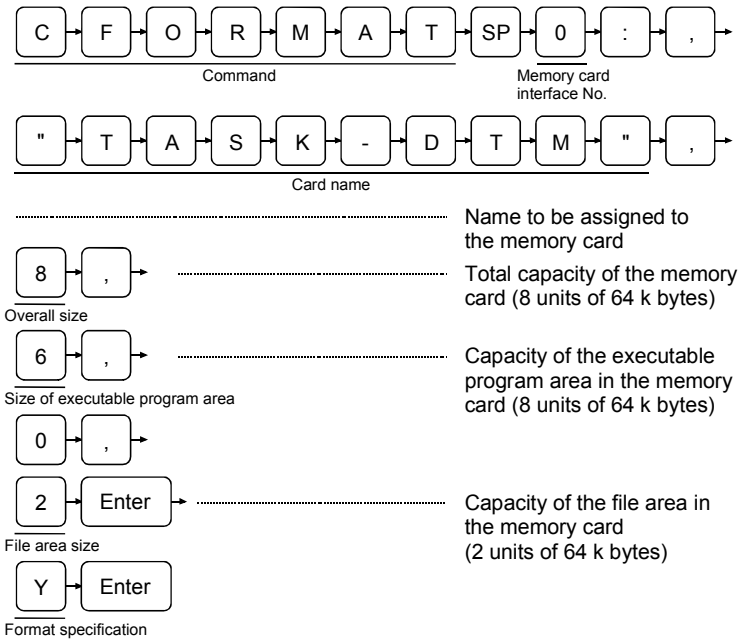
Format the memory card (with a capacity of 512 k bytes) mounted in MEMORY CARD 1 under the following conditions.

Before input

```
S>
```

After input

```
S>CFORMAT 0:,"TASK-DTM",8,6,0,2
FORMAT(Y/N)?Y
FORMAT OK
S>
```



Description

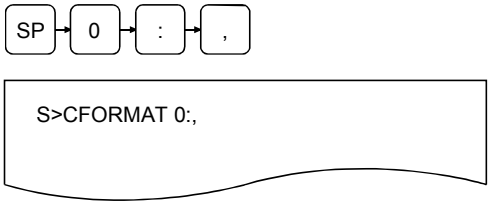
C F O R M A T

1) Enter the CFORMAT command to format a memory card.

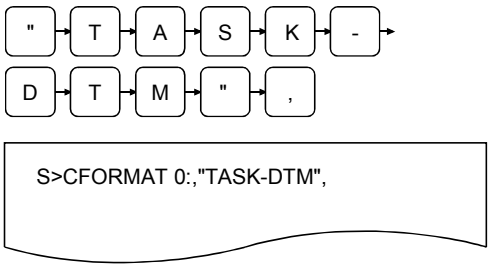
```
S>CFORMAT
```

(1) Precautions when using the CFORMAT command

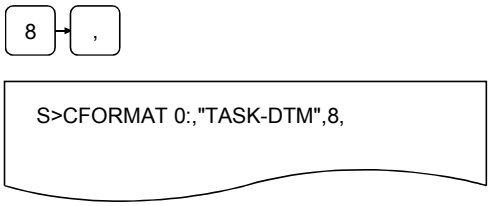
- If a memory card is formatted, all data that was written is deleted.
- When formatting a memory card that is write protected, the write protect should be canceled first.
- When formatting a memory card mounted in MEMORY CARD 1, the memory protection key switch of the AD51H-S3 module should be turned off first.



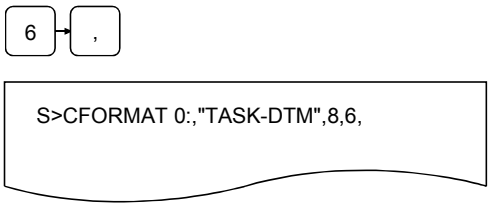
2) Enter the number of the memory card interface in which the memory card to be formatted is mounted followed by a colon (:). Only 0 or 1 can be specified.
   0 : The MEMORY CARD [1] drive on the AD51H-S3.
   1 : The MEMORY CARD [2] drive on the AD51H-S3.
   In the example figure to the left, the memory card mounted in MEMORY CARD [1] is specified.



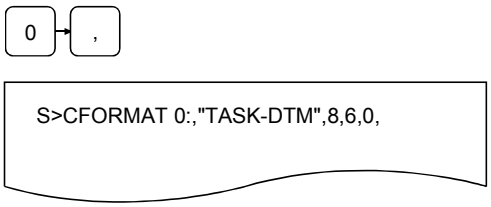
3) Enter a name of maximum 16 alphanumeric characters and symbols that will be assigned to the memory card after formatting.
   The first character must be an alphabetic character and the name area should be enclosed by double quotation marks (").
   In the example figure to the left, the memory card is named TASK-DTM.



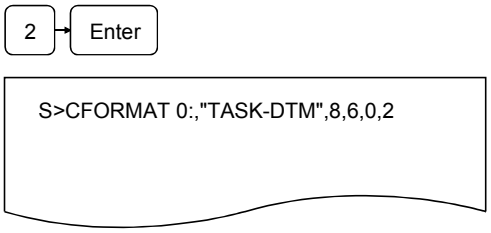
4) Enter the total capacity of the memory card to be formatted. This value must be 1 or greater (unit: 64 k bytes).
   The total capacity must be the total value of each of the sizes specified in the following formula.
   Overall size (total capacity) = (size of executable program area + file area size)
   In the example figure to the left, the memory card is formatted to contain 512 k bytes.
   (8 × 64 k bytes → 512 k bytes)



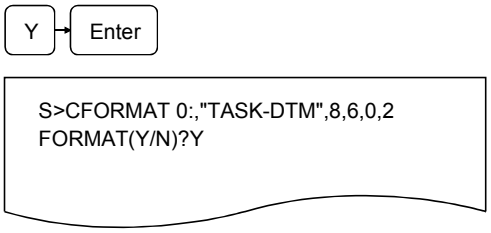
5) Enter the capacity reserved for the executable program area in the memory card after the formatting. This value must be from 0 to 6 (unit: 64 k bytes).
   The maximum capacity of the executable program area is 384 k bytes. It is used for the OS area (128 k bytes) and all of the BASIC task areas (where executable programs are stored).
   In the example figure to the left, 384 k bytes are reserved for the executable program area.



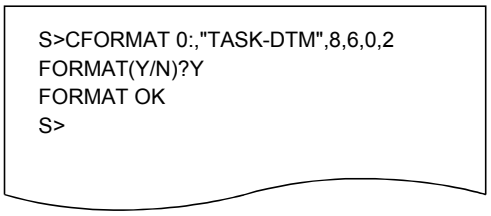
6) Enter 0 as a placeholder.



7) Enter the capacity reserved for the file area in the memory card after the formatting. This value must be 0 or greater (unit: 64 k bytes).
This area is used to store BASIC programs and data files that are not stored in the BASIC task areas.
In the example figure to the left, 128 k bytes are reserved for the file area. (2 × 64 k bytes → 128 k bytes)



8) The screen displays "FORMAT (Y/N)? "
Enter Y to format.
Enter N to stop formatting. (The console returns to waiting for a system command entry.)
In the example figure to the left, formatting is specified.



9) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "FORAMT OK" is displayed.
If the command ends abnormally, an error message or similar is displayed.
In the example figure to the left, a display where the command ends normally is shown.

10) "S>" is displayed in the line following the command execution result.
Enter the next command.

(2) Precautions on specifying each of the sizes in the CFORMAT command
- The overall size (total capacity) should be specified so that it matches with the capacity of the memory card to be formatted. Moreover, it must be equal to the total value of the sizes of the executable program area and file area.
- If all the remaining area, excluding the OS area, in the executable program area of the memory card is divided into eight BASIC task areas and each area has the same capacity, the maximum capacity of one area is approximately 48 k bytes.
- Sizes can be specified in hexadecimal digits ("&H□□□□ ") or binary digits ("&B□□ to □ "), instead of decimal digits.

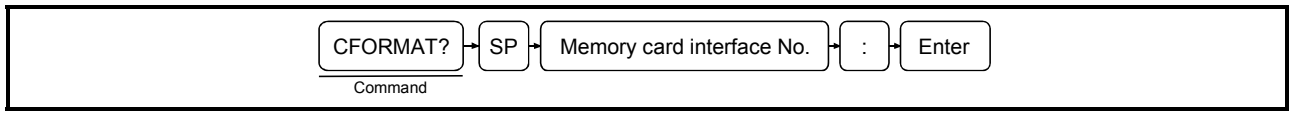
(3) About logical formatting of a memory card
- When the SET or MSAVE commands are executed for the first time, the executable program area of the memory card is logically formatted.
- Use the FFORMAT command for logical formatting of the file area.

(4) Reference
- Operation for displaying formatting information of a memory card : CFORMAT? command (Section 4.2.3)

4.2.3 Displaying Formatting Information of a Memory Card (CFORMAT? Command) For AD51H-S3 Only

This operation displays the formatting information of a memory card mounted in MEMORY CARD 1 or 2 on the AD51H-S3 module.

Input format (shortcut for the command CF?)

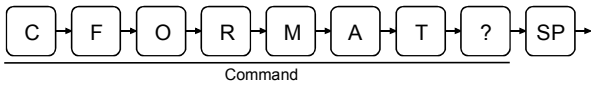


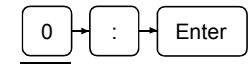
Operation example

Display the formatting information of a memory card mounted in MEMORY CARD 1 of the module.

Before input

```
S>
```

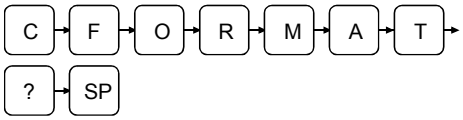




After input

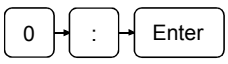
```
S>CFORMAT? 0:
    Card Name    : "TASK-DTM"
    Card Size    : 512K bytes (8)
    Program Size : 384K bytes (6)
    Canvas Size  :   0K bytes (0)
    File Size    : 128K bytes (2)
```

Description




```
S>CFORMAT?
```

1) Enter the CFORMAT? command to display the formatting information of the memory card.




```
S>CFORMAT? 0:
```

2) Enter the memory card interface number for the memory card for which the formatting information is to be displayed followed by a colon (:). Only 0 or 1 can be specified.
   0 : The MEMORY CARD 1 drive on the AD51H-S3.
   1 : The MEMORY CARD 2 drive on the AD51H-S3.

   Note that "0" may be omitted when specifying the memory card interface number. If it is omitted, simply press Enter. In the example figure to the left, the memory card in MEMORY CARD 1 is specified.

```
S>CFORMAT? 0:
   Card Name    : "TASK-DTM"
   Card Size    : 512K bytes (8)
   Program Size : 384K bytes (6)
   Canvas Size  :   0K bytes (0)
   File Size    : 128K bytes (2)
```

3) The screen displays the result of the command execution.
If the command ends normally, the formatting information of the specified memory card is displayed from the next line.
If the command ends abnormally, an error message or similar is displayed.
The following information is displayed when the command ends normally (see the figure to the left).

- Card Name : Memory card name assigned during formatting
- Card Size : The capacity corresponding to the value specified as the overall size during formatting (total capacity of the memory card)
  This is the value specified as the overall size during formatting.
- Program Size : The capacity corresponding to the value specified as the executable program area size during formatting (the capacity of the executable program area)
  The value in parentheses is the value specified for the executable program area size when the CFORMAT command was used to format the memory card.
- Canvas Size : Please ignore.
- File Size : The capacity corresponding to the value specified for the file area size during formatting (the capacity of the file area)
  The value in parentheses is the value specified for the file area size when the CFORMAT command was used to format the memory card.

4) "S>" is displayed on the line following the command execution result.
Enter the next command.

(1) Reference
- Operation for formatting a memory card : CFORMAT command (Section 4.2.2)

4.3 Operating Procedure for Loading/Saving Executable Programs

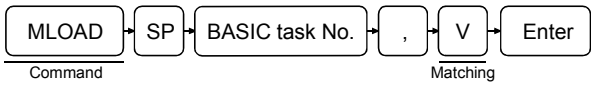
This chapter explains how to use each of the system commands for controlling executable program information and the operating procedure. These commands can be used to load an executable program in a BASIC task number area of the communication module to a memory card/EEP-ROM/flash ROM, and vice versa.

4.3.1 Loading Executable Programs to the Communication Module from a Memory Card/EEP-ROM/Flash ROM (MLOAD Command)

This operation loads an executable program from the specified BASIC task area in a memory card/EEP-ROM/flash ROM to the specified BASIC task area of the communication module.

Input format (shortcut for the command ML)

When checking that the contents of the load source and destination match after loading



When simply loading

Operation example

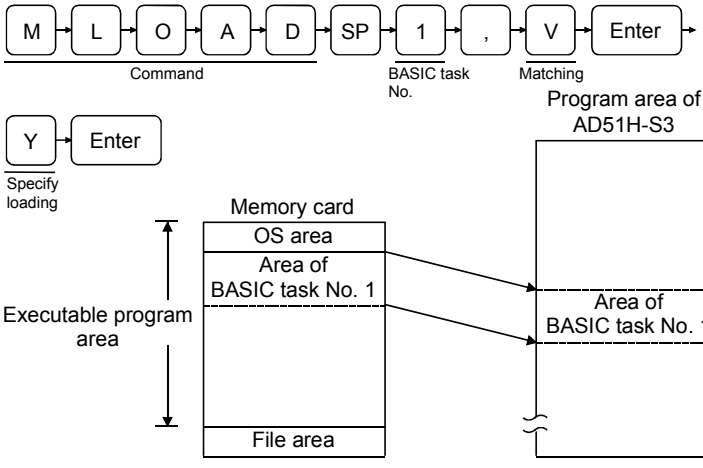
Load the executable program in the area of BASIC task No. 1 of the memory cards executable program area into the area of BASIC task No. 1 of the AD51H-S3 and check that their contents match.

Before input

```
S>
```

After input

```
S>MLOAD 1,V
LOAD(Y/N)?Y
LOAD OK
S>
```



(1) Target memory card

- The target memory card of the MLOAD command should be the memory card mounted in MEMORY CARD 1 of the AD51H-S3.

Description

M → L → O → A → D → SP

```
S>MLOAD
```

1) Enter the MLOAD command to load an executable program from a memory card/EEP-ROM/flash ROM into the executable program area of the communication module.

1

```
S>MLOAD 1
```

2) Enter the BASIC task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the executable program area of the memory card/EEP-ROM/flash ROM from which the executable program should be loaded.
In the example figure to the left, the executable program is loaded from the area of BASIC task No. 1.

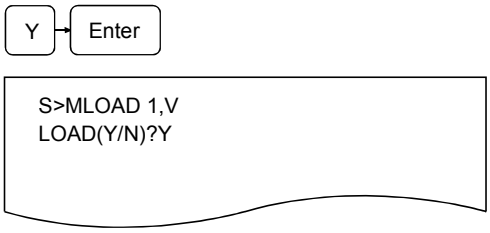
, → V → Enter

```
S>MLOAD 1,V
```

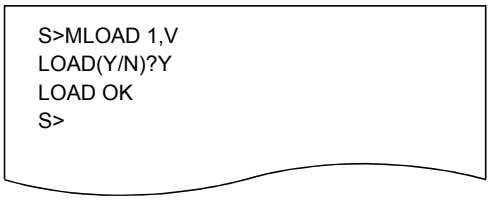
3) Specify "V" if it should be checked that the contents of the load source and destination match after loading.
Simply press Enter if it is not required to check that the contents match.
In the example figure to the left, the contents are checked after loading.

(2) Precautions when using the MLOAD command

- The size of the specified BASIC task area of the memory card/EEP-ROM/flash ROM (specified by the MSAVE or SET command) and the size of the corresponding BASIC task area of the communication module (specified by the START command) must be the same.
- Specify the interpreter in such a way that it does not run in the BASIC task area of the communication module to which the executable program of the memory/EEP-ROM/flash ROM is going to be saved.
  The operation of the interpreter should be terminated using the TKILL command if it is running.



4) The screen displays "LOAD (Y/N)? "
Enter [Y] to load.
Enter [N] to stop loading. (The console returns to waiting for a system command entry.)
In the example figure to the left, loading is specified.



5) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "LOAD OK" is displayed.
If the command ends abnormally, an error message or similar is displayed.
In the example figure to the left, a display where the command ends normally is shown.

6) "S>" is displayed in the line following the command execution result.
Enter the next command.

(3) Reference

- Operation for saving executable programs of the communication module to a memory card/EEP-ROM/flash ROM : MSAVE command (Section 4.3.2)
- Operation for specifying multitask settings and changing already set data : SET command (Section 4.4.1)
- Operation for displaying specified data of multitask settings : SET? command (Section 4.4.2)
- Operation for changing the mode of the communication module to the edit mode (1) : START command (Section 4.5.1)
- Operation for ending the operation of the interpreter in the specified BASIC task area : TKILL command (Section 4.6)

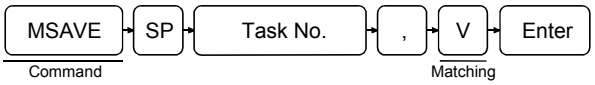
4 - 12

4.3.2 Saving Executable Programs to a Memory Card/EEP-ROM/Flash ROM from the Communication Module (MSAVE Command)

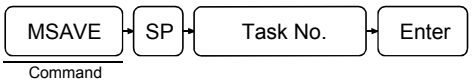
This operation saves an executable program in the specified BASIC task area of the communication module onto the target BASIC task area in a memory card/EEP-ROM/flash ROM. The multitask settings are automatically specified for the relevant task area by this operation.

Input format (shortcut for the command MS)

When checking that the contents of the save source and destination match after saving



When simply saving



Operation example

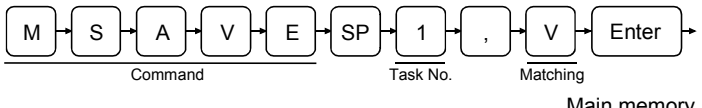
Save an executable program in the area of BASIC task No. 1 of the AD51H-S3 to the area of BASIC task No. 1 of the memory cards executable program area and check that the contents match.

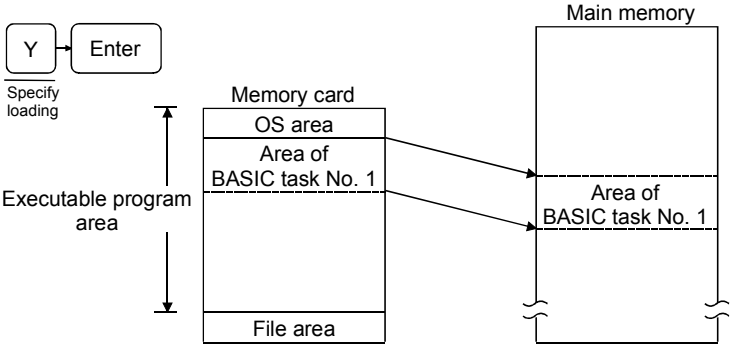
Before input

```
S>
```

After input

```
S>MSAVE 1,V
SAVE(Y/N)?Y
SAVE OK
S>
```





(1) Target memory card
- The target memory card of the MSAVE command should be the memory card mounted in MEMORY CARD 1 of the AD51H-S3.

(2) Precautions when using the MSAVE command
- Start up the interpreter with the START command, then execute the MSAVE command immediately after executing the SYSTEM command to the interpreter or pressing Ctrl + D.
- The following tasks should be performed again if the save capacity (the size specified by the START command) exceeds the capacity of the BASIC task area when saving again to a BASIC task area of a memory card to which executable programs have already been saved:
  1) Save all the executable programs to the executable program area of a memory card/EEP-ROM/flash ROM.
  2) Modify the setting contents of the multitask settings accordingly.

Description

M → S → A → V → E → SP

```
S>MSAVE
```

1) Enter the MSAVE command to save an executable program to a memory card/EEP-ROM/flash ROM from the communication module.

1

```
S>MSAVE 1
```

2) Enter the BASIC task area (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the communication module from which the executable program is going to be saved.
The illustrated on the left is an example for writing an execution program for the BASIC task No.1 area.

, → V → Enter

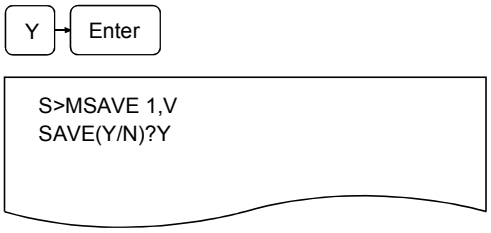
```
S>MSAVE 1,V
```

3) Specify "V" if it should be checked that the contents of the save source and destination match after saving.
Simply press [Enter] if it is not required to check that the contents match.
In the example figure to the left, the contents are checked after saving.

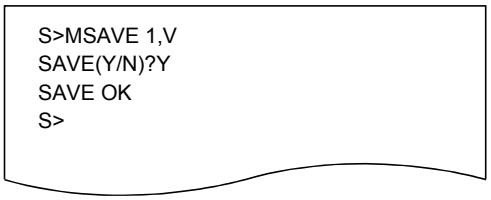
(3) Processing of the MSAVE command

- The contents of the memory corresponding to the size of the BASIC task area of the communication module specified by the START command are saved in the target BASIC task area of a memory card/EEP-ROM/flash ROM as an executable program.
- After saving the executable program, the multitask settings are automatically specified for the relevant BASIC task area.
  The following settings are specified. See the reference section in the SET command for details.
  Startup condition : The "BOOT" attribute is set.
  Size : The task size value specified at the START command execution is set.
  Startup order : No setting is made.

4 - 14



4) The screen displays "SAVE (Y/N)?"
Enter [Y] to save.
Enter [N] to stop saving. (The console returns to waiting for a system command entry.)
In the example figure to the left, the save is specified.



5) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "SAVE OK" is displayed.
If the command ends abnormally, an error message or similar is displayed.
In the example figure to the left, a display where the command ends normally is shown.

6) "S>" is displayed in the line following the command execution result.
Enter the next command.

(4) Reference

- Operation for saving executable programs from a memory card/EEP-ROM/flash ROM to the main memory : MLOAD command (Section 4.3.1)
- Operation for specifying multitask settings and changing the setting contents : SET command (Section 4.4.1)
- Operation for displaying the setting contents of the multitask settings : SET? Command (Section 4.4.2)
- Operation for changing the mode of the communication module to the edit mode (1) : START command (Section 4.5.1)

4 - 15

4.4 Operating Procedure for Specifying Multitask Settings, Changing Set Data, and Displaying Set Data

This chapter explains how to use each of the system commands for controlling multitask settings and the operating procedure to specify multitask settings, modify set data, and display set data.
Multitask settings refer to the startup condition settings used when starting up the communication module in run mode and executing multiple BASIC programs in multitasking.
The multitask settings include the following items. They are specified with the MSAVE command or the SET command.

(a) Startup conditions

Specifies the startup conditions under which the BASIC program in the target BASIC task area is executed.

1) START
   - After powering on or resetting the communication module, executable programs in the specified target BASIC task areas of a memory card/EEP-ROM/flash ROM are loaded into the corresponding executable program areas of the communication module, after which the programs are executed.

2) BOOT
   - Executable programs in the specified target BASIC task areas of a memory card/EEP-ROM/flash ROM are loaded into the corresponding executable program areas of the communication module when the communication module is started up.
   - They are executed when a currently running BASIC program directs an order to execute by the ZSTART instruction.

3) IT
   - Executable programs in the target BASIC task areas of a memory card/EEP-ROM/flash ROM are loaded into the executable program area of the communication module when the communication module is started up.
   - They are executed when the PLC CPU turns on the specific output (the startup task number specification flag and task startup signal) of the communication module.

4) ON
   - The specified programs are loaded from the file area of a memory area, etc. and executed when a currently running BASIC program directs an order by the ZSTART instruction after the communication module has been started up.

5) OFF
   - The multitask settings of the target task area are canceled. BASIC programs cannot be run in the target task areas.

(b) Task size

Set the size (16 k bytes, 32 k bytes, 48 k bytes, 64 k bytes) of the target BASIC task area.

(c) Startup order

Specify which program should be executed first when multiple BASIC programs are loaded into the corresponding task areas and executed when the communication module is started up.

If executable programs are saved into the executable program area (used as multiple BASIC task areas) of a memory card/EEP-ROM/flash ROM using the MSAVE command, the multitask settings are specified automatically for the target BASIC task areas. This section explains the available operations for the aforementioned multitask settings, and for changing and verifying set data.

(1) About changing the task size of the multitask setting

- The following tasks should be performed again if the size of the multitask setting is changed so that the size of the target task area exceeds the current size.
  1) Save all the executable programs in the communication module with the SAVE instruction.
  2) Change the set data in the multitask setting accordingly.
     (Specify each task size in such a way that all the executable programs can be saved within the executable program area size specified when the target memory card was formatted.)
  3) Reset the communication module.
  4) Load the executable programs with the LOAD instruction and execute the MSAVE command.

(2) See Section 4.3.2 for more information about the MSAVE command.

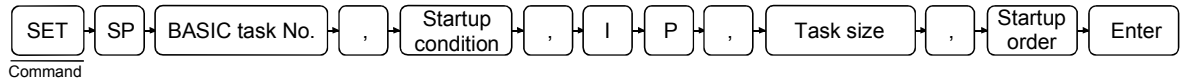
4 - 17

4.4.1 Specifying Multitask Settings and Changing Set Data (SET Command)

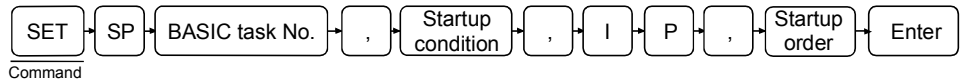
This operation allows the user to specify multitask settings for task areas for which multitask settings have not been specified and change the multitask settings of task areas that have already been set.

Input format (shortcut for the command S)

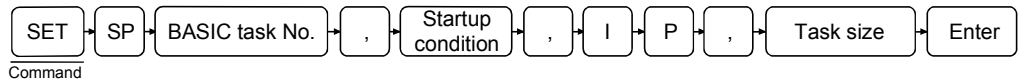
When setting/changing startup conditions, size, and startup order



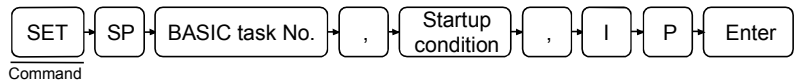
When changing startup condition and startup order



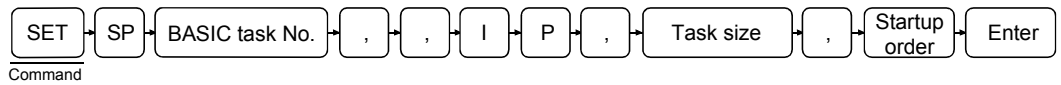
When changing startup condition and size



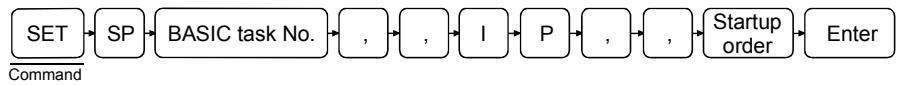
When changing startup condition



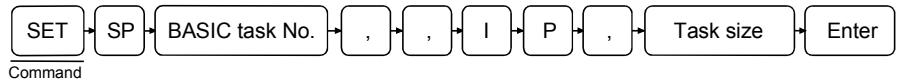
When changing size and startup order



When changing startup order



When changing size

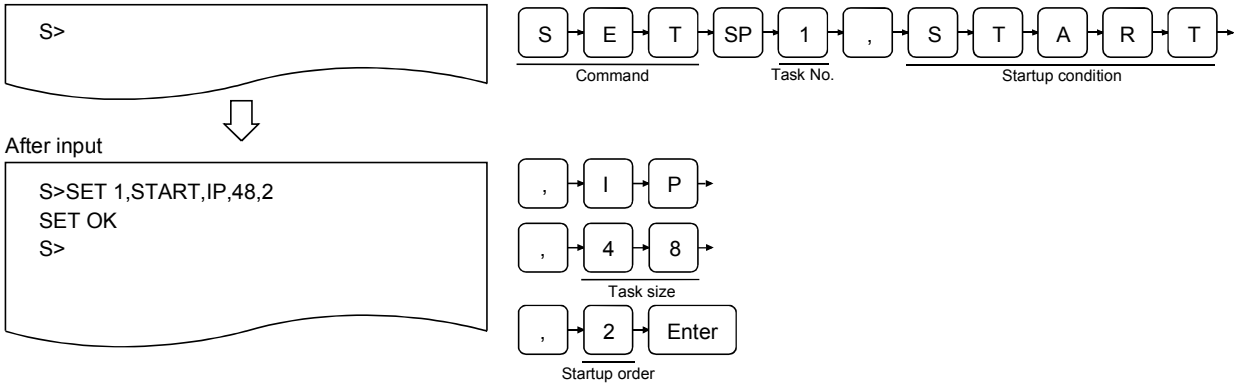


4 - 18

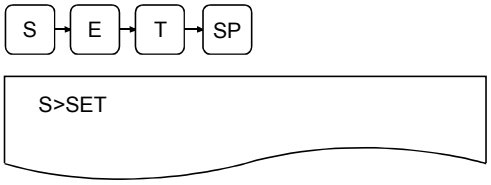
Operation example

Specify the multitask settings for BASIC task No. 1 area

Before input

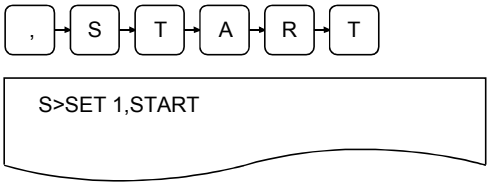


Description



1) Enter the SET command for specifying the multitask settings or changing the set data.

2) Enter the BASIC task area (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) for which the settings should be specified/changed.
In the example figure to the left, the multitask settings/ set data of BASIC task No. 1 area of the communication module will be specified/changed.

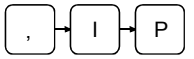


3) Enter one of the following attributes in order to specify/change the startup condition under which a BASIC program is executed in the target BASIC task area.
- START
- BOOT
- IT
- ON
- OFF

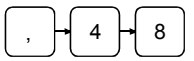
Simply enter a comma (,) if a startup condition is not to be specified.
In this case, it is assumed that the startup condition that has already been set will not be changed.
In the example figure to the left, the START attribute is set as the startup condition.

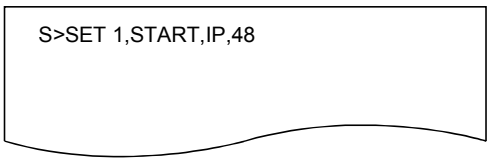
(1) See Section 4.4 for more information on the options for the startup condition.

4 - 19

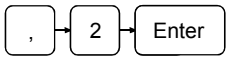


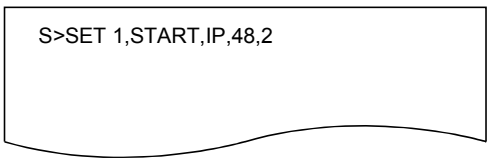
4) Enter IP as the type of program to be executed.



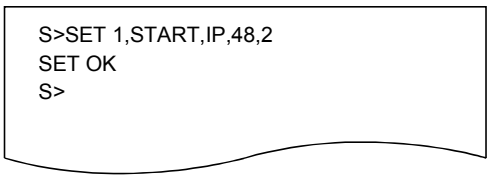


5) Enter one of the following values in order to set/change the task size of the target BASIC task area.
16, 32, 8, 64
Simply enter a comma (,) if a task size is not to be specified.
In this case, it is assumed that the current size of the target BASIC task area will not be changed.
In the example figure to the left, the task size is set to 48 K bytes.





6) Enter a number in the range from 1 to 8 in order to set/change the execution order (execution startup order) of programs in multiple BASIC task areas for which the "START" attribute is set as the startup condition when the communication module is initiated (1 is the top priority).
If the same number is set for multiple task areas, the program with the smaller task number is executed first.
Simply Enter a comma (,) if a startup order is not to be specified.
In this case, it is assumed that the startup order that has already been set will not be changed.
In the example figure to the left, a startup order of 2 is set.



7) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "SET OK" is displayed.
If the command ends abnormally, an error message or similar is displayed.
In the example figure to the left, a display where the command ends normally is shown.

8) "S>" is displayed in the line following the command execution result.
Enter the next command.

(2) About the size specification
- Sizes can be specified in hexadecimal digits ("&H□□□□") or binary digits ("&B□□ to □"), instead of decimal digits.

(3) Reference
- Operation for saving BASIC task area information of the communication module to a memory card/EEP-ROM/flash ROM : MSAVE command (Section 4.3.2)
- Operation for displaying the multitask settings : SET? Command (Section 4.4.2)
- Operation for changing the mode of the communication module to the edit mode (1) : START command (Section 4.5.1)

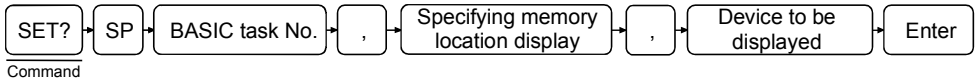
4 - 20

4.4.2 Displaying Set Data for Multitask Settings (SET? Command)

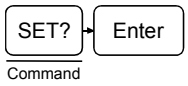
This operation displays the specified contents of the multitask settings for each BASIC task area of the communication module.

Input format (shortcut for the command S?)

When specifying one of the BASIC task areas



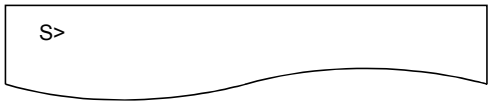
When specifying all the BASIC task areas

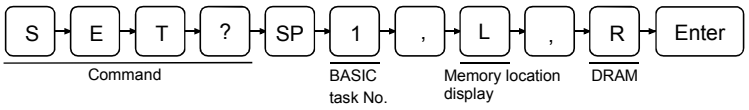


Operation example

Display the specified contents of the multitask settings for BASIC task No. 1 area of the communication module.

Before input

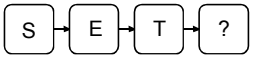




After input

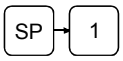
```
S>SET 1,L,R
          Task No. Type Size Start Condition Start No. Location
             1       IP   16          START     2          48

          Location Size Task No.
             48     16      1
             64     32      5
             96     16      -
            112     16      2
```

Description




```
S>SET?
```

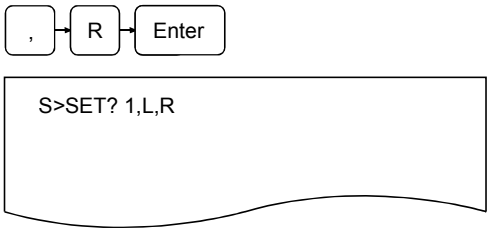
1) Enter the SET? command to display the contents of the multitask settings.




```
S>SET? 1
```

2) Enter the target BASIC task area (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) whose the set data is to be displayed.
Simply press Enter if all the BASIC task areas are to be specified.
In the example figure to the left, BASIC task No. 1 is specified.

3) Enter L if the location allocation of each task is to be displayed when booted to RAM.
Simply enter "," if the location allocation is not to be displayed.
The following information is displayed:
- Head location
- Size
- BASIC task No.



4) Enter the device (U/R) whose multitask settings are to be displayed.

U : Display the multitask settings of a user ROM.

R : Display the multitask settings booted on the current RAM.

4 - 22

5) The screen displays the result of the command execution. If the command ends normally, the multitask settings of the specified task area as well as the location allocation of each task number are displayed from the succeeding line.

```
S>SET? 1,L,R
          Task No. Type Size Start Condition Start No. Location
             1      IP   16        START     2          48

          Location Size Task No.
             48     16     1
             64     32     5
             96     16     -
            112     16     2
```

If the command ends abnormally, an error message or similar is displayed in the succeeding line. The following information is displayed when the command ends normally (in the example figure to the left, the settings for BASIC task No. 1 area are displayed). See the SET command explanation page for the meaning of each item of information displayed.

- Task No. : Task number of the task area displayed.
- Type : This corresponds to the IP/CP specification entered immediately after the startup condition is set with the SET command.
- Size : Size of the target task area. This corresponds to the "task size" specified by the SET command.
- Start Condition : The condition under which a BASIC program starts running in the target area. This corresponds to the "startup condition" specified by the SET command.
- Start No. : The execution startup order when START is set as the startup condition attribute (4) above). This corresponds to the "startup order" specified by the SET command. If the startup condition is different from the "START" attribute, the setting in this item is meaningless, and "-" is displayed.
- Location : This shows the memory location allocated for the task (in case of Type CP only).

6) "S>" is displayed in the line following the command execution result.
Enter the next command.

(2) Reference

- Operation for saving information from BASIC task areas in the communication module to a memory card/EEP-ROM/flash ROM : MSAVE command (Section 4.3.2)
- Operation for specifying the multitask settings/change the already set data : SET command (Section 4.4.1)
- Operation for changing the mode of the communication module to the edit mode (1) : START command (Section 4.5.1)

4 - 24

4.5 Operating Procedure for Changing the Mode of the Communication Module

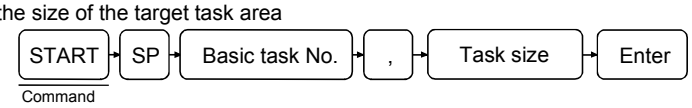
This section explains how to use each of the system commands for controlling modes and the operating procedure to change the mode of the communication module.

4.5.1 Changing the Mode of the Communication Module to the Edit Mode (1) (START Command)

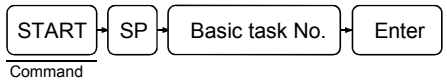
This operation allows the user to edit and debug each of the BASIC programs.

Input format (shortcut for the command ST)

When setting/changing the size of the target task area



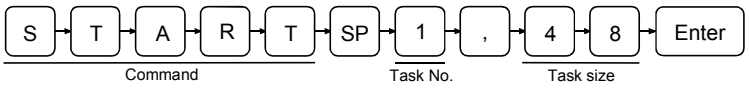
When the size of the target task area is not to be changed



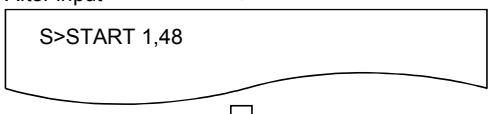
Operation example 1

Start editing a new BASIC program in the area of BASIC task No. 1. The size of the area of task No. 1 is set to 48 k bytes.

Before input



After input

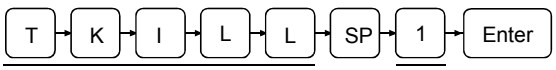


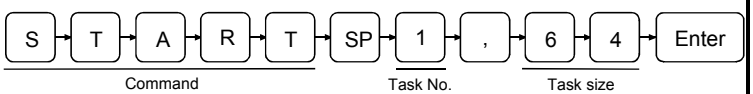
If the interpreter has not been started, the following message is displayed immediately before OK:
"AD51H-BASIC ON-LINE PROGRAMMING Ver[ ]"

Operation example 2

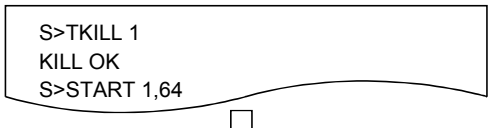
Start editing and debugging the BASIC program in the area of task No. 1. The size of the area of task No. 1 is changed to 64 k bytes.

Before input





After input



If the interpreter has not been started, the following message is displayed immediately before OK:
"AD51H-BASIC ON-LINE PROGRAMMING Ver[ ]"

Description

S → T → A → R → T → SP

```
S>START
```

1) Enter the START command to switch the mode of the communication module into edit mode (1).

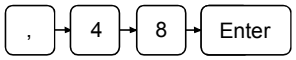
1

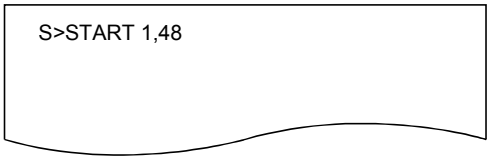
```
S>START 1
```

2) Enter the task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the task area in which a BASIC program is to be edited/debugged. The task number may be omitted.
If omitted, it is assumed that the next task number is specified.
- When the START command is entered for the first time, it is assumed that "1" is specified.
- If the START command has already been used, it is assumed that the task number specified by the last START command is specified.

In the example figure to the left, the BASIC program in the area of task No. 1 will be edited/debugged.

4 - 26





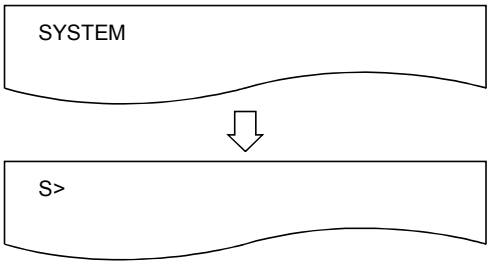
3) Enter one of the following numerical values in order to set/change the task size of the target task area (in K byte units)
16, 32, 48, 64
The task size entered here will be the size set in the automatic multitask setting, which is specified when saving the contents of the target BASIC task area to a memory card/EEP-ROM with the MSAVE command after the completion of the BASIC program editing/debugging.
Make sure to enter the task size if a BASIC task number whose multitask settings have not already been specified has been selected.
Also, make sure to enter the task size if the multitask settings have already been specified, but the task size change is required.
Simply press Enter if the task size already set is not to be changed.
In the example figure to the left, the task size of the area of BASIC task No. 1 is set/changed to 48 K bytes.

Or

4) The screen displays the result of the command execution.
If the command ends normally, the display shows in the figure to the left; the BASIC program can be edited/debugged.
The operating procedure for editing/debugging BASIC programs is explained in the AD51H-BASIC Programming Manual.
If the command ends abnormally, an error message or similar is displayed in the succeeding line.
The upper figure to the left shows the display when the interpreter has not been started.
The lower figure to the left shows the display when the interpreter has already been started.

(1) About the size specification
- Sizes can be specified in hexadecimal digits ("&H□□□□") or binary digits ("&B□□ to □") instead of decimal digits.

4 - 27



5) Perform one of the following operations when the editing/debugging of the BASIC program is completed and the communication module is returned from edit mode (1) to system mode.

[Execute the SYSTEM instruction of BASIC program to stop]

- The execution of the BASIC program is stopped.
- All open files and communication lines are closed.

[Press Ctrl + D]

- The execution of the BASIC program is stopped.
- Open files and communication lines are kept open.
- If the BASIC program whose execution was stopped did not require modification, its execution can be restarted (continued) with the CONT instruction of the Basic program when the START command is used to change the mode to edit mode (1) again.

(2) Precautions when using the START command

- In cases where a BASIC program is edited/debugged in a task area that falls into one of the categories listed below and the task size must be changed, the operation of the interpreter in the target task area should be terminated using the TKILL command before entering the START command.
  1) Task areas whose multitask settings have been specified
  2) Task areas where BASIC programs are already stored

  In addition, if the task size is increased, the following tasks should be performed once again when saving a BASIC program to the execution area of a memory card/EEP-ROM/flash ROM (MSAVE command) after the completion of editing and debugging.
  3) All the executable programs should be saved again in the execution area of the memory card.
  4) The set data of the multitask settings should be changed accordingly. (Set each task size in such a way that the maximum 8 units of executable programs can be saved within the executable program area size specified when the target memory card was formatted.)

(3) About debugging BASIC programs after the START command execution

- Debug BASIC programs in edit mode (1) according to the method explained in the programming manual.
- The system commands listed in Chapter 4 cannot be used.

(4) Reference

- Operation for saving task area information from the main memory to a memory card : MSAVE command (Section 4.3.2)
- Operation for changing the mode of the communication module : GO command (Section 4.5.2)
- Operation for ending the interpreter operation in the specified task area : TKILL command (Section 4.6)

4.5.2 Changing the Mode of the Communication Module to Run Mode/System Mode (GO Command)

This operation changes the mode of the communication module from system mode to run mode/multitask debug mode/run mode, or changes it back to system mode.
By changing to multitask debug mode, it becomes possible to debug each BASIC program executed with the multitask settings by entering debug commands to the debugger terminal (see Chapter 5).
By changing to run mode, each BASIC program starts running according to the multitask settings.
By changing back to system mode, it becomes possible to edit/debug each BASIC program by entering system commands to the console.
The following table lists the relationship between the mode and debugging start specification when the GO command is entered, and the statuses of the console and the debugger terminal after the GO command has been executed.

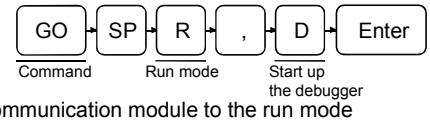
<table>
<tr><th>Mode setting</th><th>Debugging startup specification Yes/No</th><th>Console status</th><th>Status of debugger terminal</th><th>Remarks</th></tr>
<tr><td rowspan="2">R
(Run mode)</td><td>Yes
(To multitask debug mode)</td><td rowspan="2">The contents of the display are deleted. The console display changes to the one used for BASIC programs.</td><td>The debugger initiates, the contents of the display are deleted, and "D>" is displayed. It becomes possible to enter debug commands.</td><td rowspan="2">Each BASIC program is reloaded to the corresponding task area according to the multitask settings, and is executed.</td></tr>
<tr><td>No
(To run mode)</td><td rowspan="2">The contents of the display are kept as is. The terminal becomes a general-purpose port for BASIC programs.</td></tr>
<tr><td>P
(System mode within programming mode)</td><td>Cannot be specified.</td><td>The contents of the display are deleted and "S>" is displayed. It becomes possible to enter system commands.</td><td>BASIC programs in each task area stop being executed.</td></tr>
</table>

(1) Status of each BASIC program due to execution of the GO command
- When Run mode has been specified, BASIC program execution is started.
  This behavior is the same with the case where the communication module is started up with Run or Multitask debug mode setting.

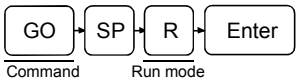
4 - 29

Input format (shortcut of the command None)

When changing the mode of the communication module to the multitask debug mode



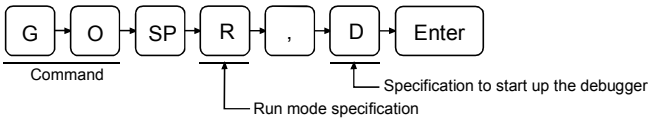
When changing the mode of the communication module to the run mode



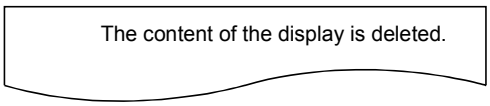
Operation example

Change the mode of the communication module to multitask debug mode.

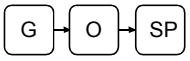
Before input



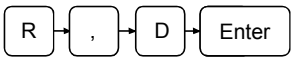
After input



Description



1) Enter the GO command to change the mode of the communication module.

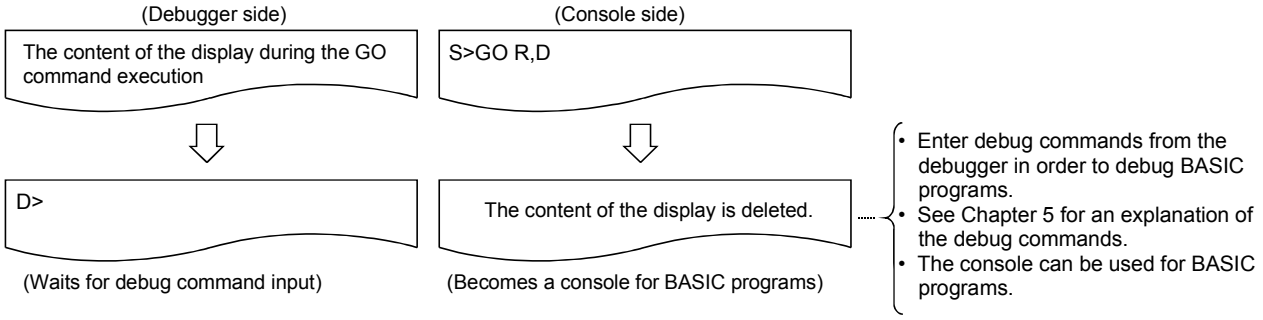


2) Enter the mode.
Enter R to change to run mode.
Enter R, D to change to multitask debug mode.
In the example figure to the left, the mode of the communication module is changed to multitask debug mode.

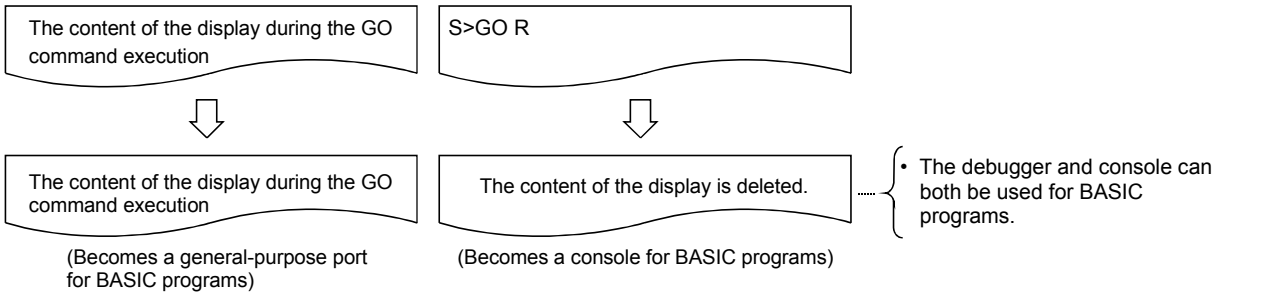
3) The screen displays the result of the command execution.
If the command ends normally, the display shows as follows depending on the specifications.
If the command ends abnormally, an error message or similar is displayed in the succeeding line.

The example below illustrates the contents of the display when the command ends normally.

1) When the mode is changed to the multitask debug mode



2) When the mode is changed to the run mode



(2) About changing the modes of the communication module
- See Section 2.2 for the mode change diagram of the communication module.

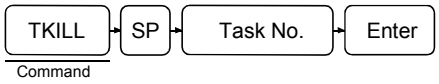
(3) Reference
- Operation for displaying the main menu screen on the console : EXIT command (Section 4.7)

4 - 31

4.6 Ending the Interpreter Operation in the Specified Task Areas (TKILL Command)

This section explains how to use the system command TKILL for controlling the interpreter operation and the operating procedure to end the operation of the interpreter in specified task areas.

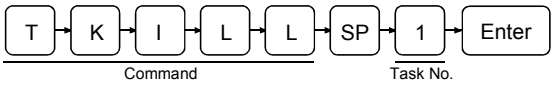
Input format (shortcut for the command TK)



Operation example

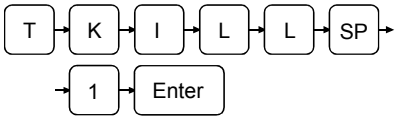
End the operation of the interpreter in the area of BASIC task No. 1.

Before input



After input

Description



1) Enter the TKILL command to end the operation of the interpreter in the specified task area or one of the tasks (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the target task area.
The example figure to the left shows how to end the operation of the interpreter in the area of task No. 1.

2) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "TKILL OK" is displayed.
If the command ends abnormally, an error message or similar is displayed.
In the example figure to the left, a display where the command ends normally is shown.

3) "S>" is displayed in the line following the command execution result.
Enter the next command.

(1) Usage of the TKILL command

The operation of the interpreter should be ended in the target task area before executing the GO command when performing the following operations.

1) Change the mode of the communication module to system mode and change the task size of a task area using the system commands START/SET.
2) Change the mode of the communication module to system mode and load executable programs from the specified BASIC task area in a memory card/EPP-ROM to the target task area of the communication module using the system command MLOAD.

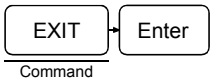
(2) Reference

- Operation for changing the mode of the communication module to edit (1) mode : START command (Section 4.5.1)
- Operation for changing the mode of the communication module : GO command (Section 4.5.2)

4.7 Operating Procedure for Displaying the Main Menu Screen on the Console (EXIT Command)

This section explains how to use the EXIT command to display the main menu screen on the console.

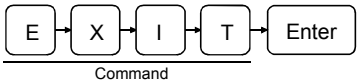
Input format (shortcut for the command E)



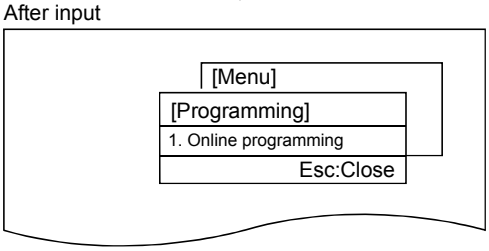
Operation example

The main menu screen is displayed in the console.

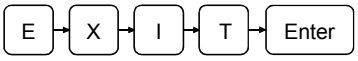
Before input



After input



Description



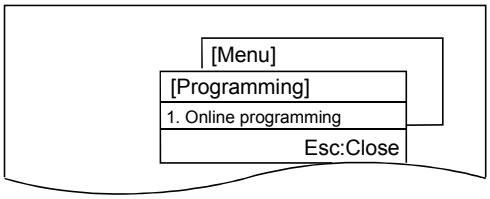
1) Enter the EXIT command to display the main menu screen.

(1) Status of each BASIC program by the execution of the EXIT command

The BASIC programs in each BASIC task area execute continuously even when the EXIT command is executed.

(2) Precaution when entering the command

Stop the execution of BASIC programs before entering the EXIT command so that the execution does not interfere with the system control when displaying the main menu to edit BASIC programs in each BASIC task area, etc.



2) The screen displays the result of the command execution; the user can perform necessary operations from this point.

If the command ends normally, the main menu screen is displayed on the console; select a displayed item to perform a necessary operation.
The content is displayed when the command ends normally.
See the following manual for an explanation of the operations from the main menu screen:
AD51H-BASIC Package type SW1IVD-AD51HP-E Operating Manual

If the command ends abnormally, an error message or similar is displayed.

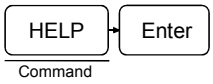
(2) About changing the mode of the communication module
See Section 2.2 for the mode change diagram of the communication module.

(3) Reference
Operation for changing the mode of the communication module : GO command (Section 4.5.2)

4.8 Operating Procedure for Checking the Input Formats of the System Commands (HELP Command)

This section explains how to use the system command HELP to display the input format, etc. of each of the system commands on the console.

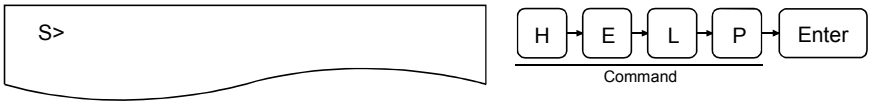
Input format (shortcut for the command H)



Operation example

Display input format, etc. of each system command.

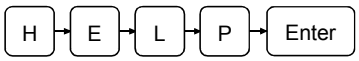
Before input



After input

```
S>H
( 1)CCOPY   Memory Card Copy   CC {Source Drive No.:}
                                  ,{Destination Drive No.:}
                                  [,V]
( 2)CFORMAT Memory Card Format CF {Drive No.:}
```

Description




```
S>HELP
```

1) Enter the HELP command to display input format, etc. of each system command.

4 - 36

2) The screen displays the result of the command execution.
If the command ends normally, functions/input formats of the system commands are displayed from the succeeding line.

```
S>H
( 1)CCOPY   Memory Card Copy   CC {Source Drive No.:}
                                ,{Destination Drive No.:}
                                [,V]
( 2)CFORMAT Memory Card Format CF {Drive No.:}
```

Press any key other than ESC to display functions/input formats of system commands in the suceeding page. Press the ESC key to end the HELP command.
(Example)

(1) (a)	CCOPY (b)	Memory Card Copy (c)	CC {Source Drive No.}: ,{Destination Drive No.}: [,V] (d)

(a) Number for description
(b) Command
(c) Function of command
(d) Description of input format (shortcut for command)

If the command ends abnormally, an error massage or similar is displayed in the succeeding line.

3) "S>" is displayed in the line following the command execution result.
Enter the next command.

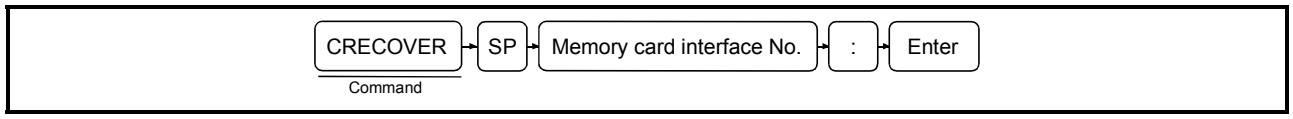
(1) About displaying the command input format
A one column space entered immediately after a command indicates the entry of the SP key (space).
Brackets ("{" and "}") are symbols that indicate separation of command arguments; it is not necessary to enter brackets.
Square brackets ("[" and "]") are symbols that indicate that the arguments inside them are optional; it is not necessary to enter square brackets.

4.9 Recovering an Area in Unusable File Area in a Memory Card (CRECOVER Command)

For AD51H-S3 Only

This operation locates a data area that is in the unusable status in the file area of a memory card mounted in the specified drive and recovers it to the usable status again.

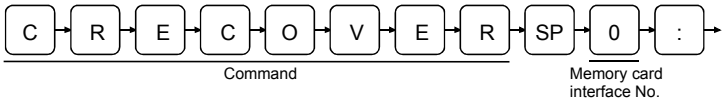
Input format (shortcut of the command CR)



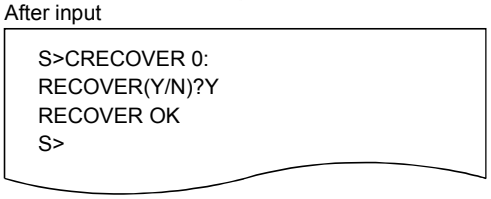
Operation example

Recover an unusable file area in the memory card mounted in the modules MEMORY CARD 1.

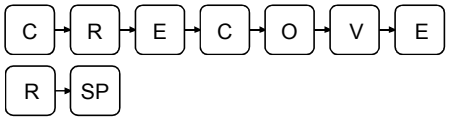
Before input

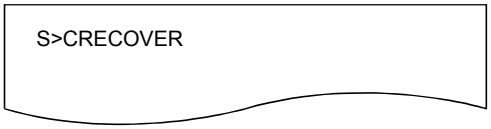


After input

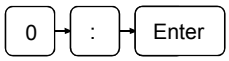


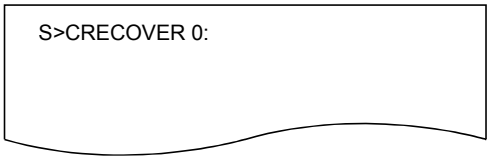
Description





1) Enter the CRECOVER command to recover a file area of a memory card.



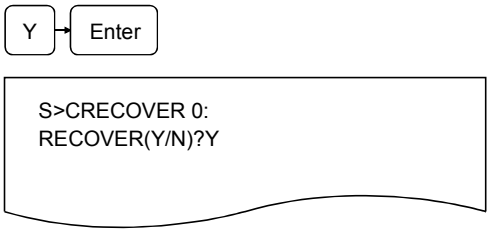


2) Enter the memory card interface number of the memory card whose file area is to be recovered followed by a colon (:). Only 0 or 1 can be specified for the memory card interface number.

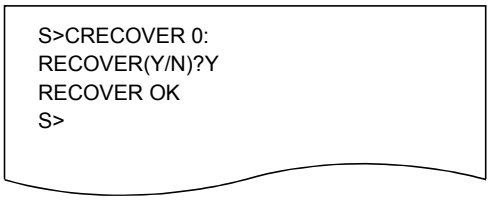
0 : The MEMORY CARD 1 drive on the AD51H-S3.

1 : The MEMORY CARD 2 drive on the AD51H-S3.

In the example figure to the left, the memory card mounted on MEMORY CARD 1 is specified.



3) The screen displays "RECOVER (Y/N)?"
Enter [Y] to recover.
Enter [N] to stop recovering. (The console returns to waiting for a system command entry.)
In the example figure to the left, recovery is specified.



4) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "RECOVER OK" is displayed.
If the command ends abnormally, an error message or similar is displayed.
In the example figure to the left, a display where the command ends normally is shown.

5) "S>" is displayed in the line following the command execution result.
Enter the next command.

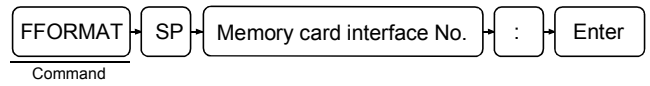
4 - 39

4.10 Formatting (Logical Format) the File Area of a Memory Card (FFORMAT Command)

For AD51H-S3 Only

This operation formats (logically) the file area of a memory card mounted in MEMORY CARD 1 or 2 on the AD51H-S3.

Input format (shortcut of the command FFM)

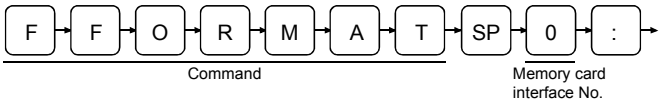


Operation example

Format the memory card mounted in MEMORY CARD 1.

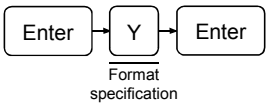
Before input

```
S>
```

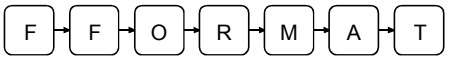


After input

```
S>FFORMAT 0:
FORMAT(Y/N)?Y
FORMAT OK
S>
```

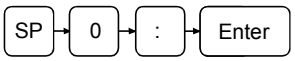


Description




```
S>FFORMAT
```

1) Enter the FFORMAT command to format a memory card.




```
S>FFORMAT 0:
```

2) Enter the memory card interface number in which the memory card to be formatted is mounted followed by a colon (:). Only 0 or 1 can be specified.

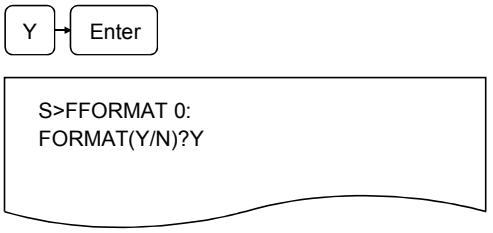
   0 : The MEMORY CARD 1 drive of the AD51H-S3.

   1 : The MEMORY CARD 2 drive of the AD51H-S3.

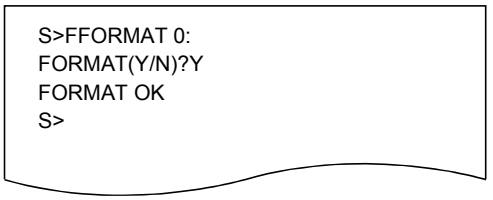
   In the example figure to the left, the memory card mounted in MEMORY CARD 1 is specified.

(1) Precautions when using the FFORMAT command

- If a memory card is formatted, all data that was saved on it will be deleted.
- When formatting a memory card that is write-protected, the write protect should be canceled first.
- When formatting a memory card mounted in MEMORY CARD 1, the memory protection key switch of the AD51H-S3 module should be turned off first.



3) The screen displays "FORMAT (Y/N)?"
Enter [Y] to format.
Enter [N] to stop formatting.
In the example figure to the left, formatting is specified.



4) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "FORMAT OK" is displayed.
If the command ends abnormally, an error message or similar is displayed.
In the example figure to the left, a display where the command ends normally is shown.

5) "S>" is displayed in the line following the command execution result.
Enter the next command.

(2) Reference
Operation for physically formatting a memory card : CFORMAT command (Section 4.2.2)

4 - 41

5 MULTITASK DEBUGGING OPERATIONS

Multitask debugging refers to operations for finding and correcting errors in each program while executing multiple BASIC programs at the same time.

This chapter explains how to use the debug commands entered from the debugger in order to start executing BASIC programs with multitask settings and debug each BASIC program.

(1) This chapter mainly explains the key inputs and displays on the debugger side.
It is therefore generally omitted to state this fact explicitly for most key inputs and displays.
When necessary, it is pointed out explicitly that key inputs and displays are on the console side.

(2) It is necessary to perform the following tasks in advance in order to perform the multitask debugging described in this chapter.
Perform each operation beforehand according to the explanation in the applicable chapters/sections below.

- Setting the switches of the communication module for debugging : See Chapter 2
- Connecting the debugger terminal : See Chapter 2.
- Creating and debugging each individual BASIC program : See Chapter 4.
- Saving the programs to a memory card See Section 4.3.2.
- Specifying multitask settings : See Section 4.4.1.

(4) Precautions when entering the debug commands
If the debugger function (an OS that analyzes and executes debug commands) cannot immediately process a command the user has entered, the debugger function suspends operation until the processing can be resumed.
The next debug command can be entered after "D>" is displayed again.

5.1 Debug Command List

Table 5.1 lists the debug commands used in multitask debugging operations.

Table 5.1 Debug Command List

Classification	System command	Function overview		Availability for module			Reference section
				AD51H-S3	A1SD51S	QD51 (-R24)	
Task control	TSTATUS ＊1	Displays the status of the BASIC program residing in the specified task area.		○	○	○	Section 5.2.1
	TRUN ＊1	Starts executing the BASIC program residing in the specified task area.					Section 5.2.2
	TSTOP ＊1	Stops executing the BASIC program currently being executed in the specified task area.					Section 5.2.3
	TCONTINUE ＊1	Resumes executing the BASIC program in the specified task area that has been stopped.					Section 5.2.4
	T? ＊1	Displays values of specified variables in the BASIC program residing in the specified task area.					Section 5.2.5
	TLET ＊1	Assigns value to specified variables in the BASIC program residing in the specified task area.					Section 5.2.6
Memory access control	MREAD	Displays memory values that can be shared among BASIC programs.	• Buffer memory • Common memory • Extension registers (ED)	○	○	○	Section 5.3.1
	MWRITE	Writes values to specified addresses in memory that can be shared by different BASIC programs.					Section 5.3.2
	B@	Displays bit information of an internal device that can be shared by input/output signals to the PLC CPU as well as BASIC programs.	• General-purpose inputs (X) • General-purpose outputs (Y) • Extension relays (EM)				Section 5.3.3
	B@	Writes bit information to an internal device that can be shared by input/output signals to the PLC CPU as well as BASIC programs.					Section 5.3.4
	W@	Displays word information of an internal device that can be shared by different BASIC programs.	• Extension registers (ED)				Section 5.3.5
	W@	Writes word information to an internal device that can be shared by different BASIC programs.					Section 5.3.6
OS information check	ZSTATUS	Displays usage statuses of events/message ports/resources that can be shared by different BASIC programs.		○	○	○	Section 5.4.1 Section 5.4.2 Section 5.4.3
Mode control	START ＊1	Changes the mode of the communication module from multitask debug mode to edit mode (2). (For program editing during multitask execution)		○	○	○	Section 5.5.1
	GO	Changes the mode of the communication module from multitask debug mode to system mode/run mode, or changes back to multitask debug mode.					Section 5.5.2
Others	EXIT	Displays the main menu screen on the debugger terminal.		○	○	○	Section 5.6
	HELP	Displays the list of debug commands, function overviews, and command input formats.					Section 5.7

＊1 These commands cannot be executed on tasks in which compiled BASIC programs reside.

5 - 2

5.2 Operations for Controlling the Operation of BASIC Programs

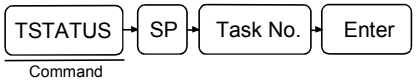
This section explains how to use each of the debug commands for controlling tasks and the operating procedure to control the operation of BASIC programs.

5.2.1 Displaying the Status of the Specified BASIC Program (TSTATUS Command)

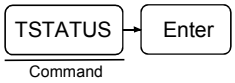
This operation displays the status of the BASIC program in the specified task area.

Input format (shortcut for the command TS)

When specifying one of the task areas



When specifying all the task areas

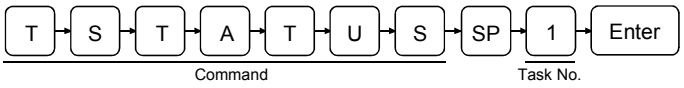


Operation example

Display the status of the BASIC program in the area of task No.1.

Before input

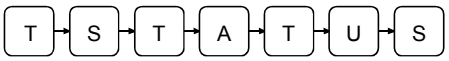
```
D>
```



After input

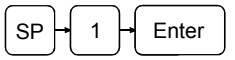
```
D>TSTATUS 1
TASK NO STATUS PRIORITY STEP NO
    1      WAIT        1         150
```

Description




```
S>TSTATUS
```

1) Enter the TSTATUS command to order to display the status of a BASIC program.




```
S>TSTATUS 1
```

2) Enter the task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the task area whose status is going to be displayed.
Simply press Enter if all task areas are to be displayed.
In an example figure to the left, task No. 1 is specified.

```
D>TSTATUS 1
TASK NO STATUS PRIORITY STEP NO
    1    WAIT      1        150
```

3) The screen displays the result of the command execution in the next line.
If the command ends normally, the status of the BASIC program in the specified task area is displayed in the center of the display.
If the command ends abnormally, "Error" and an error code are displayed in the succeeding line.
The following information is displayed when the command ends normally. (In an example figure to the left, the status of the area of task No. 1 is displayed.)

- TASK NO : Task number of the task area whose status is displayed.
- STATUS : Status of the BASIC program
  DORMANT: The interpreter has not been started in the target area.
  RUN: The program is being executed.
  WAIT: In waiting status. (Waiting for a timeout, etc.)
  STOP: The program is not being executed. The interpreter is waiting for command entry. (＊1)
- PRIORITY : Current priority of the BASIC program. The value 0 is displayed if [STATUS] above is DORMANT.
- STEP ON : The current line number being executed. The number of the last executed line is displayed If [STATUS] above is STOP. If it is DORMANT, the value 0 is displayed.

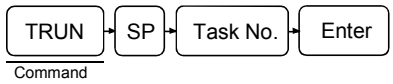
4) "D>" is displayed in the line following the command execution result.
Enter the next command.

＊1: The program is also forced into the STOP status when the execution of the specified BASIC program is stopped using the debug command "TSTOP."

5.2.2 Starting the Execution of the Specified BASIC Program (TRUN Command)

This operation starts the execution of the BASIC program residing in the specified task area.

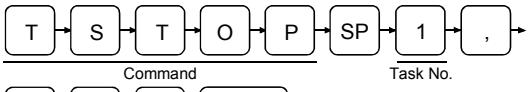
Input format (shortcut of the command TR)

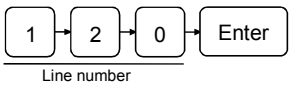


Operation example

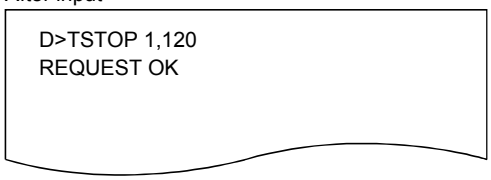
Stop the execution at line 120 of the BASIC program being executed in the area of task No. 1.

Before input

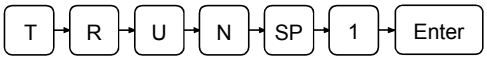




After input



Description



1) Enter the TRUN command to order to start executing a BASIC program along with the task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the task area where the target program resides. In an example figure to the left, the BASIC program residing in the area of task No. 1 starts executing.

2) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "REQUEST OK" is displayed.
If the command ends abnormally, an error message and error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.

3) "D>" is displayed in the line following the command execution result.
Enter the next command.

(1) Precautions when entering the command
- If a task area is specified in which a BASIC program is already being executed, an error occurs. If there is no BASIC program in the specified task area, an error occurs.

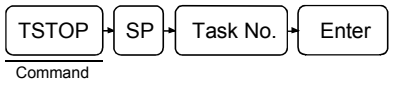
(2) Reference
- Operation for stopping the execution of the specified BASIC program : TSTOP command (Section 5.2.3)

5.2.3 Stopping the Execution of the Specified BASIC Program (TSTOP Command)

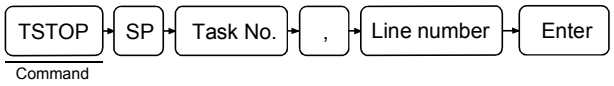
This operation stops the execution of the BASIC program in the specified task area.

Input format (shortcut for the command TP)

When stopping at the line currently being executed



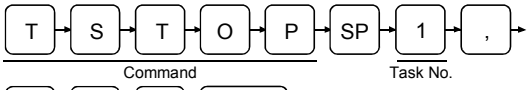
When stopping at the specified line

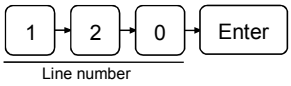


Operation example

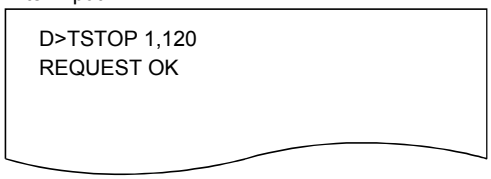
Stop the execution at line 120 of the BASIC program being executed in the area of task No. 1.

Before input

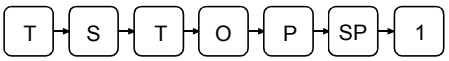




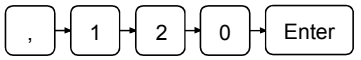
After input



Description



1) Enter the TSTOP command to stop the execution of a BASIC program along with the task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the task area where the target program is being executed.
   In an example figure to the left, the BASIC program being executed in the area of task No. 1 stops executing.




```
D>TSTOP 1,120
```

2) Enter the line number of the BASIC program whose execution is to be stopped using a decimal number.
Simply press Enter if it is desired to stop the execution at the line currently being executed.
Note that if this command is entered again by specifying the values 65535 or -1 as the line number before executing the line specified by the line number entry, the first execution stop specification is canceled.
In an example figure to the left, it is specified to stop execution at line number 120.

```
D>TSTOP 1,120
REQUEST OK
```

3) The screen displays the result of the command execution in the next line.
If the command is received normally, "REQUEST OK" is displayed.
In addition, the following stop message is displayed when the BASIC program stops executing.

BREAK:Task No.□ Line No.□□□

The line number at which the execution was stopped.

The task number of the task area for which the execution was stopped.

Note that if 65535 or -1 was specified as a line number and the command was received normally, "BREAK Cancel: Task No. □ " is displayed.
If there was an error in the command entry, "Error" message and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.
If the command ends normally, the target BASIC program goes into the STOP status and the execution can be resumed with the TCONTINUE command.

4) "D>" is displayed in the line following the command execution result.
Enter the next command.

(1) Precautions when entering the command

- When specifying the line number, a line number that exists in a program should be entered using the same decimal format as the description format in the program.
  If a line number that does not exist in the program is entered, the program will not be stopped. In this case, specify 65535 or -1 as a line number and enter this command again.
- It is only possible to specify one line number at which the execution should be stopped for any one BASIC program in any one task area.
- The execution of a BASIC program should be stopped in such a way that it does not interfere with the system control.

(2) Reference

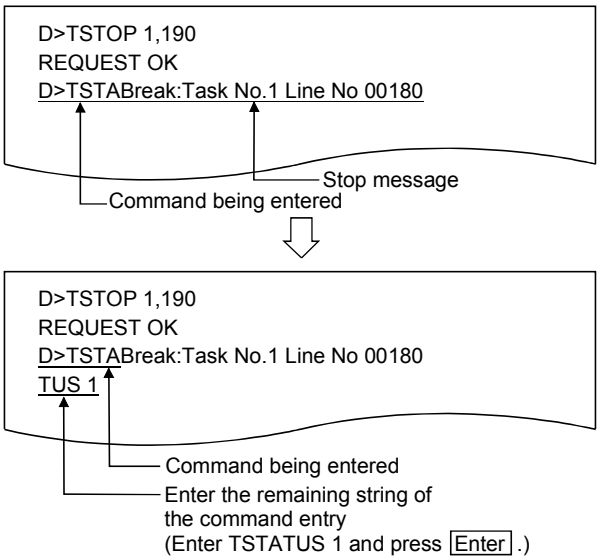
- Operation for checking the status of the current BASIC program : TSTATUS command (Section 5.2.1)
- Operation for starting the execution from the start line again : TRUN command (Section 5.2.2)
- Operation for resuming (continuing) execution from a line at which execution was stopped : TCONTINUE command (Section 5.2.4)
- Operation for checking values of specified variables : T? command (Section 5.2.5)
- Operation for assigning values to specified variable : TLET command (Section 5.2.6)

(3) Operation of the TSTOP command

- If a line number is specified at the TSTOP command entry, the program execution is stopped before executing the line with the specified number. Therefore, none of the instructions in the specified line have been executed when the execution of the BASIC program is stopped by the command.
  If a line number was not specified, the execution is stopped after the interpreter processes the instruction that is being executed at the time of pressing the Enter key. Therefore, if multiple instructions were entered in one line (multi-statement), the instructions after the instruction being executed at the time the Enter key was pressed would not have been executed.

(4) Precautions when stopping the execution of a BASIC program by entering the TSTOP command

- When a BASIC program stops being executed with the TSTOP command, the debugger (OS) displays a stop message at the current cursor position.
  If the user enters the command while the debugger displays the stop message, the message and command are mixed and both are displayed.
  In this case, the command being entered is valid; continue entering the remaining text string of the command entry.
  (Example) If a stop message is displayed while entering the TSTATUS command:



(5) How to end the operation of the interpreter in the specified task area

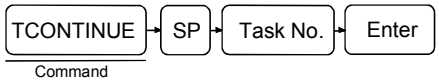
- If the user wants to end the operation of the interpreter in the specified task area, the program should be created in such a way as to execute the BASIC instruction "END."

5.2.4 Resuming the Execution of the Specified BASIC Program Whose Execution Has Been Stopped (TCONTINUE Command)

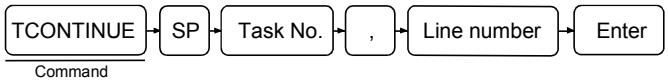
This operation resumes the execution of the BASIC program in the specified task area whose execution was stopped by the TSTOP command.

Input format (shortcut for the command TC)

When resuming execution from the instruction after the last instruction executed when the execution was stopped



When resuming execution from the specified line number

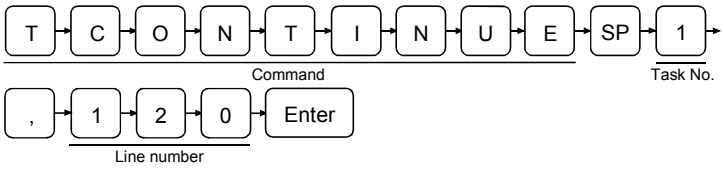


Operation example

Resume execution of the BASIC program in the area of task No. 1 which is currently being stopped, from line number 120.

Before input

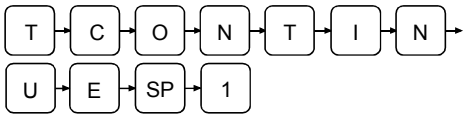
```
D>TCONTINUE 1,120
```



After input

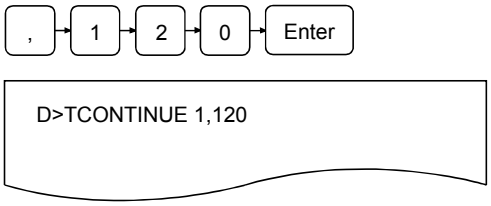
```
D>TCONTINUE 1,120
REQUEST OK
D>
```

Description

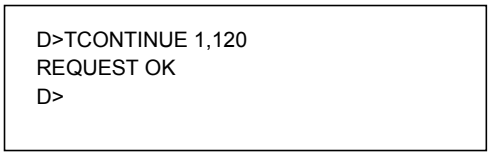



```
D>TCONTINUE 1
```

1) Enter the TCONTINUE command in order to resume execution of a BASIC program whose execution was stopped by the TSTOP command and the task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the task area where target program resides.
In an example figure to the left, the BASIC program in the area of task No. 1 which is currently being stopped, is being resumed.



2) Enter a line number from which execution is to be resumed using a decimal number.
Simply press Enter if the execution should be resumed from the instruction after the last instruction executed when the execution was stopped.
In an example figure to the left, line number 120 is specified.



3) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "REQUEST OK" is displayed.
If the command ends abnormally, "Error" and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.
If the command ends normally, the BASIC program in the specified task area is placed in the RUN status.

4) "D>" is displayed in the line following the command execution result.
Enter the next command.

(1) Precautions when entering the command
- The TCONTINUE command can be executed on a BASIC program in the STOP status whose execution was stopped by the TSTOP command.
  An error occurs if the command is entered for a BASIC program that is in a status other than STOP.
  In addition, if a task area that contains a BASIC program in the STOP status is specified, and the mode of the communication module is changed to edit mode (2), the execution of this BASIC program in the STOP status cannot be resumed using the TCONTINUE command. It cannot even be resumed if the mode of the module is returned to multitask debug mode by executing the SYSTEM instruction.
  The status of a BASIC program can be checked using the TSTATUS command.

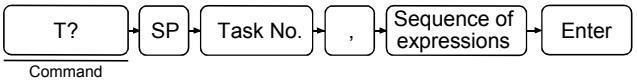
(2) Precautions when specifying a line number
- When specifying the line number, a line number that exists in a program should be entered using the same decimal format as the description format in the program.
  If a line number that does not exist in a program is entered, the execution is resumed from the line with the first number after the specified number.

(3) Reference
- Operation for checking the status of the current BASIC program : TSTATUS command (Section 5.2.1)
- Operation for stopping the execution of the specified BASIC program : TSTOP command (Section 5.2.3)
- Operation for starting the execution from the start line again : TRUN command (Section 5.2.2)

5.2.5 Displaying Values of Specified Variables in the Specified BASIC Program (T? Command)

This operation displays the current value of the specified variable used in the BASIC program in the specified task area.

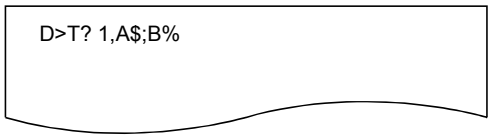
Input format (shortcut for the command none)

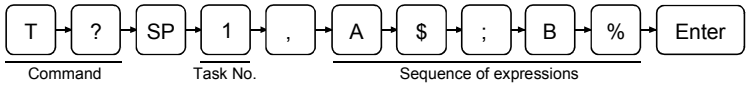


Operation example

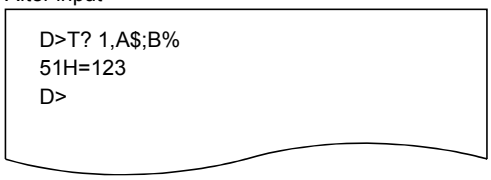
Display the current values of variables A$ and B% used in the BASIC program in the area of task No. 1, whose execution has been stopped.

Before input

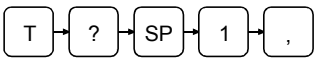




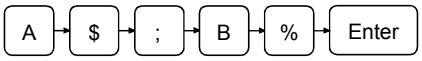
After input



Description



1) Enter the T? command in order to display the values of the variables along with the task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the task area where the target program resides.
In an example figure to the left, the BASIC program in task No. 1 area is specified.



2) Enter the name of the variables you want the values to be displayed.
The variables can be specified as numerical or string expressions in the same way as for the BASIC instruction PRINT.
Moreover, several variables can be displayed by separating the expressions by comma (,) or semicolon (;).
In an example figure to the left, it is specified to display the values of A$ and B%.

```
D>T? 1,A$;B%
51H=123
D>
```

3) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, the values of the specified variables, etc. are displayed.
If the command ends abnormally, "Error" and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown. It indicates that A$ contains "51H=" and B% contains 123.

4) "D>" is displayed in the line following the command execution result.
Enter the next command.

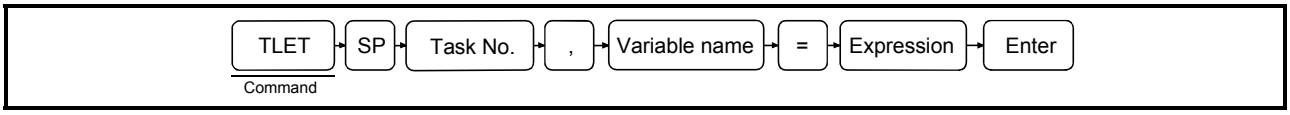
(1) Precautions when entering the command
- If you specify a BASIC program in the DORMANT status, an error occurs.
- The T? command, including the sequence of expressions, must be entered in such a way that the entire command is contained in one line.
  The expression order should furthermore be entered in such a way that the number of characters displayed is 1,024 characters or less.
- It is recommended to place the target BASIC program in the STOP status using the TSTOP command before entering the T? command.

(2) Reference
- Operation for checking the status of the current BASIC program : TSTATUS command (Section 5.2.1)
- Operation for stopping the execution of the specified BASIC program : TSTOP command (Section 5.2.3)
- Operation for resuming (continuing) execution from a line at which execution was stopped : TCONTINUE command (Section 5.2.4)
- Operation for assigning values to specified variables : TLET command (Section 5.2.6)

5.2.6 Assigning Values to Specified Variables in the Specified BASIC Program (TLET command)

This operation assigns values to the specified variables used in the BASIC program in the specified task area.

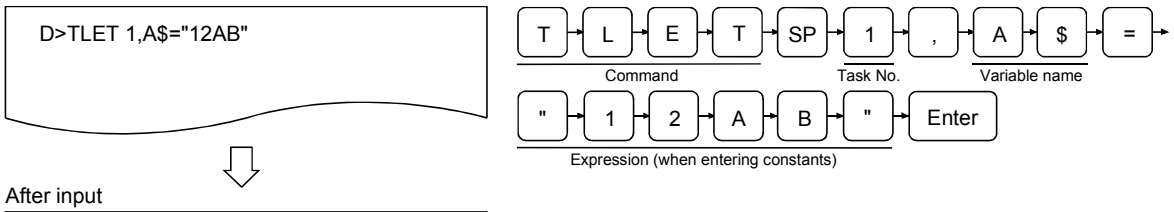
Input format (shortcut for the command TL)



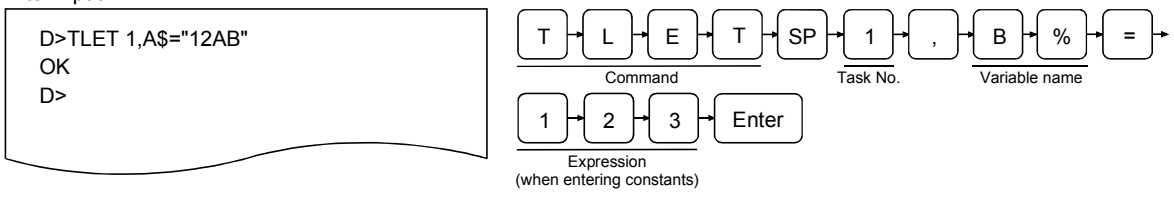
Operation example

Assign values to variables A$ and B% used in the BASIC program area of task No. 1, whose execution has been stopped.

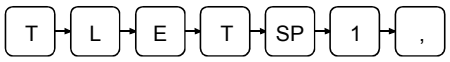
Before input



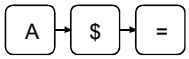
After input



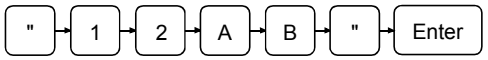
Description



1) Enter the TLET command in order to assign values to variables along with the task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the task area where the target program resides.
In an example figure to the left, the BASIC program in the area of task No. 1 is specified.



2) Enter the name of the variables to which the values are to be assigned.
It is possible to specify names of array variables (e.g., C(0), D$(1%)) in the same way as for the BASIC instruction LET.
In an example figure to the left, it is specified that a value is assigned to character variable A$.




```
D>TLET 1,A$="12AB"
```

3) Enter the value to be assigned.
The values to be assigned can be specified as numerical or string expressions in the same way as for the BASIC instruction LET.
In an example figure to the left, it is specified to assign character constant "12AB" to character variable A$.

```
D>TLET 1,A$="12AB"
TLET OK
D>
```

4) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, "OK" is displayed.
If the command ends abnormally, "Error" message and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.

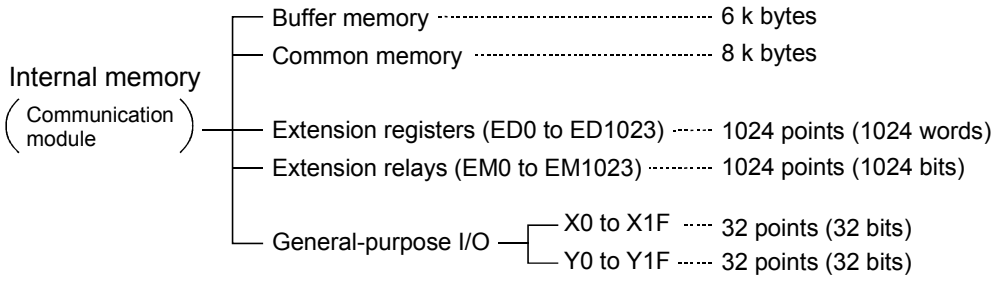
5) "D>" is displayed in the line following the command execution result.
Enter the next command.

(1) Precautions when entering the command
- An error will occur if a BASIC program in the DORMANT status is specified.
- It is recommended to place the target BASIC program in the STOP status using the TSTOP command before entering the TLET command.

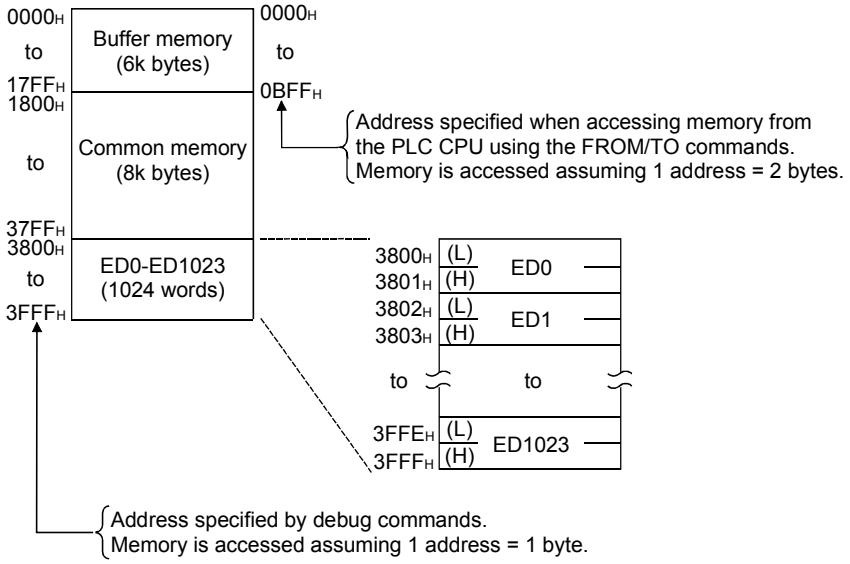
(2) Reference
- Operation for checking the status of the current BASIC program : TSTATUS command (Section 5.2.1)
- Operation for stopping the execution of the specified BASIC program : TSTOP command (Section 5.2.3)
- Operation for resuming (continuing) execution from a line at which execution was stopped : TCONTINUE command (Section 5.2.4)
- Operation for checking values of specified variables : T? command (Section 5.2.5)

5.3 Internal Memory Read/Write Operations

This section explains how to use each of the debug commands for controlling memory access and the operating procedure for reading/writing from/to the module's internal memory.



The addresses are specified when accessing internal memory via the commands MREAD and MWRITE, explained in this section. The figure below illustrates the relationship between addresses specified by these commands and internal memory. Access should be made within the address range of each memory area.

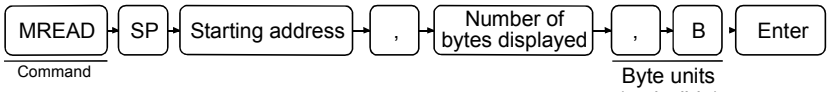


5.3.1 Displaying Values of Buffer Memory/Common Memory/Extension Registers (ED) (MREAD Command)

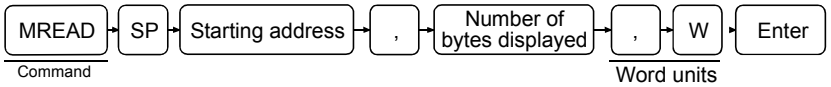
This operation displays the values currently stored in specified memory, which can be either buffer memory, common memory, or extension register (ED).

Input format (shortcut for the command MR)

When displaying the values stored in memory in byte units



When displaying the values stored in memory in word units

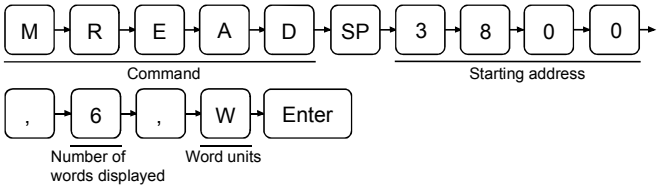


Operation example

Display the values stored in ED0 to ED5 in word units.

Before input

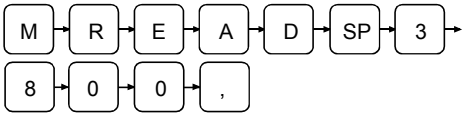
```
D>
```



After input

```
D>MREAD 3800,6,W
3800:0000 0000 0000 0000 0000 0000
D>
```

Description




```
D>MREAD 3800,
```

1) Enter the MREAD command in order to display values stored in device memory along with the starting address of the memory range for which values are to be displayed using a hexadecimal number up to 4 digits (0 to 3FFF). The relationship between device memory and addresses is shown in Section 13.3. In an example figure to the left, address 3800H of ED0 is shown. An even number should be specified for the least significant digit when displaying in word units.
The specified address will not be displayed if the least significant digit is an odd number.

(1) Precautions when entering the number of bytes/words to be displayed

- Enter the number of bytes/words to be displayed in such a way that it satisfies the following condition:
  Address + number of bytes/words displayed – 1 < 3FFFH
  If addresses that exceed 3FFFH are specified, only the values stored in memory up to 3FFFH are displayed.

6 → , → W → Enter

```
D>MREAD 3800,6,W
```

2) Enter the number of bytes/words in the memory range whose values are to be displayed along with the display type.
If B (omissible) is specified as display type, enter the number of bytes in the memory.
If W is specified as display type, enter the number of words in the memory range. In either case, the number of bytes/words should be entered as a hexadecimal number.
When specifying in number of bytes:

1_H < (number of bytes) < 4000_H

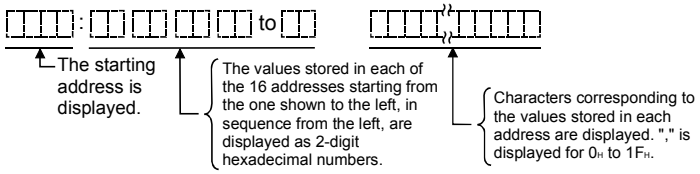
When specifying in number of words:

1_H < (number of words) < 2000_H

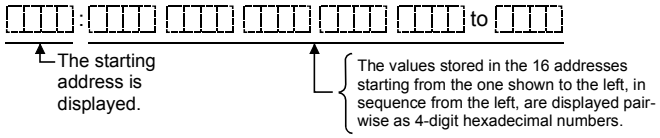
In an example figure to the left, 6 words are specified in word units.

```
D>MREAD 3800,6,W
3800:0000 0000 0000 0000 0000 0000
D>
```

3) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, the values stored in the specified memory range are displayed in the specified units.
In case of byte units, the values stored in up to 16 addresses, □□□0 to □□□F, are displayed in one line as shown below. Note that if the least significant digit of the specified address is different from 0 (n) the values displayed for □□□0 to □□□n-1 become blank.



In case of word units, the values stored in 16 addresses are displayed in one line as shown below.



If the command ends abnormally, "Error" message and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.

4) "D>" is displayed in the line following the command execution result.
Enter the next command.

(2) Operation when it is specified to display 17 or more lines

- The screen can display a maximum of 16 lines of values stored in memory. Press any key other than [ESC] to display values from the subsequent address, if it is specified to display 17 or more lines of values.
  Press [ESC] if you want to stop displaying values stored in memory.

(3) Reference

- Operation for writing values to specified memory : MWRITE command (Section 5.3.2)
- Operation for checking word information in extension registers ED : W@ command (Section 5.3.5)
- Operation for writing word information to extension registers ED : W@ command (Section 5.3.6)

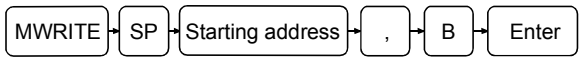
5 - 19

5.3.2 Writing Values to Buffer Memory/Common Memory/Extension Registers (ED) (MWRITE Command)

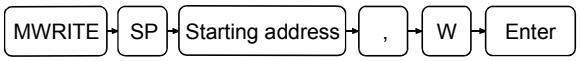
This operation writes values to the specified memory, which can be either buffer memory, common memory, or extension memory (ED).

Input format (shortcut for the command MW)

When writing values to memory in byte units (one value per one address)



When writing values to memory in word units (one value per two addresses)



Operation example

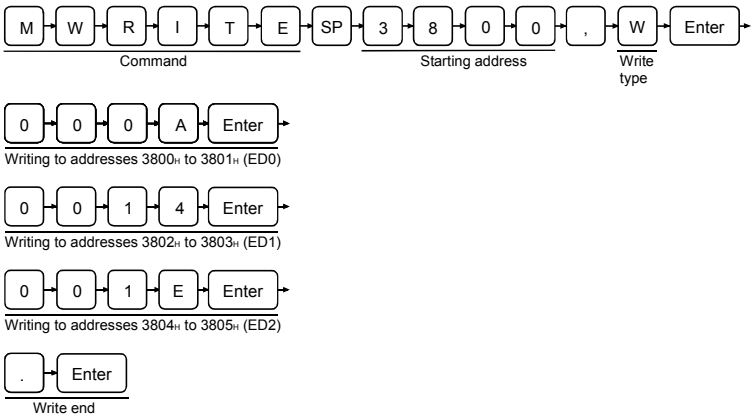
Write the values ($0A_H$, 14_H, $1E_H$) to ED0 to ED2 in word units.

Before input

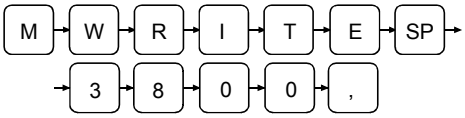
```
D>
```

After input

```
D>MWRITE 3800,W
3800:0000 000A
3801:0000 0014
3802:0000 001E
3803:0000 .
D>
```

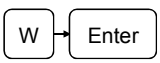


Description




```
D>MWRITE 3800,
```

1) Enter the MWRITE command in order to write values to device memory along with the memory address where the writing should start using a hexadecimal number up to 4 digits (0 to 3FFF).
The relationship between device memory and addresses is shown in Section 5.3.
In an example figure to the left, address 3800_H of ED0 is specified.




```
D>MWRITE 3800,W
```

2) Enter type of values to be written.
Enter B (omissible) to write the values in byte units.
Enter W to write the values in word units.
In an example figure to the left, it is specified to write in word units.

```
D>MWRITE 3800,W
3800:0000
```

0 → 0 → 0 → A → Enter

```
D>MWRITE 3800,W
3800:0000 000A
```

0 → 0 → 1 → 4 → Enter

```
D>MWRITE 3800,W
3800:0000 000A
3801:0000 0014
```

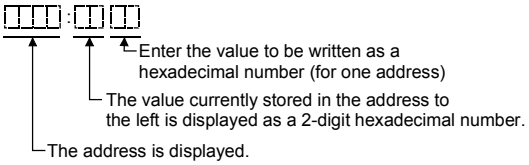
0 → 0 → 1 → E → Enter

```
D>MWRITE 3800,W
3800:0000 000A
3801:0000 0014
3802:0000 001E
```

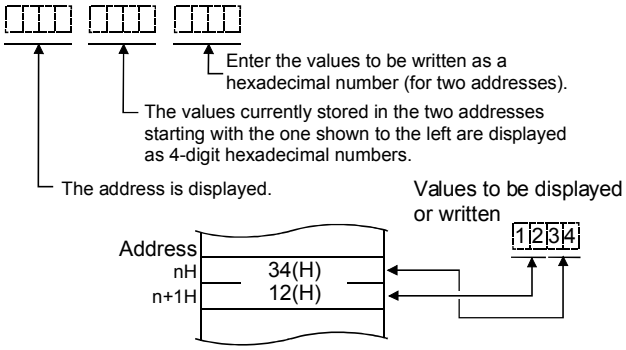
. → Enter

```
D>MWRITE 3800,W
3800:0000 000A
3801:0000 0014
3802:0000 001E
3803:0000.
D>
```

3) The screen displays the result of the command execution in the succeeding line.
If the command ends normally, the specified address and currently stored value are displayed with the specified units in hexadecimal numbers.
In case of byte units, the display shows as follows; enter the value to be written using a hexadecimal number with 2 digits or less (only valid digits can be entered).



In case of word units, the display shows as follows; enter the value to be written using a hexadecimal number with 4 digits or less (only valid digits can be entered).



The following keys are used in the write operation:

0 to 9, A to F : Used to enter the values to be written.
. : Used to write the entered values to device memory.
\ : Used when backing up the target address.
Enter : Used when executing entries made using the keys above.
Used when the value currently stored in memory is not to be overwritten.

If the command ends abnormally, "Error" message and an error code are displayed.
In an example figure to the left, a display where values are written to memory addresses 3800H to 3805H (ED0 to ED2) in word units successfully is shown.

4) When writing to device memory is completed, "D>" is displayed.
Enter the next command.

(1) Operation when exceeding memory range of target device
- The processing of the MWRITE command is automatically terminated if the address of device memory to which a value is written exceeds 3FFF$_H$.

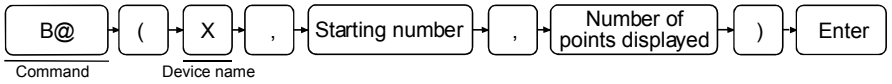
(2) Reference
- Operation for checking values of specified memory : MWRITE command (Section 5.3.1)
- Operation for checking word information of extension registers ED : W@ command (Section 5.3.5)
- Operation for writing word information to extension registers ED : W@ command (Section 5.3.6)

5.3.3 Displaying Bit Information of General-Purpose Inputs (X)/General-Purpose Outputs (Y)/Extension Relays (EM) (B@ Command)

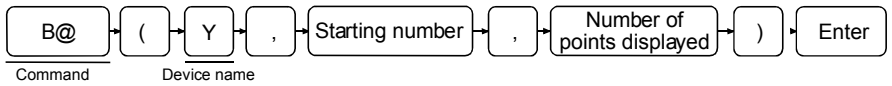
This operation displays bit information of general-purpose inputs (X)/outputs (Y) used by the PLC CPU or extension relays (EM) used by BASIC programs for data communication.

Input format (shortcut for the command none)

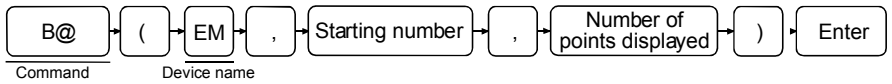
When displaying bit information of general-purpose inputs (X)



When displaying bit information of general-purpose outputs (Y)



When displaying bit information of extension relays (EM)

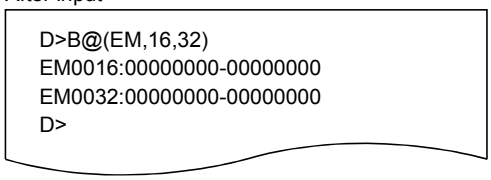


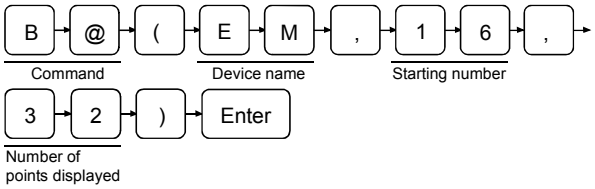
Operation example

Display bit information of EM16 to EM47.

Before input

After input

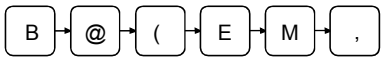




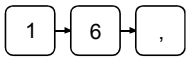
(1) About general-purpose input/output devices between the PLC CPU and the communication module

- General-purpose input/output devices handle bit information communicated between the sequence programs on the PLC CPU side and the BASIC programs on the communication module side, as well as bit information controlled by each OS.

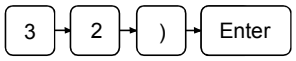
Description

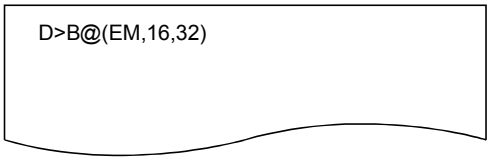


1) Enter the B@ command to display bit information and the type of the target device.
Enter the device type using the following characters.
X : When specifying general-purpose inputs (PLC CPU ← Communication module)
Y : When specifying general-purpose outputs (PLC CPU → Communication module)
EM : When specifying extension relays
In an example figure to the left, extension relays are specified.



2) Enter the starting number of the target device range for which the bit information is to be displayed.
Enter a hexadecimal number with 2 digits or less if X or Y have been specified, or a decimal number with 4 digits or less if EM has been specified.
X/Y: 0 to 1F EM: 0 to 1023
In an example figure to the left, EM16 is specified.

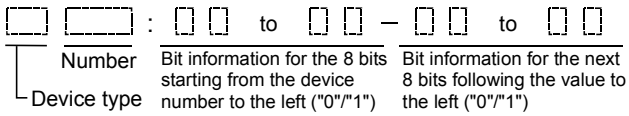




3) Enter the number of points displayed (number of bits) of the target device range for which bit information is to be displayed using a decimal or hexadecimal number.
X/Y : 1 (1_H) ≤ "number of points displayed" 32 ≤ (20_H)
EM : 1 (1_H) ≤ "number of points displayed" 1024 ≤ (400_H)
In an example figure to the left, 32 points (32 bits) are specified.

```
D>B@(EM,16,32)
EM0016:00000000-00000000
EM0032:00000000-00000000
```

4) The result of the command execution is displayed in the following line.
If the command ends normally, the bit information of the specified device range is displayed.
For X/Y, 16 points of bit information corresponding to device numbers from □□□0 to □□□F are displayed as "0" (off) or "1" (on) in one line using the format shown below.
For EM, 16 points of bit information corresponding to device numbers from [an integral multiple of 16] to [the next integral multiple of 16 minus 1] are displayed as "0" (off) or "1" (on) in one line using the format shown below.
Note that if the least significant digit of the specified number for X/Y is different from 0 (n), or the specified number for EM is different from an integral multiple of 16 (n), the bit information corresponding to 0 or an integral multiple of 16 up to the specified number minus 1 becomes blank.



If the command ends abnormally, "Error" message and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.

5) "D>" is displayed in the line following the command execution result.
Enter the next command.

(1) Precautions when entering the number of points displayed
- Enter the number of points displayed in such a way that it satisfies the following conditions:
  X/Y : device number + number of points displayed – 1 ≤ 1F (H)
  EM : device number + number of points displayed – 1 ≤ 1023
- If a range exceeding the maximum number of the device type is specified, the bit information up to the maximum number of the device is displayed.

(2) Operation when it is specified to display 17 or more lines
- The screen can display a maximum of 16 lines of bit information. Press any key other than [ESC] to display bit information of the following device numbers, if it is specified to display 17 or more lines of values.
- Press [ESC] if you want to stop displaying bit information.

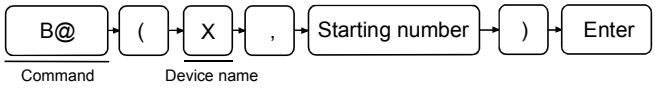
(3) Reference
- Operation for writing bit information to extension relays EM : B@ command (Section 5.3.4)

5.3.4 Writing Bit Information to General-Purpose Inputs (X)/Extension Relays (EM) (B@ Command)

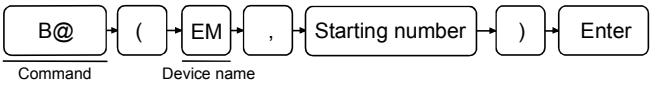
This operation writes bit information to general-purpose inputs (X) used by the PLC CPU or extension relays (EM) used by BASIC programs for data communication.

Input format (shortcut for the command none)

When writing bit information to general-purpose inputs (X)



When writing bit information to extension relays (EM)



Operation example

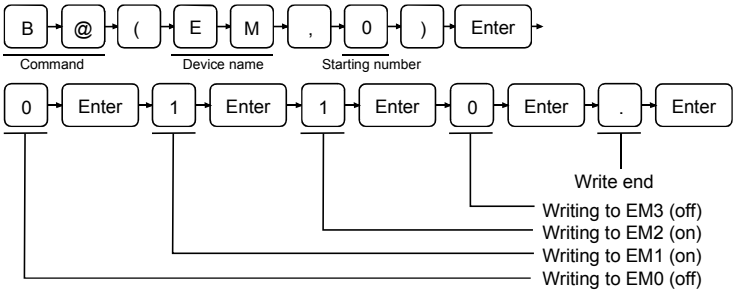
Write the bit information (0, 1, 1, 0) to EM0 to EM3.

Before input

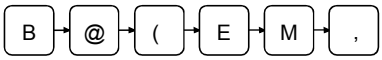
```
D>
```

After input

```
D>B@(EM,0)
EM0000:0 0
EM0001:0 1
EM0002:0 1
EM0003:0 0
EM0004:0 .
D>
```



Description

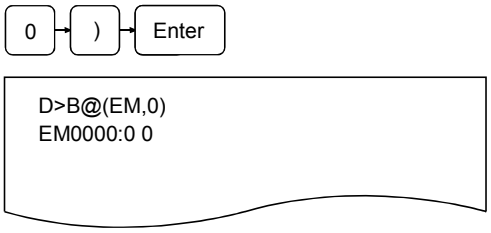



```
D>B@(EM,
```

1) Enter the B@ command to write bit information along with the type of the target device.
   Enter the device type using the following characters:
   X : Name when specifying general-purpose inputs (PLC CPU ← Communication module)
   EM : Name when specifying extension relays
   In an example figure to the left, extension relay EM is specified.

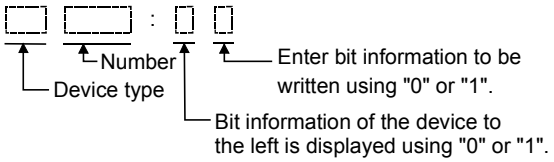
(1) About general-purpose input/output devices between the PLC CPU and the communication module
- General-purpose input/output devices handle bit information communicated between sequence programs on the PLC CPU side and BASIC programs on the communication module side, as well as bit information controlled by each OS.

(2) Precautions when using the command
- Do not write bit information to general-purpose inputs X0B to X0F. (In order to operate the communication module normally)



2) Enter the number of the target device from which the writing of bit information is started.
Enter a hexadecimal number with 2 digits or less if X has been specified, or a decimal number with 4 digits or less if EM has been specified.
X : 0 to 1F EM: 0 to 1023
In an example figure to the left, EM0 is specified.

3) The result of the command execution is displayed in the succeeding line.
If the command ends normally, the specified device number and the bit information are displayed as shown below using "0" or "1."
Enter the bit information to be written at the cursor position using “0” or ”1.”



The following keys are used in the bit information write operation.
[0] : Used to turn the target bit off.
[1] : Used to turn the target bit on.
[\] : Used when backing up the target device number.
[.] : Used to end the bit information write operation.
If the command ends abnormally, "Error" message and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.

4) "D>" is displayed in the line following the command execution result.
Enter the next command.

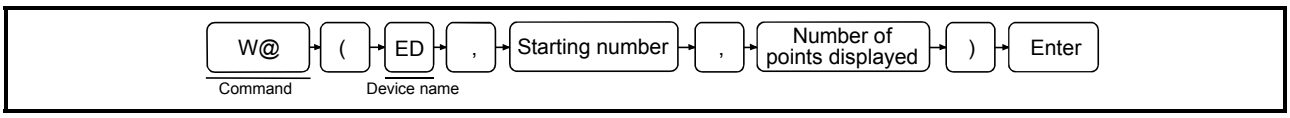
(1) Processing when exceeding target device memory range
- The processing of the B@ command is automatically terminated if the device number to which bit information is written exceeds the maximum number of the target device type.

(2) Reference
- Operation for checking bit information of extension relays EM : B@ command (Section 5.3.3)

5.3.5 Displaying Word Information of Extension Registers (ED) (W@ Command)

This operation displays word information (values) of extension registers (ED) used by BASIC programs for data communication.

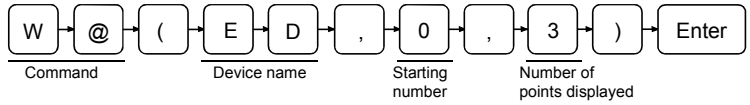
Input format (shortcut for the command none)



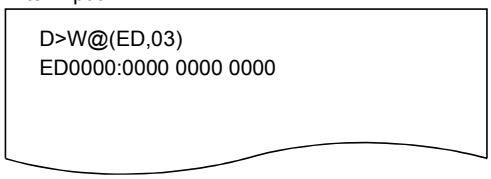
Operation example

Display word information (values) of ED0 to ED2.

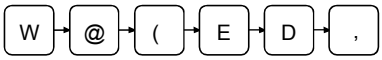
Before input



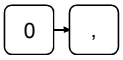
After input

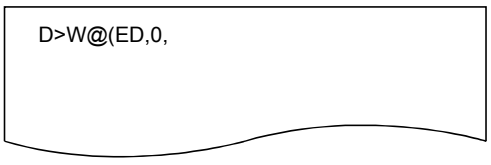


Description



1) Enter the W@ command to display word information of extension registers and the type of internal device, ED.





2) Enter the starting number of the ED range for which the word information is to be displayed using a decimal number up to 4 digits (0 to 1023).
In an example figure to the left, ED0 is specified.

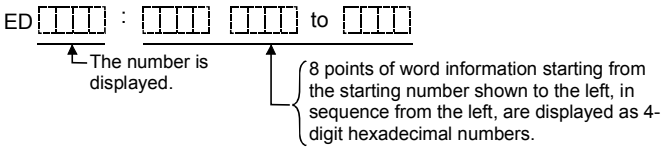
5 - 28

3 →) → Enter

3) Enter the number of points displayed (number of words) of the range for which word information is to be displayed in decimal number.
ED : 0 ≤ number of points displayed ≤ 1023
In an example figure to the left, 3 points (3 words) are specified.

```
D>W@(ED,0,3)
ED0000:0000 0000 0000
```

4) The result of the command execution is displayed.
If the command ends normally, the bit information of the specified range is displayed.
8 points of word information corresponding to device numbers from [an integral multiple of 8] to [the next integral multiple of 8 minus 1] are displayed as 4-digit numbers in one line using the format shown below.
Note that if the specified number is different from an integral multiple of 8 (n), the word information corresponding to the integral multiple of 8 up to the specified number minus 1 becomes blank.



If the command ends abnormally, "Error" message and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.

5) "D>" is displayed in the line following the command execution result.
Enter the next command.

(1) Precautions when entering number of points displayed
- Enter the number of points displayed in such a way that it satisfies the following conditions:
  ED : Starting number + number of points displayed – 1 ≤ 1023
- If a number exceeding ED 1023 is specified, only word information up to ED 1023 is displayed.

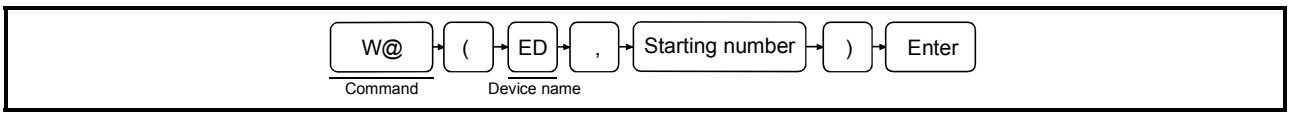
(2) Operation when it is specified to display 17 or more lines
- The screen can display a maximum of 16 lines of word information. Press any key other than [ESC] to display word information of the following device number, if it is specified to display 17 or more lines of values.
- Press [ESC] if you want to stop displaying bit information.

(3) Reference
- Operation for checking values of specified memory (address specification) : MREAD command (Section 5.3.1)
- Operation for writing values to specified memory (address specification) : MWRITE command (Section 5.3.2)
- Operation for writing word information to extension registers ED : W@ command (Section 5.3.6)

5.3.6 Writing Word Information to Extension Registers (ED) (W@ Command)

This operation writes word information (values) to extension registers (ED) used by BASIC programs for data communication.

Input format (shortcut for the command none)



Operation example

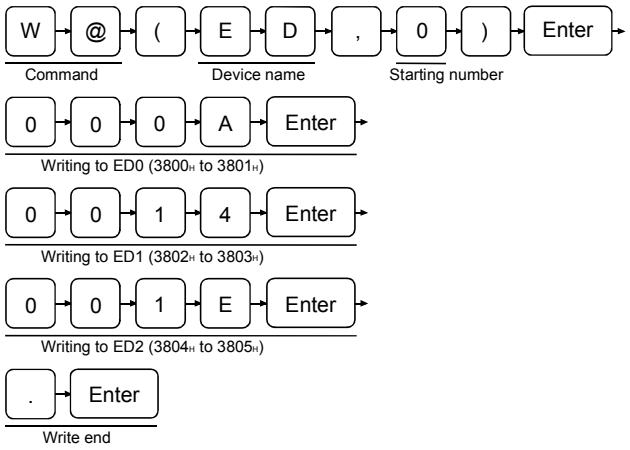
Write the word information ($0A_H$, 14_H, $1E_H$) to ED0 to ED2.

Before input

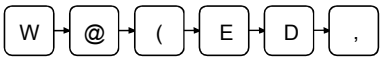
```
D>
```

After input

```
D>W@(ED,0)
EM0000:0000 000A
EM0001:0000 0014
EM0002:0000 001E
EM0002:0000 .
D>
```



Description




```
D>W@(ED,
```

1) Enter the W@ command to write word information to ED and the type of internal device, ED.

(1) About writing to ED internal devices

- Writing to ED internal devices can also be performed by the debug command MWRITE.

0 →) → Enter

```
D>W@(ED,0)
```

```
D>W@(ED,0)
ED0000:0000
```

0 → 0 → O → A → Enter

```
D>W@(ED,0)
ED0000:0000 000A
```

0 → 0 → 1 → 4 → Enter

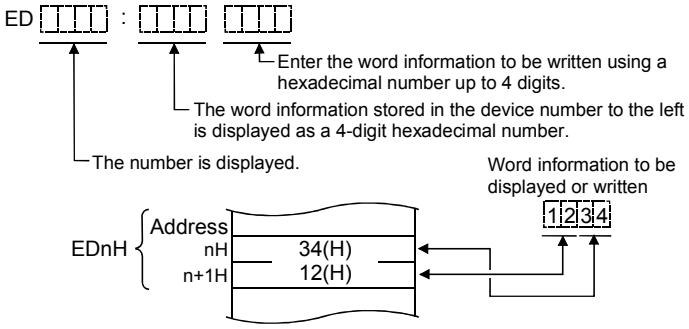
```
D>W@(ED,0)
ED0000:0000 000A
ED0001:0000 0014
```

0 → 0 → 1 → E → Enter

```
D>W@(ED,0)
ED0000:0000 000A
ED0001:0000 0014
ED0002:0000 001E
```

2) Enter the starting number of the ED range to which the word information is written using a decimal number up to 4 digits (0 to 1023).
In an example figure to the left, ED0 is specified.

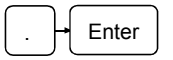
3) The result of the command execution is displayed in the succeeding line.
If the command ends normally, the specified device number and the word information are displayed as shown below.
Enter the word information to be written using a hexadecimal number up to 4 digits (only valid digits can be entered).



The following keys are used in the word information write operation.

[0] to [9], [A] to [F] : Used when entering word information to be written.
[\] : Used when backing up the ED target device number.
[.] : Used to end the word information write operation to ED.
[Enter] : Used when executing entries made using the keys above.

If the command ends abnormally, "Error" message and an error code are displayed.
In an example figure to the left, a display where word information is successfully written to ED0 to ED2 (address 3800H to 3805H) is shown.




```
D>W@(ED,0)
ED0000:0000 000A
ED0001:0000 0014
ED0002:0000 001E
ED0003:0000 .
D>
```

4) "D>" is displayed in the line following the command execution result.
Enter the next command.

(1) Processing when exceeding ED 1023
- The processing of the W@ command is automatically terminated if the ED internal device number to which word information is written exceeds 1023.

(2) Reference
- Operation for checking values of specified memory (address specification) : MREAD command (Section 5.3.1)
- Operation for writing values to specified memory (address specification) : MWRITE command (Section 5.3.2)
- Operation for checking word information of extension registers ED : W@ command (Section 5.3.5)

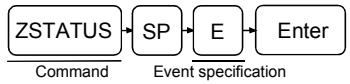
5.4 Operations for Checking the Usage of Events/Message Ports/Resource Numbers

This section explains how to use the debug command ZSTATUS and the operating procedure when checking the current usage of events, message ports, or resource numbers shared among BASIC programs. ZSTATUS is generally used for checking OS information.

5.4.1 Displaying the Event Enable/Disable Declaration Status (ZSTATUS Command)

This operation displays the BASIC program's current even enable/disable declaration status for events shared among BASIC programs for each event number.

Input format (shortcut for the command ZS)

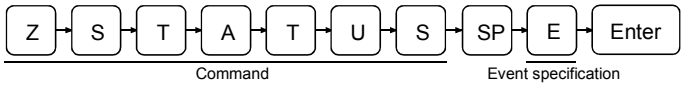


Operation example

Display the current even enable/disable declaration status by each event number.

Before input

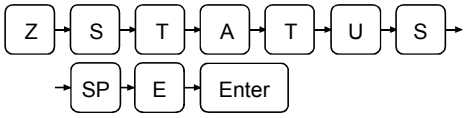
```
D>
```



After input

```
D>ZSTATUS E
No. EN/DI  ON/OFF  No.  EN/DI  ON/OFF
  0 DISABLE  OFF     1  DISABLE  OFF
  3 DISABLE  OFF     4  DISABLE  OFF
  6 DISABLE  OFF     7  DISABLE  OFF
  9 DISABLE  OFF    10  DISABLE  OFF
 12 DISABLE  OFF    13  DISABLE  OFF
 15 DISABLE  OFF
```

Description



1) Enter the ZSTATUS to display information managed by the OS and "E" to specify events as the information type to be displayed.

```
D>ZSTATUS E
```

(1) About events

Events are used in BASIC programs according to the following instructions.
See the AD51H-BASIC Programming Manual (Commands) for more information.

- Event definition : DEF ZEVENT instruction
- Event enable/disable declaration : ZEVENT instruction
- Event occurrence : ZSIGNAL instruction
- Wait for event occurrence : ZWAIT EVENT instruction

```
D>ZSTATUS E
No. EN/DI  ON/OFF  No.  EN/DI  ON/OFF
  0 DISABLE  OFF    1  DISABLE  OFF
  3 DISABLE  OFF    4  DISABLE  OFF
  6 DISABLE  OFF    7  DISABLE  OFF
  9 DISABLE  OFF   10  DISABLE  OFF
 12 DISABLE  OFF   13  DISABLE  OFF
 15 DISABLE  OFF
```

2) The result of the command execution is displayed in the succeeding line.
If the command ends normally, the even enable/disable declaration status corresponding to each of the event numbers from 0 to 63 is displayed from the succeeding line.
The descriptions of the displayed information are as follows:

EN/DI column	ON/OFF column	Meaning
ENABLE	ON	The event corresponding to the number is defined and declared to be enable.
ENABLE	OFF	The event corresponding to the number is defined but not declared to be enable.
DISABLE	ON	The event corresponding to the number is defined and declared to be disable.
DISABLE	OFF	The event corresponding to the number is not defined.

If the command ends abnormally, "Error" message and an error code are displayed in the succeeding line.

In an example figure to the left, a display where the command ends normally is shown.

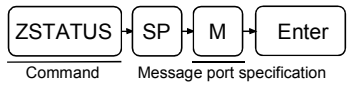
3) "D>" is displayed in the line following the command execution result.
Enter the next command.

5 - 34

5.4.2 Displaying the Status of Transmission to Message Ports (STATUS Command)

This operation displays the transmission status of current messages to message ports shared among BASIC programs for each message board.

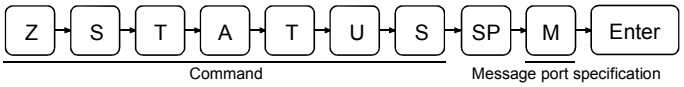
Input format (shortcut for the command ZS)



Operation example

Display transmission status of the current messages.

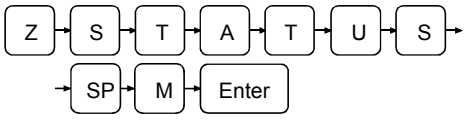
Before input



After input

```
D>ZSTATUS M
Message Port No.   PRI/FIFO Length Count
```

Description




```
D>ZSTATUS M
```

1) Enter the ZSTATUS command to display information managed by the OS and "M" to specify messages transmitted to message ports as the information type to be displayed.

---

(1) About message transmission via message ports

Message transmission/reception between BASIC programs becomes possible by defining message ports within BASIC programs.
See the AD51H-BASIC Programming Manual (Commands) for more information.
All instructions related to this subject begin with "ZMESSAGE."

```
D>ZSTATUS M
Message Port No.   PRI/FIFO Length Count
```

2) The result of the command execution is displayed.
If the command ends normally, the transmission status of messages to each message port is displayed from the next line (information on transmission messages that have not been received yet).
The descriptions of the displayed information are as follows:

Message Port No. column	:	Number of message ports defined by the user
PRI/FIFO column	:	Shows the type of the corresponding message port PRI : No "FIFO" specification in the port definition FIFO : "FIFO" is specified in the port definition
Length column	:	The "byte length" specified when defining the message port
Count column	:	Number of messages transmitted to the corresponding message port but not received yet

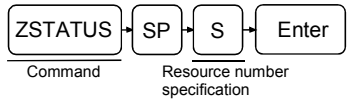
If the command ends abnormally, "Error" message and an error code are displayed.
In an example figure to the left, a display where the command ends normally is shown.

3) "D>" is displayed in the line following the command execution result.
Enter the next command.

5.4.3 Displaying the Reserved/Released Status of Resource Numbers for Exclusive Access Control (ZSTATUS Command)

This operation displays the reserved/released status of resource numbers when access to shared resources such as memory and peripheral devices is limited to one BASIC program at a time by each resource number.

Input format (shortcut of the command ZS)

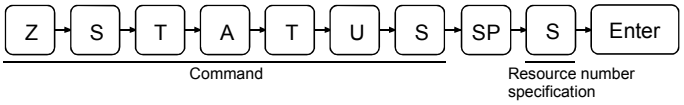


Operation example

Display reserved/released status of the current resource numbers.

Before input

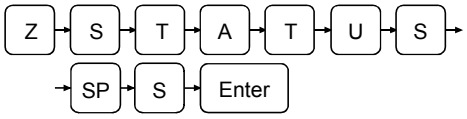
```
D>
```



After input

```
D>ZSTATUS S
Semaphore     PESERVE/    Basic
   No.        RELEASE      No.
```

Description




```
D>ZSTATUS  S
```

1) Enter the ZSTATUS command to display information managed by the OS and "S" to specify resource numbers as the information type to be displayed.

---

(1) About exclusive access control of resources by reserving/freeing resource numbers

Exclusive access control of resources can be achieved using the following instructions in BASIC programs when multiple BASIC programs are being executed at the same time.
See the AD51H-BASIC Programming Manual (Commands) for more information.

- Reserving a resource number : ZRESERVE instruction
- Releasing a resource number : ZRELEASE instruction

5 - 37

```
D>ZSTATUS S
Semaphore     PESERVE/    Basic
   No.        RELEASE      No.
```

2) The result of the command execution is displayed.
If the command ends normally, the reserved/released status corresponding to each of the resource numbers from 0 to 31 is displayed from the succeeding line.
The descriptions of the displayed information are as follows:

Semaphore No. column	:	Resource number
RESERVE/RELEASE column	:	Shows the reserved/released status of the corresponding resource number. RESERVE : Shows that the status is reserved. RELEASE : Shows that the status is released.
Basic No. column	:	Corresponding resource number.

If the command ends abnormally, "Error" message and an error code are displayed in the succeeding line.
In an example figure to the left, a display where the command ends normally is shown.

3) "D>" is displayed in the line following the command execution result.
Enter the next command.

5 - 38

5.5 Operations for Changing the Mode of the Communication Module

This section explains how to use each of the debug commands for controlling modes and the operating procedure when changing the mode of the communication module.

5.5.1 Changing the Communication Mode to Edit Mode (2) (START Command)

This operation changes the mode of the communication module to edit mode (2) during the execution of multiple BASIC programs so that the program in the specified task area or other programs can be edited (for instance, to add or modify a code).

Input format (shortcut for the command ST)

START → SP → Task No. → Enter

Command

Operation example

Change the mode of the communication module to edit mode (2) in order to modify the BASIC program in the area of task No. 1, whose execution has been stopped.

Before input

```
D>
```

S → T → A → R → T → SP → 1 → Enter

Command: S T A R T　　Task No.: 1

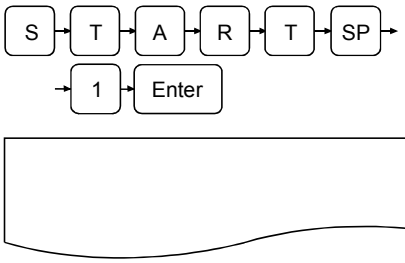
After input

```
D>START 1
```

```
OK
```

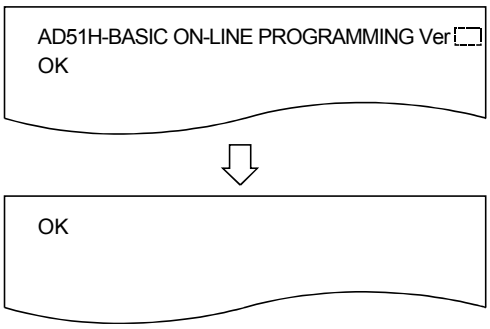
If the interpreter has not been started up, the following message is displayed immediately before OK.
"AD51H-BASIC ON-LINE PROGRAMMING Ver [------] "

Description

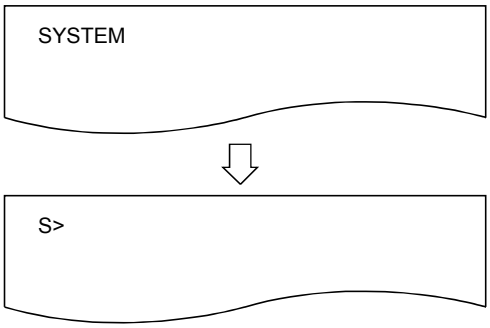


1) Enter the START command to change the mode of the communication module to edit mode (2) and the task number (task No. 1 to 8 can be specified for AD51H-S3, 1 or 2 for A1SD51S/QD51 (-R24)) of the task area where the BASIC program to be edited resides. The entry of the task number may be omitted.
If the task number is omitted, it is assumed that the following task number is specified.
- It is assumed that "1" is specified when the START command is entered for the first time.
- It is assumed that the same number as specified by the last START command is specified if the START command is already used.

In an example figure to the left, the BASIC program is specified to be edited in the area of task No. 1.



2) The result of the command execution is displayed.
If the command ends normally, the display in the figure shows, and then the user can edit the BASIC program from this point.
The editing operations of BASIC programs are explained in the programming manual.
If the command ends abnormally, an error message or similar is displayed.
The upper figure to the left shows the display when the interpreter has not been started up.
The lower figure to the left shows the display when the interpreter has already been started up.



3) Perform one of the following operations when the editing of the BASIC program is finished in edit mode (2) and the mode of the communication module should be returned to multitask debug mode.
[Execute the BASIC instruction SYSTEM]
- The BASIC program stops executing.
- All open files and communication lines are closed.

[Press Ctrl + D]
- The BASIC program stops executing.
- Open files and communication lines are kept open.
- If the BASIC program whose execution was stopped did not require modification, its execution can be resumed with the debug command TCONTINUE. Furthermore, the execution can be resumed using the BASIC instruction CONTINUE when the mode of the communication module is changed to edit mode (2) the next time.

(1) Precautions when using the START command
- The execution should be stopped using the TSTOP command if a BASIC program is being executed in the task area where the editing is going to take place.

(2) Operation of other BASIC programs when the START command is executed
- If multiple BASIC programs are being executed, and program editing is started in any task area with the START command, the BASIC programs in the other task areas will be continuously executed.

(3) Required processing when it is necessary to change the task size of the specified task area
- Change the mode of the communication module to edit mode (1) using the following method.
  1) Return the communication module to multitask debug mode using the BASIC instruction SYSTEM.
  2) Stop the execution of the BASIC programs in each task area with the STOP command so that they will not interfere with the system control.
  3) Change the mode of the communication module to system mode with the GO command.
  4) End the operation of the interpreter in each task area with the system command "TKILL. "
  5) Change the mode of the communication module to edit mode (1) with the system command START. When the START command is entered, the task size can be changed and the BASIC program can be edited.

  See Section 2.2 for the mode change diagram, for how to change the mode of the communication module.

(1) Reference
- Operation for changing the mode of the communication module: GO command (Section 5.5.2)

5.5.2 Changing the Mode of the Communication Module to System mode/Run Mode/ Multitask Debug Mode (GO Command)

This operation changes the mode of the communication module from multitask debug mode to system mode/run mode (2) or returns it to multitask debug mode again.
By changing to system mode, it becomes possible to edit/debug each BASIC program residing in each task area by entering the system commands to the console (see Chapter 4).

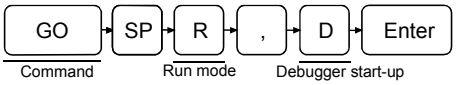
1) When changed to run mode (2), each BASIC program starts its execution according to the multitask settings.
By changing back to multitask debug mode, the debugger restarts and each BASIC program starts being executed according to the multitask settings.
The following table shows the relationship between the mode and debugging start specification when the GO command is entered, and the statuses of the console and the debugger terminal after the GO command has been executed.

Mode setting	Debugging start-up specification Yes/No	Console status	Status of debugger terminal	Remarks
R (Run mode)	Yes (To multitask debug mode)	The contents of the display are deleted. The console display changes to the one used for BASIC programs.	The debugger initiates, the contents of the display are deleted, and "D>" is displayed. It becomes possible to enter debug commands.	Each BASIC program is reloaded to the corresponding task area according to the multitask settings, and is executed.
	No (To run mode)		The contents of the display are kept as is. The terminal becomes a general-purpose port for BASIC programs.	
P (System mode within programming mode)	Cannot be specified.	The contents of the display are deleted and "S>" is displayed. It becomes possible to enter system commands.		BASIC programs in each task area stop being executed.

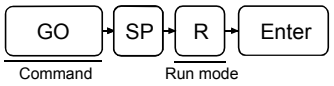
(1) Status of each BASIC program by execution of the GO command
- When Run mode has been specified, BASIC program execution is started.
  This behavior is the same with the case where the communication module is started up with Run or Multitask debug mode setting.
- If system mode is specified, the BASIC programs in task area stop being executed. (BASIC programs other than the ones in the DORMANT status go into the STOP status.)
  The memory status of each task area on the main memory does not change, so the BASIC programs in each task area remain as they are.
  It becomes possible to change the mode of the communication module from system mode to edit mode (1) in order to edit/debug the BASIC programs in each task area.

Input format (shortcut for the command none)

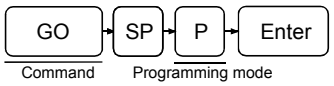
When changing the mode of the communication module back to multitask debug mode



When changing the mode of the communication module to run mode



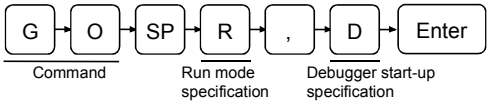
When changing the mode of the communication module to system mode
(one of the programming modes)



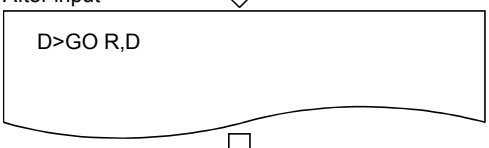
Operation example

Returns the communication module to multitask debug mode.

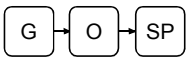
Before input



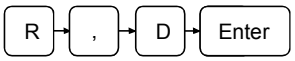
After input



Description



1) Enter the GO command to change mode of the communication module.



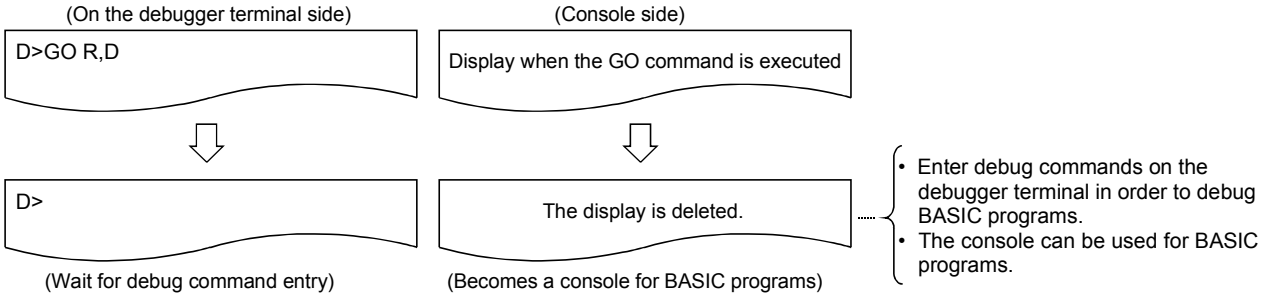
2) Enter a mode.
Enter R to change the mode of the communication module to run mode or multitask debug mode.
Enter P to change the mode of the communication module to system mode.
Enter R followed by D to change the mode of the communication module back to multitask debug mode.
In an example figure to the left, the mode of the communication module is changed back to multitask debug mode.

(2) Precautions when changing mode
It is recommended to stop each BASIC program in advance (see the TSTOP command) so that it does not interfere with the system control when changing the mode of the communication module from multitask debug mode.

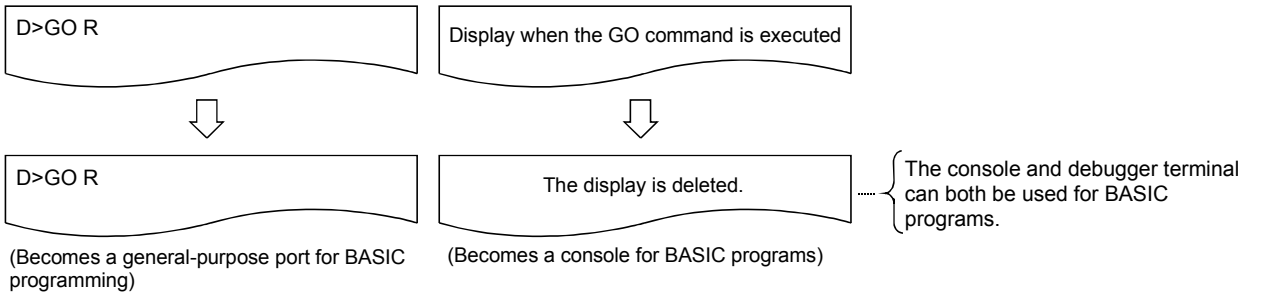
3) The result of the command execution is displayed.
If the command ends normally, the display varies depending on the specification as shown below.
If the command ends abnormally, an error message or similar is displayed.

The following examples illustrate what is displayed when the command ends normally.

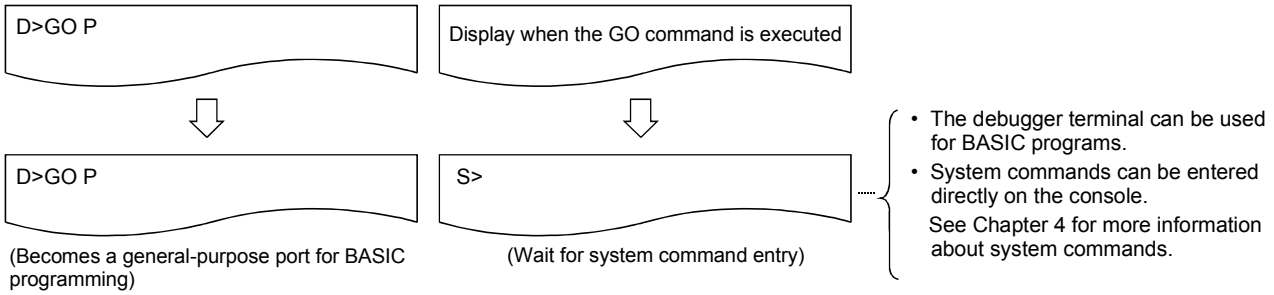
1) When the mode is changed back to multitask debug mode



2) When the mode is changed to run mode



3) When changed to system mode



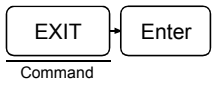
(2) About changing the mode of the communication module
- See Section 2.2 for the mode change diagram of the communication module.

(3) Reference
- Operation for displaying the main menu screen on the debugger terminal : EXIT command (Section 5.6)

5.6 Operation for Displaying the Main Menu Screen on the Debugger (EXIT Command)

This section explains how to use the EXIT command to display the main menu screen of the AD51H-BASIC package on the debugger.

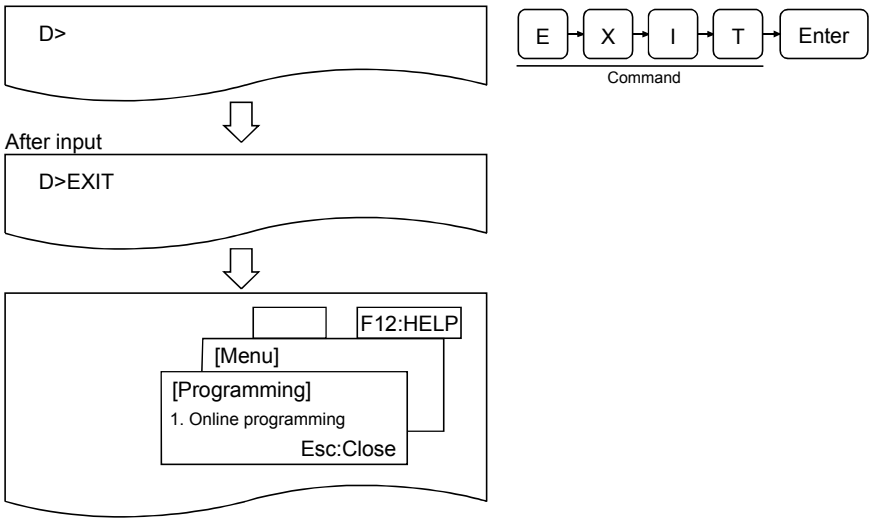
Input format (shortcut for the command E)



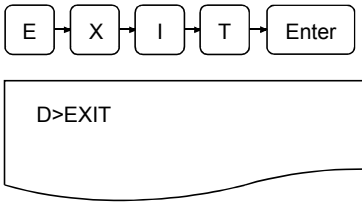
Operation example

The main menu screen is displayed in the debugger.

Before input



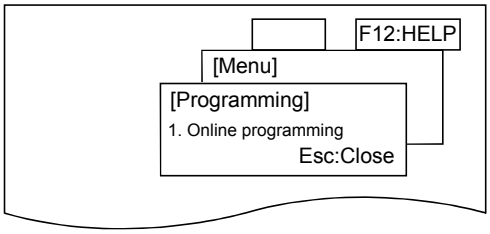
Description



1) Enter the EXIT command to display the main menu screen.

(1) Status of each BASIC program by the EXIT command execution
- The BASIC programs in each task area are continuously executed as they are even when the EXIT command is executed.

(2) Precautions when entering the command
- It is recommended to stop each BASIC program with the TSTOP command in advance before entering the EXIT command so that the execution does not interfere with the system control when displaying the main menu in order to edit the BASIC programs in each BASIC task area, etc.



2) The result of the command execution is displayed; the user can perform the corresponding operations from this point.

If the command ends normally, the main menu screen is displayed on the debugger; select a displayed item to perform the corresponding operation. In an example figure to the left, a display where the command ends normally is shown.
See the AD51H-BASIC Type SW1IVD-AD51HP-E Operating Manual for operations from the main menu screen.

If the command ends abnormally, an error message or similar is displayed.

(2) About changing the mode of the communication module
- See Section 2.2 for the mode change diagram of the communication module.

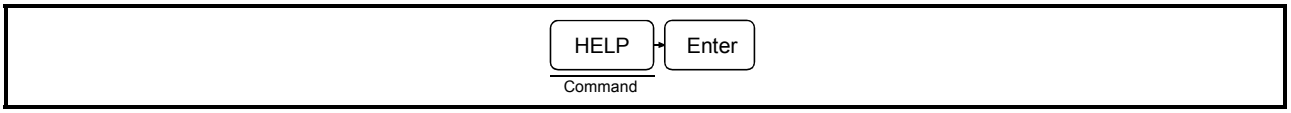
(3) Reference
- Operation for stopping the execution of the specified BASIC program : TSTOP command (Section 5.2.3)
- Operation for changing the mode of the communication module : GO command (Section 5.5.2)

5 - 46

5.7 Operation for Checking the Input Formats of the Debug Commands (HELP Command)

This section explains how to use the command HELP to display the input format, etc. of each of the commands on the debugger in order to check the input format of the debug commands.

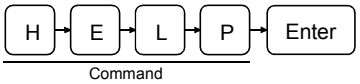
Input format (shortcut for the command H)



Operation example

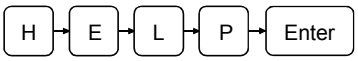
Display input format, etc. of each debug command.

Before input



After input

Description



1) Enter the HELP command to display input format, etc. of each debug command.

2) The screen displays the result of the command execution. If the command ends normally, functions/input formats of each debug command are displayed from the succeeding line.
(Example)

(1)	TSTATUS	Task Status Info.	TS{task No.}
(a)	(b)	(c)	(d)

(a) Number for description
(b) Command
(c) Function of command
(d) Description of input format (shortcut for command)

If the command ends abnormally, an error massage or similar is displayed in the succeeding line.

3) "D>" is displayed in the line following the command execution result.
Enter the next command.

5 - 47

(1) About description of the command input format

- A one column space immediately after a command indicates that the SP key (space) should be entered.
  Parentheses ("("and")") indicate that symbols should be entered as they are.
  Brackets ("{"and"}") are symbols that indicate separation of command arguments; it is not necessary to enter brackets.
  Square brackets ("["and"]") are symbols that indicate that the arguments inside them are omissible; it is not necessary to enter square brackets.

6 CREATING BASIC PROGRAMS WITH A GENERAL-PURPOSE EDITOR

This chapter explains how to create BASIC programs using a general-purpose editor.

Please read this chapter and understand the restrictions, etc. before you start using a general-purpose editor.

6.1 Difference between the General-Purpose Editor and Software Package

It is possible to use BASIC programs created in a general-purpose editor instead of using the type SW1IVD-AD51HP/SW1NX-AD51HP software package in the communication module.

As described below, BASIC programs are created differently in the general-purpose editor and the software package.

General-purpose editor : BASIC programs can be edited online. Their operations cannot be checked while the programs are being executed.

Software package : BASIC programs can be created either online or offline. In online programming, it is possible to execute programs and then edit them while checking their operations.

6 - 1

6.2 Flow of BASIC Program Creation Using a General-Purpose Editor

The flow chart below illustrates the procedure from creation to execution of a BASIC program using a general-purpose editor.

1) Select DOS from the basic utility menu.
↓
2) Start up the general-purpose editor. :
   See the manual of the general-purpose editor in question for start-up method.
↓
3) Create the BASIC program using the general-purpose editor.
↓
4) Save the BASIC program. :
   The program should be saved in the location C:\AD51H\USR\○○○ and have the file extension .BAS.
↓
5) End the general-purpose editor.
↓
6) Start up the line numbering tool to add line numbers. :
   Enter DRENUM <file name> and add line numbers.
↓
7) Start up the software package.
↓
8) Execute the program using the interpreter to check the operation.

6.3 Software Required to Create Programs with a General-Purpose Editor

It is necessary to purchase any of the following software in order to create programs using a general-purpose editor.

Any software that can convert text can be used.

Word
Word Pad
Notepad
Ichitaro
MIFES, etc.

6 - 2

6.4 Precautions when Using a General-Purpose Editor

Please take the following precautions when using a general-purpose editor.

(1) About the end of line processing within a program

Make sure to enter CR(&H0A) or LF(&H0A) at the end of each line in the program.
(In a general-purpose editor, they are automatically entered by pressing the Return key (↵) or the Enter key.)
Failing to enter CR or LF will cause an error.
The file has nonetheless been read, it should be saved using the BASIC instruction SAVE as is in order to make it a valid file.

(2) About the end of file processing of a BASIC program file

Enter EOF(&H1A) at the end of a BASIC program file.
(In a general-purpose editor, EOF is normally appended automatically if no specific action is taken.)

(3) About control codes in a program

If a program contains control codes, it does not work as a normal program.

(4) About description of the PRINT instruction

The PRINT command can be expressed by the abbreviation "?," but this abbreviation cannot be used in a general-purpose editor.

(5) About assigning line numbers

Assign line numbers from the start line in ascending order.

(6) About the number of characters in one line

The maximum number of characters that can be contained in one line is 254.
(Here, "one line" refers to until the end of the line.)

6.5 Addition of Line Numbers Using the Line Numbering Tool

This section explains how to start up the line numbering tool and the precautions on the use.
The line numbering tool is provided with the type SW1IVD-AD51HP/SW1NX-AD51HP AD51H-BASIC software package.

6.5.1 Starting up the Line Numbering Tool

It is necessary to start up the line numbering tool in order to add or modify line numbers in a program created using a general-purpose editor.

It is explained below how to start the line numbering tool and specify each option.

DRENUM [-s XXX] [-t XXX] [-i XXX] [-e XXX] source file name [.BAS] [output file name]

-s XXX	:	Specify a new start line number at XXX. If this is omitted, "10" is used.
-t XXX	:	Specify the previous start line number at XXX. If this is omitted, the line at the beginning of the program is used.
-i XXX	:	Specify an increment value at XXX. If this is omitted, "10" is used.
-e XXX	:	Specify the line number where the line number change operation should end at XXX. If this is omitted, the last line of the program is used.
Source file name	:	Specify the source file name of the BASIC program. If no extension is specified, it is assumed to be ".BAS. "
Output file name	:	Specify the name of the output file to which the result of the line number change operation should be output. If no name is specified, the file name obtained by changing the extension of the source file name to ".BAS" is used as output file name. If only the extension is not specified, the extension of an output file name is assumed to be ".BAS." The extension of the source file is changed to ".OLD."

POINT

Make sure to use lowercase characters to specify the options.

DRENUM -s 10 -t 100 TEST.BAS

Lowercase

The procedure below shows an example of the steps involved when using the line numbering tool to add line numbers to a program created in a general-purpose editor.

1) Create the program using the general-purpose editor (the program is created without line numbers).

```
Source file name: C:\AD51H\USR\TEST.BAS
'Branching according to the condition.
INPUT "X = " ;X
IF X>=0 AND X<=10 GOTO ＊OK ELSE ＊NG
＊OK
PRINT "Within the interval from 0 to 10."
END
＊NG
PRINT "Outside the interval. "
END
```

2) Save the program

3) End the general-purpose editor.

4) Check that the program was created.

5) Start up the line numbering tool.

```
C: \>DRENUM TEST.BAS

C: \>C:
C: \>CD\AD51H\USR>

C: \AD51H\USR>C: \AD51H\SYSTEM\DRENUM TEST.BAS

C: \AD51H\USR>CD\

C: \>
```

6) The addition of line numbers is completed. : The line number addition is complete.

7) Check the program : The source file is saved with the extension ".OLD."

```
10 'Branching according to the condition.
20 Input "X = " ;X
30 IF X>=0 AND X<=10 GOTO＊OK ELSE＊NG
40 ＊OK
50 PRINT "Within the interval from 0 to 10."
60 END
70 ＊NG
80 PRINT "Outside the interval."
90 END
```

6.5.2 Precautions when Using the Line Numbering Tool

Please take the following precautions when using the line numbering tool.

(1) About handling identical file names

If the source file and output file have the same file names, the line numbering tool changes the extension of the source file to ".OLD" and then performs the processing. If a file with the file name in question and the extension ".OLD" already exists, that file will be overwritten.

(2) About the number of characters in one line

An error will occur if the number of characters in one line exceeds 254 as a result of changing the number of characters in one line or reassigning line numbers to a source file.

(3) About syntax error of a program

Line numbers may not be reassigned properly on the lines that caused the errors if syntax errors or similar occur.

(4) About situations where line number cannot be reassigned

Please be cautious in the following cases where line number cannot be reassigned.

(a) Line numbers of other programs

The line number of execution start for the CHAIN instruction

Example:

```
CHAIN MERGE "0 : A. BAS" , 200, ALL DELETE 500-1000
                           ↑                ↑  ↑
                    Not reassigned          Reassigned
```

However, the line numbers in the DELETE option are reassigned.

(b) Line numbers for instructions that cannot be used in a program

Line number of instructions AUTO, DELETE, LIST, LLIST, MERGE, and RENUM

Example:

```
LIST 100
     ↑
Not reassigned
```

(c) About handling errors caused by the line numbering tool

The source file is processed according to the following if a line number change operation is forcefully stopped while the line numbering tool is being run, or the reading/writing or renaming of a file failed due to I/O error, etc.

- If the processing has not reached renaming of a source file, the source file remains as is.
- If the source file has been renamed, the extension changes to ".OLD".

(d) About temporary work files used by the line numbering tool

The file names shown below are reserved for files that are temporarily created by the line numbering tool. Thus, the user should not use them.

DRENUM.TMP Work file 1
D_NCHT.TMP Work file 2

7 CREATING PROGRAMS USING A COMPILER

This chapter explains how to create BASIC programs using a compiler.

7.1 Differences between Compiler BASIC and Interpreter BASIC

It is possible to use both compiler BASIC and interpreter BASIC in the communication module.
Programs created in compiler BASIC and interpreter BASIC run differently in the following way:

Compiler BASIC : In this type of BASIC the program is compiled (the instructions are translated into machine language) once it is completed, and the communication module executes the machine code directly.

Interpreter BASIC : In this type of BASIC the communication module translates the program into machine language during the execution of the program.

It is difficult to determine which type is better. The following table compares the advantages and disadvantages of Compiler BASIC and Interpreter BASIC.

	Advantage	Disadvantage
Compiler BASIC	• Execution speed is fast.	• Debugging is difficult. • There are many detailed restrictions.
Interpreter BASIC	• Debugging is easy.	• Execution is slower than Compiler BASIC.

Select either Compiler BASIC or Interpreter BASIC according to the intended purpose.

It is not necessary to read this chapter if Interpreter BASIC is used.
If Compiler BASIC is used, on the other hand, please read this chapter and understand the restricted items before starting to use it.

7.2 Flow of Program Creation Using a Compiler

The following flow chart provides an overview of how to create and compile a BASIC program, and run it on the communication module.

1) Start the software package.
   C:\>AD51HP
   : Start up the type SW1IVD-AD51HP/SW1NX-AD51HP software package from MS-DOS.

↓

2) Create a BASIC program through online programming.

↓

3) Run the program using the interpreter to check the operation.
   : Check whether it runs within the restrictions of the compiler.

↓

4) Save the program to the hard disk using the SAVE instruction.
   : Save to drive C (specify drive number 3) of the hard drive.

↓

5) End the software package.

↓

6) Select DOS from the basic utility menu.

↓

7) Run the compiler.
   : Enter DBC<file name> and compile.

↓

8) Check the executable file (.EXE).
   C:\AD51H\USR>DIR /W
   : After running the compiler, check that the -.EXE file has been created

↓

9) Restart the software package.
   C:\>AD51HP
   : Start the type SW1IVD-AD51HP/SW1NX-AD51HP software package from MS-DOS.

↓

10) Select Online programming from the menu and change to system mode.

↓

11) Save the program to an executable program area of a memory card with the MSAVE command.
   : MSAVE<task number>

↓

12) Change the start condition to "START" with the SET command.
   : Change the start condition from "BOOT" to "START."

↓

13) Run the program.
   : Change the mode of the communication module to Run mode to execute the program. (Refer to Section 2.2.)

7

7.3 Software Required for Compilation

An assembler and linker are required to compile a program created in the communication module. Because the IVD-AD51HP software package does not include an assembler and linker; these must be purchased separately.
Some assembler and linker products are recommended below.

• Recommended products (assembler and linker)

(1) For IBM PC/AT compatible PCs

Please purchase the product Turbo Assembler 5.0.
For Turbo Assembler, contact Borland Software Corporation.

Product name	Type	Remarks
Turbo Assembler	Turbo Assembler Ver 5.0	English version for IBM PC/AT compatible

7.4 Installing Assembler and Linker

Refer to the manual for the software for how to install the assembler and linker for compilation.

POINT

The path of the assembler and linker should be added to the Autoexec.bat file. Otherwise, they will not run.

7.5 Starting up the Compiler

In order to compile a created program, it is necessary to start up the compiler. The compiler can be run by simply entering DBC<file name> in MS-DOS.

The format for the compiler and specification of each option are described in the following.

DBC [-4] [-6] [-v] [-w-] [-d] source file name [.BAS] [executable file name.EXE]

Source file name : Specify the name of the BASIC source file. If no extension is specified, it is assumed to be ".BAS." Enter the system name followed by the file name if there is a system name involved.

Executable file name.EXE : Specify the name of the executable file to which the result of the compilation should be output. If no name is specified, the file name obtained by changing the extension of the source file name to ".EXE" is used as the execution file name.

-4 : Specify this option when using Ver 4.0. If it is not specified, it is assumed that Ver 5.0 is used.

-6 : Specify this option when using Ver 6.0. If it is not specified, it is assumed that Ver 5.0 is used.

-v : Specify this option to display the status during compilation in detail.

-w- : Specify this option in order to prevent warnings from being displayed.

-d : Specify this option in order to make the compiled program perform the following error checks at execution (used for debugging).

- Check addition, subtraction, and multiplication operations on integer values at execution, and generate an 'Overflow' error in case an overflow occurs.
- Check array subscript ranges during execution, and generate a 'Subscript out of range' error if a reference to an array entry outside the subscript range occurs.

Note, however, that if this option is specified the size of the executable program becomes bigger and the execution speed becomes slower.

POINT

Make sure to use lowercase characters to specify the options.

DBC -v -w TEST.BAS

Lowercase

7.5.1 For IBM PC/AT Compatible PCs

The procedure below shows an example of the steps involved when compiling a program created by the interpreter (file name: INTER.BAS) to an executable program (file name: COMP.EXE).

1) C:\ AD51H\USR>c:\ ad51h\system\DBC -6 -v INTER.BAS COMP.EXE ↵
: Compilation start. The options -6 and –v are specified.
C: \ >c:
C: \>cd\ad51h\usr

2) FATAL --- 'masm' failed:No such file or directory : Ends with an error.
compiler aborted

3) Check that the following file is created.
basic$$$.inc
$n.asm
(A number is placed in stead of n. Several $n.asm files may be created.)

4) Start Turbo Assembler and assemble the program.
C:\ AD51H\USR>TASM $n.asm ↵
(If there are several $n.asm files, repeat this step for each file.)

5) Check that the following file is created.
$n.obj
(A number is places instead of n. Several $n.obj files may be created.)

6) C:\ AD51H\USR>TLINK c:\ad51h\system\dbb.obj $n.obj,COMP,,c:\ad51h\system\dbc.lib ↵
: Start the linker.

If there are several $n.obj files, they should be listed separated by space.
$1.obj $2.obj $3.obj

Warning : No stack : Ignore this warning.
C: ¥AD51H\USR>

7) C:\AD51H\USR> : The compilation is complete.

7.6 Precautions when Compiling

(1) About compilation

Make sure to use Microsoft Macro Assembler or Turbo Assembler when compiling. It is not possible to compile programs with assemblers other than Microsoft Macro Assembler or Turbo Assembler.

(2) About work files created by the compiler

The file names shown below are reserved for files the BASIC compiler creates. The user should not use them.

$n.ASM	: Assembler source file	A number is placed in stead of n.
$n.OBJ	: Object file	
BC.TMP	: Temporary work file	
BASIC$$$.INC	: Include file	

(3) About errors at compilation

Errors generated during compilation are displayed on the screen as well as stored in the assembler source file $n.ASM. Errors can also be viewed by referring to this file.

(4) About checking the correct operation of a program

Make sure to check thoroughly that the program operates correctly by running it using the interpreter before attempting to compile it. The program cannot be edited once it has been compiled. If an error occurs after the program is compiled and run, it will be necessary to correct and re-compile the program.

(5) About the warning when using the DBC compiler

The DBC compiler displays the warning 'LINK: warning L4021:no stack segment' during the linking. This does not cause problems in the operation of a compiled program; please ignore it.

(6) About the size of variables

The interpreter uses only the string area for the length of a string, while the compiler always uses 256 bytes per variable.

(7) About the execution order of expressions

The compiler optimizes the expressions in order to improve the execution speed. Therefore, the priority order and combinatory rules do not change, but the order in which items in an expression are executed may not be the same.
For example, in case of the expression ASC (INKEY$)-ASC (INKEY$)＊2, it is not given beforehand whether the ASC (INKEY$)＊2 part or the ASC (INKEY$) part is executed first. If the result of the compiled expression comes out differently from the interpreter, the expression should be divided and the intermediate result temporarily stored in a variable, etc., after which the program can be executed.

(8) About conversion of integers to real numbers in calculations

In addition, subtraction, multiplication, and division between integers, if an intermediate result of an expression is outside the integer range, the interpreter automatically converts it to a real number and performs the calculation. However, the compiler still performs the calculation within the integer range. In this case, the integer values can be explicitly converted to real numbers using the CSNG and CDBL functions.

7.7 How to Run a Program in the Communication Module

In order to run a compiled program in the communication module, it is necessary to register the program to the executable program area of a memory card/EEP-ROM/flash ROM.
The procedure for registration to a memory card/EEP-ROM/flash ROM is shown in the following.

MSAVE<task number> [,[V], "<file name>" [, location]]
or
MSAVE<task number> [,V]

Task number : Specify the BASIC task number in the executable program area.
AD51H-S3: 1 to 8
A1SD51S/QD51 (-R24): 1 or 2

V : Specify whether or not to check if the contents match. After the writing is completed, it will be checked if the contents of the main memory and the memory card/EEP-ROM/flash ROM match.

"File name" : Specify the name of the compiled file (*.EXE) to be read to the main memory.

Location : Specify a location to which the task is assigned.
0, 16, 32, (multiples of 16) to 368
If this option is omitted, the assignment is performed automatically.

POINT

- An "Error: Location" error may occur during the MSAVE operation, making the location assignment impossible. If this occurs, specify an empty location and perform the MSAVE operation. Alternatively, set the start conditions of all tasks to OFF and perform the MSAVE operations one by one.
- A "System:code=824" error may occur during the MSAVE operation, indicating that there is no work area. If this occurs, set the start conditions of all tasks to OFF and perform the MSAVE operation for all tasks again. Reset the communication module after performing the MSAVE operation.

The procedure below shows an example of the steps involved when performing an MSAVE operation on a compiled file (COMP.EXE in this example) to task No. 1 at location 32.

1) S>MSAVE 1,,"3:COMP.EXE" ↵ : Save to the executable program area of task 1.
   SAVE (Y/N) ?Y : Select Y.
   SAVE OK
   S> : The save is complete.

2) S>SET 1,START ↵ : Change start-up condition from "BOOT" to "START."
   SET OK
   S>

3) S>GO R ↵ : Change to run mode.

7.8 Instruction/Function List

7.8.1 List of Whether or not Instructions/Functions can Be Compiled

The table below shows whether or not each command can be compiled.

○ : Can be compiled without any difficulties
△ : Can be compiled with restrictions
× : Not supported by the compiler

Table 7.1 List of whether or not instructions/functions can be compiled

Instruction/function	Compilation	Remarks	Reference page, reference section
ABS	○		Commands 11-2
ASC	○		Commands 11-3
ATN	△	With restriction	Section 7.8.2-1
AUTO	×	Not supported by the compiler	Section 7.8.2-2
BEEP	○		Commands 11-7
BIN$	△	With restriction	Section 7.8.2-3
BSWAP	○		Commands 11-10
CDBI	○		Commands 11-13
CDBL	○		Commands 11-15
CHAIN	×	Not supported by the compiler	Section 7.8.2-4
CHR$	○		Commands 11-19
CIDB	○		Commands 11-20
CINT	○		Commands 11-22
CISN	○		Commands 11-24
CLEAR	×	Not supported by the compiler	Section 7.8.2-5
CLOSE	○		Commands 11-27
CLS	○		Commands 11-28
COM ON/OFF/STOP	○		Commands 11-29
COMMON	×	Not supported by the compiler	Section 7.8.2-6
CONSOLE	○		Commands 11-31
CONT	×	Not supported by the compiler	Section 7.8.2-7
COS	△	With restriction	Section 7.8.2-8
CSNG	○		Commands 11-34
CSNI	△	With restriction	Section 7.8.2-9
CVD	○		Commands 11-37
CVDMBF	○		Commands 11-38
CVI	○		Commands 11-39
CVS	○		Commands 11-40
CVSMBF	○		Commands 11-41
DATA	△	With restriction	Section 7.8.2-10
DATE$	○		Commands 11-45
DEFDBL	△	With restriction	Section 7.8.2-11
DEFFN	△	With restriction	Section 7.8.2-12

(Continued on the following page)

Table 7.1 List of whether or not instructions/functions can be compiled (continued)

Instruction/function	Compilation	Remarks	Reference page, reference section
DEFINT	△	With restriction	Section 7.8.2-13
DEFSNG	△	With restriction	Section 7.8.2-14
DEFSTR	△	With restriction	Section 7.8.2-15
DEF ZEVENT	○		Commands 11-53
DELETE	×	Not supported by the compiler	Section 7.8.2-16
DIM	△	With restriction	Section 7.8.2-17
END	○		Commands 11-59
EOF	○		Commands 11-60
ERASE	×	Not supported by the compiler	Section 7.8.2-18
ERL	○		Commands 11-62
ERR	○		Commands 11-63
ERROR	○		Commands 11-64
EXP	△	With restriction	Section 7.8.2-19
FIELD	○		Commands 11-66
FILES	×	Not supported by the compiler	Section 7.8.2-20
FIX	○		Commands 11-68
FOR-NEXT	△	With restriction	Section 7.8.2-21
FORMAT	○		Commands 11-71
FRE	△	With restriction	Section 7.8.2-22
GET	○		Commands 11-74
GETMEM	○		Commands 11-75
GOSUB RETURN	△	With restriction	Section 7.8.2-23
GOTO	○		Commands 11-81
HEX$	△	With restriction	Section 7.8.2-24
IF GOTO ELSE	○		Commands 11-83
IF THEN ELSE	○		Commands 11-85
INKEY$	○		Commands 11-87
INPUT	△	With restriction	Section 7.8.2-25
INPUT$	○		Commands 11-91
INPUT#	○		Commands 11-94
INSTR	○		Commands 11-95
INT	○		Commands 11-97
KEY	○		Commands 11-99
KEYLIST	×	Not supported by the compiler	Section 7.8.2-26
KILL	○		Commands 11-101
LEFT$	○		Commands 11-103
LEN	○		Commands 11-105
LET	○		Commands 11-106
LFILES	×	Not supported by the compiler	Section 7.8.2-27
LINE INPUT	△	With restriction	Section 7.8.2-28
LINE INPUT#	○		Commands 11-111
LIST	×	Not supported by the compiler	Section 7.8.2-29
LLIST	×	Not supported by the compiler	Section 7.8.2-30
LOAD	×	Not supported by the compiler	Section 7.8.2-31
LOC	○		Commands 11-115
LOCATE	○		Commands 11-116

(Continued on the following page)

Table 7.1 List of whether or not instructions/functions can be compiled (continued)

Instruction/function	Compilation	Remarks	Reference page, reference section
LOF	○		Commands 11-118
LOG	△	With restriction	Section 7.8.2-32
LPRINT	○		Commands 11-120
LPRINT USING	○		Commands 11-121
LSET	○		Commands 11-122
MERGE	×	Not supported by the compiler	Section 7.8.2-33
MID$ (Part 1)	○		Commands 11-125
MID$ (Part 2)	○		Commands 11-127
MKD$	○		Commands 11-128
MKDMBF$	△	With restriction	Section 7.8.2-34
MKI$	○		Commands 11-131
MKS$	○		Commands 11-132
MKSMBF$	△	With restriction	Section 7.8.2-35
NAME	○		Commands 11-135
NEW	×	Not supported by the compiler	Section 7.8.2-36
OCT$	△	With restriction	Section 7.8.2-37
ON COM GOSUB	△	With restriction	Section 7.8.2-38
ON ERROR GOTO	○		Commands 11-145
ON GOSUB	○		Commands 11-147
ON GOTO	○		Commands 11-149
OPEN	○		Commands 11-151
PCRD	○		Commands 11-153
PCWT	○		Commands 11-239
PRINT	○		Commands 11-327
PRINT USING	△	With restriction	Section 7.8.2-39
PRINT#	○		Commands 11-332
PRINT# USING	△	With restriction	Section 7.8.2-40
PUT	○		Commands 11-334
PUTMEM	○		Commands 11-335
RDSET	△	With restriction	Section 7.8.2-41
READ	△	With restriction	Section 7.8.2-42
REM	○		Commands 11-343
RENUM	×	Not supported by the compiler	Section 7.8.2-43
RESTORE	△	With restriction	Section 7.8.2-44
RESUMU	△	With restriction	Section 7.8.2-45
RIGHT$	○		Commands 11-347
RND	○		Commands 11-348
ROT	△	With restriction	Section 7.8.2-46
RSET	○		Commands 11-351
RUN (1)	×	Not supported by the compiler	Section 7.8.2-47
RUN (2)	○		Commands 11-353
SAVE	×	Not supported by the compiler	Section 7.8.2-48
SEARCH	○		Commands 11-355
SGN	○		Commands 11-357
SHA	△	With restriction	Section 7.8.2-49
SHT	△	With restriction	Section 7.8.2-50
SIN	△	With restriction	Section 7.8.2-51

(Continued on the following page)

Table 7.1 List of whether or not instructions/functions can be compiled (continued)

Instruction/function	Compilation	Remarks	Reference page, reference section
SPACE$	○		Commands 11-363
SPC	△	With restriction	Section 7.8.2-52
SQR	△	With restriction	Section 7.8.2-53
STOP	△	With restriction	Section 7.8.2-54
STR$	○		Commands 11-367
STRING$	○		Commands 11-369
SYSTEM	×	Not supported by the compiler	Section 7.8.2-55
SWAP	○		Commands 11-372
TAB	△	With restriction	Section 7.8.2-56
TAN	△	With restriction	Section 7.8.2-57
TIME$	○		Commands 11-375
TROFF	×	Not supported by the compiler	Section 7.8.2-58
TRON	×	Not supported by the compiler	Section 7.8.2-59
VAL	△	With restriction	Section 7.8.2-60
WHILE WEND	△	With restriction	Section 7.8.2-61
WIDTH	○		Commands 11-385
WTSET	△	With restriction	Section 7.8.2-62
ZBAS	○		Commands 11-388
ZCLOSE	○		Commands 11-389
ZCNTL	○		Commands 11-390
ZEVENT	○		Commands 11-412
ZIDV	○		Commands 11-413
ZLDV	○		Commands 11-414
ZMESSAGE	○		Commands 11-415
ZMESSAGE CLOSE	○		Commands 11-418
ZMESSAGE GET	○		Commands 11-419
ZMESSAGE KILL	○		Commands 11-421
ZMESSAGE OPEN	○		Commands 11-422
ZMESSAGE PUT	○		Commands 11-423
ZMOVE	○		Commands 11-425
ZODV	○		Commands 11-430
ZOPEN	○		Commands 11-431
ZRECEIVE	○		Commands 11-433
ZRELEASE	○		Commands 11-439
ZRESERVE	○		Commands 11-440
ZSEND	○		Commands 11-442
ZSIGNAL	○		Commands 11-448
ZSTART	△	With restriction	Section 7.8.2-63
ZURGENCY	○		Commands 11-451
ZWAIT DELAY	○		Commands 11-452
ZWAIT EVENT	○		Commands 11-453

7.8.2 Instructions/Functions with Different Specifications at Compilation

This section provides a more detailed explanation of instructions whose specifications are different when using the compiler instead of executing them using the interpreter. Instructions other than the ones explained here can be used with the same specifications as in the interpreter; please refer to the explanation for the interpreter.

No.	Instruction/function	Specifications different from the interpreter, restrictions, and precautions	Corrective action	Reference page in the Commands
1	ATN	• If <numerical expression> contains a double-precision real number a double-precision value is returned, otherwise a single-precision value is returned.	• Use the CSNG function on the <numerical expression> to make sure that it becomes single-precision.	11-4
2	AUTO	• The AUTO instruction is not supported. • AUTO can be used as a variable name.	—	11-5
3	BIN$	• If a value outside the range from -32758 to 65535 is specified in <numerical expression>, the result is the same as if 32767 is specified.	• Check the range using the IF instruction before the BIN$ function, and generate an error using the ERROR instruction if necessary.	11-8
4	CHAIN	• The CHAIN instruction is not supported. • A "not supported" error is generated at compilation.	• Substitute with the RUN (2) instruction. • Be careful when using the RUN (2) instruction, however. It does not support the following functionality of the CHAIN instruction. • The program edit function by program merge and delete: Prepare a program already edited (that has been merged and surplus code deleted). • Execution from a specified line number: Pass the line number to the program using the GETMEM and PUTMEM instructions, and use the value to jump to the target line number by the ON GOTO instruction. • Passing variables by the ALL options: Pass variables to the program using the GETMEM and PUTMEM instructions.	11-17
5	CLEAR	• The CLEAR instruction is ignored. • The instruction is Ignored at compilation.	• Clear the variable using an assignment instruction.	11-26
6	COMMON	• The COMMON instruction is not supported. • A "not supported" error is generated at compilation.	• Pass variables to the program using the GETMEM and PUTMEM instructions.	11-30

No.	Instruction/function	Specifications different from the interpreter, restrictions, and precautions	Corrective action	Reference page in the Commands
7	CONT	• The CONT instruction is not supported. • A "not supported" error is generated at compilation.	—	11-32
8	COS	• If <numerical expression> contains a double-precision real number, a double-precision value is returned, otherwise a single-precision value is returned.	• Use the CSNG function on the <numerical expression> to make sure that it becomes single-precision.	11-33
9	CSNI	• The CSNI instruction does not check overflow.	• Check the range using the IF instruction before the CSNI function, and generate an error using the ERROR instruction if necessary	11-35
10	DATA	• Double quotation marks (") can be used only as symbols to enclose a string constant in the DATA instruction. • If one of a pair is missing, everything from the beginning of the expression to " or from " to the end of the line is regarded as data.	• Make sure to use the double quotation marks properly.	11-43
11	DEFDBL	• Define the variable before the line where it will be used. • Variables already declared once with the DEFINT, DEFSNG, DEFDBL, or DEFSTR instructions cannot be redefined as a different type in another instruction.	• Declare the variable before the line where it will be used. • Do not redefine the variable.	11-47
12	DEFFN	• There must be no space between 'FN' and the name when defining and calling the function. • Variable type specifications must always be placed within the <name>, parameter, and function definition expression of the DEF FN instruction. • If other user-defined functions are called within the definition expression of the function, they must be defined before it is called. • It is not possible to redefine a user-defined function once it is defined.	• Place the type specification correctly. • Make sure to define other user-defined functions before they are called.	11-48
13	DEFINT	• Define the variable before the line where it will be used. • Variables already declared once with the DEFINT, DEFSNG, DEFDBL, or DEFSTR instructions cannot be redefined as a different type in another instruction.	• Declare the variable before the line where it will be used. • Do not redefine the variable.	11-50

No.	Instruction/function	Specifications different from the interpreter, restrictions, and precautions	Corrective action	Reference page in the Commands
14	DEFSNG	• Define the variable before the line where it will be used. • Variables already declared once with the DEFINT, DEFSNG, DEFDBL, or DEFSTR instructions cannot be redefined as a different type in another instruction.	• Declare the variable before the line where it will be used. • Do not redefine the variable.	11-51
15	DEFSTR	• Define the variable before the line where it will be used. • Variables already declared once with the DEFINT, DEFSNG, DEFDBL, or DEFSTR instructions cannot be redefined as a different type in another instruction.	• Declare the variable before the line where it will be used. • Do not redefine the variable.	11-52
16	DELETE	• The DELETE instruction is not supported. • DELETE can be used as a variable name.	—	11-56
17	DIM	• Specify the size of an array using the DIM instruction. Variables are not supported in <numerical expression>. • The array subscript ranges are not checked at execution. (It is possible to specify that they should be checked using the debugging option [-d] at compilation.)	• Specify the maximum size.	11-57
18	ERASE	• The ERASE instruction is ignored. • A warning is generated at compilation.	• If this instruction is used to define a new array, a maximum size array should be defined in advance and reused instead. • If the ERASE instruction is used to delete an array, simply delete it.	11-61
19	EXP	• If <numerical expression> contains a double-precision real number, a double-precision value is returned, otherwise a single-precision value is returned.	• Use the CSNG function on the <numerical expression> to make sure that it becomes single-precision.	11-65
20	FILES	• The FILES instruction is ignored. • A warning is generated at compilation.	—	11-67
21	FOR to NEXT	• There must be one-to-one correspondence between the FOR instruction and the NEXT instruction.	• Make sure that the instructions match	11-69
22	FRE	• The FRE function always returns 0. • A warning is generated at compilation.	—	11-73

No.	Instruction/function	Specifications different from the interpreter, restrictions, and precautions	Corrective action	Reference page in the Commands
23	GOSUB-RETURN	• The compiler does not check for "RETURN without GOSUB" errors.	• Count and check the GOSUB and RETURN instructions using counters.	11-79
24	HEX$	• If a value outside the range from –32678 to 65535 is specified in <numerical expression>, the result is the same as if 32767 is specified.	• Check the range using the IF instruction before the HEX$ function, and generate an error using the ERROR instruction if necessary.	11-82
25	INPUT	• The INPUT instruction does not have a screen edit function for entering values • Since overflow check is not performed when a numerical value is entered, if a large value is entered it does not generate an error but is interpreted as a negative value instead. • If the number of items separated by "," is different from the number of variables specified to be entered, the instruction displays "Redo from start" and the INPUT instruction is executed again.	• Separate the INPUT part to another task and process it with the interpreter.	11-89
26	KEYLIST	• The KEYLIST instruction is ignored. • A warning is generated at compilation.	—	11-100
27	LFILES	• The LFILES instruction is ignored. • A warning is generated at compilation.	—	11-108
28	LINE INPUT	• The LINE INPUT instruction does not have a screen edit function for entering values.	• Separate the LINE INPUT part to another task and process it with the interpreter.	11-109
29	LIST	• The LIST instruction is not supported. • LIST can be used as a variable name.		11-112
30	LLIST	• The LLIST instruction is not supported. • LLSIT can be used as a variable name.	—	11-113
31	LOAD	• The LOAD instruction is not supported. • LOAD can be used as a variable name.	—	11-114

No.	Instruction/function	Specifications different from the interpreter, restrictions, and precautions	Corrective action	Reference page in the Commands
32	LOG	• If <numerical expression> contains a double-precision real number a double-precision value is returned, otherwise a single-precision value is returned.	• Use the CSNG function on the <numerical expression> to make sure that it becomes single-precision.	11-119
33	MERGE	• The MERGE instruction is not supported. • A "not supported" error is generated at compilation.	—	11-123
34	MKDMBF$	• Converts double-precision internal representation data of IEEE format only. If data of other types is entered, it is converted assuming it is data of IEEE format.	—	11-130
35	MKSMBF$	• Converts double-precision internal representation data of IEEE format only. If data of other types is entered, it is converted assuming it is data of IEEE format.	—	11-133
36	NEW	• The NEW instruction is not supported. • NEW can be used as a variable name.	—	11-137
37	OCT$	• If a value outside the range from -32768 to 65535 is specified in <numerical expression>, the result is the same as if 32767 is specified.	• Check the range using the IF instruction before the OCT$ function, and generate an error using the ERROR instruction if necessary.	11-138
38	ON COM GOSUB	• The interpreter performs an interrupt at the beginning of each instruction, while the compiler performs an interrupt at the beginning of each line.	• Do not write any multi-statements at the first line of an interrupt processing.	11-140
39	PRINT USING	• Up to 8 display data can be described in one PRINT USING.	• Divide the statement into several PRINT USING instructions.	11-328
40	PRINT# USING	• Up to 8 display data can be described in one PRINT# USING statement.	• Divide the statement into several PRINT# USING instructions.	11-333
41	RDSET	• The compiler does not check array subscripts ranges and bit ranges.	• Check the range using the IF instruction before the RDSET function, and generate an error using the ERROR instruction if necessary	11-340

No.	Instruction/function	Specifications different from the interpreter, restrictions, and precautions	Corrective action	Reference page in the Commands
42	READ	• When reading a decimal constant, the instruction does not cause overflow even if the constant exceeds the maximum real number, and returns the maximum real number. • When reading octal and hexadecimal constants, the instruction does not cause overflow even if the constant exceeds the maximum integer value, and returns the maximum integer value. • It the type of variable in the READ instruction and the value defined in the DATA instruction do not match, a "Syntax error" occurs on the READ instruction side. • There are cases where an error is not generated. For example: 10 DATA &H000012 20 READ A! 'An error does not occur. 30 DATA &H12X 40 READ A! 'An error occurs. • If an error occurs, the data following the data that caused the error is read.	• Make sure to read correctly.	11-342
43	RENUM	• The RENUM instruction is not supported. • RENUM can be used as a variable name.		11-344
44	RESTORE	• The use of line number 0 is prohibited.		11-345
45	RESUME	• Resumes execution line by line. • Exercise caution when multi-statements are used. RESUME: Resumes execution from the line number where the instruction occurs. RESUME NEXT: Resumes execution from the beginning of the following line. RESUME line number: Resumes execution from the beginning of the specified line.	• Divide multi-statements into several lines.	11-346
46	ROT	• The ROT function does not check overflow.	• Check the range using the IF instruction before the ROT function, and generate an error using the ERROR instruction if necessary.	11-349
47	RUN (1)	• The RUN (1) instruction is not supported.		11-352
48	SAVE	• The SAVE instruction is not supported. • SAVE can be used as a variable name.		11-354

No.	Instruction/function	Specifications different from the interpreter, restrictions, and precautions	Corrective action	Reference page in the Commands
49	SHA	• Does not check argument overflow.	• Check the range using the IF instruction before the SHA function, and generate an error using the ERROR instruction if necessary.	11-358
50	SHT	• Does not check argument overflow.	• Check the range using the IF instruction before the SHA function, and generate an error using the ERROR instruction if necessary.	11-360
51	SIN	• If <numerical expression> contains a double-precision real number a double-precision value is returned, otherwise a single-precision value is returned.	• Use the CSNG function on the <numerical expression> to make sure that it becomes single-precision.	11-362
52	SPC	• Starts a new line when the SPC function is placed after the PRINT instruction.	• Place ";" (semicolon) after the SPC function.	11-364
53	SQR	• If <numerical expression> contains a double-precision real number a double-precision value is returned, otherwise a single-precision value is returned.	• Use the CSNG function on the <numerical expression> to make sure that it becomes single-precision.	11-365
54	STOP	• The STOP instruction ends a program (the same as for the END instruction). • A warning is generated at compilation.	—	11-366
55	SYSTEM	• The SYSTEM instruction is not supported. • SYSTEM can be used as a variable name.	—	11-371
56	TAB	• Starts a new line when the TAB function is placed after the PRINT instruction.	• Place ";" (semicolon) after the TAB function.	11-373
57	TAN	• If <numerical expression> contains a double-precision real number a double-precision value is returned, otherwise a single-precision value is returned.	• Use the CSNG function on the <numerical expression> to make sure that it becomes single-precision.	11-374
58	TROFF	• The TROFF instruction is not supported. It does not display errors at compilation; the operation at execution cannot be guaranteed.	—	11-377
59	TRON	• The TRON instruction is not supported. It does not display errors at compilation; the operation at execution cannot be guaranteed.	—	11-378
60	VAL	• Always returns double-precision values.	• Convert the value in question using the VAL function and then use it by assigning to a variable of required type.	11-379

No.	Instruction/function	Specifications different from the interpreter, restrictions, and precautions	Corrective action	Reference page in the Commands
61	WHILE-WEND	• There must be one-to-one correspondence between the WHILE instruction and the WEND instruction.	• Make sure that the instructions match	11-383
62	WTSET	• The compiler does not check array subscripts ranges and bit ranges.	• Check the range using the IF instruction before the WTSET function, and generate an error using the ERROR instruction if necessary.	11-386
63	ZSTART	• If the multitask setting of a task specified by <number> is "IP", this instruction starts the program in the interpreter. If it is "CP", it starts a compiled program. • If the task specified by the <number> argument to the ZSTART argument refers to compiler BASIC, the file specified by the <file> argument must be a file (-.EXE) created by DBC (BASIC compiler). If a file that is not created by DBC is specified, an error occurs or the communication module's system operates unpredictably. • Once a task is executed, it cannot be restarted without being started by specifying the file name using the ZSTART instruction.	• Make sure to specify an executable file name (-.EXE) of a compiled program to restart.	11-449

APPENDIX

Appendix-1 Error Messages When Using the Line Numbering Tool

Error message	Corrective action
There is an error in the description of an option.	Correct the specified option.
Extension '.old' cannot be specified for an output file name (output file name).	Since extension '.old' cannot be specified for the output file name, change to other extension.
There are more than 254 characters in one line. (Line number)	Change the number of characters of the corresponding line to 254 or less.
The number of characters in one line exceeded 254. (Line number)	Change the number of characters of the corresponding line to 254 or less, by considering the increased number of characters when the line number is changed.
The file cannot be opened. (File name)	Correct the file name. Check the relevant drive. (There may be no free space, it may be write protected, a floppy disk may not be mounted, the drive may be in a poor condition, etc.)
The file cannot be closed. (File name)	Correct the file name. Check the relevant drive. (There may be no free space, it may be write protected, a floppy disk may not be mounted, the drive may be in a poor condition, etc.)
The backup (file name) of the source file (file name) failed.	Correct the file name. Check the relevant drive. (There may be no free space, it may be write protected, a floppy disk may not be mounted, the drive may be in a poor condition, etc.)
The output file (output file name) cannot be created.	Correct the file name. Check the relevant drive. (There may be no free space, it may be write protected, a floppy disk may not be mounted, the drive may be in a poor condition, etc.)
Path and file name is too long.	Correct the file name.
The temporary file failed to delete. (File name)	Correct the file name. Check the relevant drive. (There may be no free space, it may be write protected, a floppy disk may not be mounted, the drive may be in a poor condition, etc.)
There is an error in the status after the line number change operation.	Line numbers changed are not in ascending order due to the way -S/-e option is specified, etc. Correct the specified option.
The corresponding file cannot be found. (File name)	Correct the file name.
Sufficient memory space cannot be allocated.	Increase the free space in memory and run again.
The drive is not ready.	Correct the file name. Check the relevant drive. (There may be no free space, it may be write protected, a floppy disk may not be mounted, the drive may be in a poor condition, etc.)
A write error occurred. (File name)	
A read error occurred. (File name)	
Abort the processing.	Another error message must have been generated before this message. Refer to that message to take corrective actions.
Warning: The referenced line number cannot be changed. (Line number)	One of the line numbers referenced in the corresponding line did not exist in the source file, etc. (The line number change operation is continued.) Correct the line numbers referenced in the corresponding line.

App

App - 1

Appendix-2 Error Messages at Compilation

This section explains the error messages generated at compilation.
Error messages are divided into the following three types.

(1) Fatal errors

These are errors that make it difficult for the compilation to continue, such as errors related to files, errors due to memory shortage and errors inside the compiler.
[Example of error display]

```
FATAL--- file I/O error
```

If a fatal error is detected, the compiler immediately stops compiling. Remove the cause of the error and compile again. When the compiler stops compiling, the message "compiler aborted" is displayed.

(2) Errors

These error messages point out instructions, etc. that cannot be compiled correctly, such as syntax errors in the BASIC program or infringements on restricted items.
[Example of error display]

```
; ; 10A$=12345
; ;          ^
---syntax error in string expression
```

The compiler detects as many errors as possible before it stops compiling. Remove the causes of the errors and check the operation using the interpreter, then compile the program again. When the compiler stops compiling, the message "compiler aborted" is displayed.

(3) Warnings

Warnings are messages that point out instructions, etc. that are problematic but not as bad as errors. Typically, a warning is generated when a statement ignored by the compiler is detected.
[Example of error display]

```
; ; 10 CLEAR
; ;    ^
```

The compiler does not stop compiling when it displays a warning. It continues to compile until the generation of an executable file (-.EXE) is complete. Check the cause of the warning and then either ignore it as it is, or modify the program and compile again. The display of warnings can be suppressed using option [-w-] of the compiler.

App - 2

[Caution]

Upon detecting an error the compiler skips reading from the location of the error to the end of the sentence. Because of this, it may detect a false error at a location immediately after the error by mistake, or conversely, it may not be able to detect an error immediately after the previous error. Consequently, errors from the next and afterward may not necessarily be correct.

The program line and ^ displayed together with an error indicate the position where the compiler was reading at the time the error was detected. The displayed position indicates the vicinity of where the error occurred. However, if the compiler cannot determine an error until it reads further in the program, the displayed position indicates the position further ahead. If it is difficult to pinpoint the location where an error occurred in lines connected by complicated expressions and multi-statement, try to divide the expressions and lines into separate pieces.

(1) Fatal Errors

Error	Meaning	Corrective action
can't create output file	An intermediate $x.ASM file (a number is placed instead of x) cannot be created in the current directory.	This may happen because directories cannot be created, etc. (directory full). Delete unnecessary files and compile again.
can't create work file #1	An intermediate BASIC$$$ INC file cannot be created in the current directory.	This may happen because directories cannot be created, etc. (directory full). Delete unnecessary files and compile again.
can't create work file #2	An intermediate BC.TMP file cannot be created in the current directory.	This may happen because directories cannot be created, etc. (directory full). Delete unnecessary files and compile again.
compiler stack overflow	There is not enough stack memory area allocated inside the compiler.	Avoid complicated expressions and use simple ones. Reduce the number of nested expressions such as FOR-NEXT. (This error typically occurs when there are 20 to 35 nested expressions.)
data area overflow (65000 bytes)	The data area necessary for variables and constants is too big to allocate.	In many cases, the cause is declaration of huge arrays. Reduce the size of the arrays.
evaluation stack overflow	An expression in one statement is too complicated.	Make the expression less complicated by assigning an intermediate value of the expression to a variable, etc.
file I/O error	An error occurred when a source file or intermediate file was accessed.	This may happen because the disk is damaged, there is not enough free space on the disk (disk full), etc. If the disk is full, delete unnecessary files and compile again.
line too long	The length of one line in a source program is too long. (This error occurs at 299 characters or more.)	This may happen because the source file was not stored by the AD51H-S3 BASIC interpreter, the source file is damaged, etc. Save the file correctly.
source file 'XXXX' not found	A source file specified in a command line cannot be found.	Specify a correct source file.

Error	Meaning	Corrective action
symbol table overflow	There are too many variables, labels, or FN functions. (This error occurs when there are 500 variables with 9-character names.)	Change the names of variables, labels, and FN functions with long names to short ones. Delete unnecessary variables, labels, and FN functions.
too many target line numbers	There are too many line numbers referenced in statements such as GOTO and GOSUB.	Replace some of the line numbers with labels, or divide the program.
unexpected end of file in 'XXXX'	A source file ends in the middle of a sentence.	Complete the program before compiling.
'dbb. obj' not found 'dbc. lib' not found	The startup module file and/or library file cannot be found.	Two files, dbb.obj and bdc.lib, must be in the compiler startup path or the current directory. Check that these two files exist.
'XXXX' failed: error level X	The assembler and linker reported an execution error.	Look up the error message in the manual for the assembler and linker to find the error cause. [Note] When compiling using Microsoft Macro Assembler Ver 4.0, this error occurs if the option [-4] is not specified.
'XXXX' failed: Exec format error	The content of the executable file of the assembler and linker is damaged.	Reinstall the assembler and linker on the hard drive again.
'XXXX' failed: No such file or directory	The assembler and linker cannot be found.	Place the assembler and linker in the current directory or a directory specified in the environment variable path.
'XXXX' failed: Not enough memory	The assembler and linker cannot be started because there is not enough memory.	Increase the available memory by: canceling the residence of a terminate-and-stay-resident program for the PRINT command, etc., disabling unnecessary device drivers, making the BUFFERS specifications for CONFIG and SYS smaller, etc.

(2) Errors

Error	Meaning	Corrective action
bad line number XXXXX	Syntax error: A line number is outside the range from 1 to 65529.	Use line numbers in the range from 1 to 65529.
DEF --- syntax error	Syntax error: There is a syntax error in the DEFINT, DEFSNG, DEFDBL, and DEFSTR instructions.	Check the content of a program and make the appropriate correction to the program.
DEF --- what?	Syntax error: There is a syntax error in the DEF instruction.	Check the content of a program and make the appropriate correction to the program.
DIM --- syntax error	Syntax error: There is a syntax error in the DIF instruction.	Check the content of a program and make the appropriate correction to the program.
divide by 0	Illegal parameter: Somewhere in a numerical expression a value is divided by 0 (/, MOD).	Correct the program in such a way that there is no division by 0. [Note] The compiler detects only division between constants. Division by 0 at execution does not cause an error, but returns the maximum number. Single-precision: + 1.70141E+38 Double-precision: + 1.70141183460469D+38
expression too complex	Compiler restricted item: A numerical expression of real number is too complicated.	Make the expression less complicated by assigning an intermediate value of the expression to a variable, etc.
FOR --- syntax error	Syntax error: • There is no assignment statement for a control variable or initial value in the FOR – NEXT instruction. • A string variable is used as a control variable by mistake.	Specify the control variable or initial value correctly.
FOR without NEXT	Syntax error: • The NEXT corresponding to a FOR cannot be found. • The FOR and NEXT instructions do not match.	Modify the program so that the FOR and NEXT instructions match.

Error	Meaning	Corrective action
GOSUB not found GOSUB/GOTO not found GOTO not found	Syntax error: GOTO/GOSUB cannot be found in the ON XX GOSUB/GOTO instruction.	Make the appropriate correction to the program.
illegal constant	Syntax error: A character not allowed as a numerical value is used in an octal or hexadecimal constant.	Make the appropriate correction to the program.
illegal parameter	Illegal parameter: • There is an illegal parameter. • A variable or array name is required but another data type is specified.	Make the appropriate correction to the program.
index must be 0...32766	Compiler restricted item: In the DIM instruction, the value for the array size specification is too large or a variable or expression is used for the size specification.	Reduce the size. Change the size specification to a constant.
INPUT---, or; not found	Syntax error: Characters other than ", " or ";" are placed after INPUT "<character>."	Make the appropriate correction to the program.
LINE INPUT---';' not found	Syntax error: Characters other than "; " are placed after INPUT "<character>."	Make the appropriate correction to the program.
LINE INPUT---must be string variable	Syntax error: A value other than a string variable is specified for the input destination variable in the LINE INPUT instruction.	Specify a string variable as the storing destination variable.
line number not found	Syntax error: There are no line numbers in a source program.	Add the line numbers.
line number or label not found	Syntax error: • There is no line number or label after GOTO or GOSUB. • An illegal label name is used in connection with THEN, ELSE, RETURN, RESUME, and RESTORE instructions.	• Specify line numbers and labels. • Specify a correct label name.

App - 6

Error	Meaning	Corrective action
line number XXXXX not sequential	Syntax error: The line numbers in a source program are not in ascending order.	Change the line numbers in such a way that they are in ascending order.
missing operand	Syntax error: An argument or expression does not exist where it is necessary.	Specify the missing argument or expression.
NEXT without FOR	Syntax error, compiler restricted item: NEXT without corresponding FOR was detected.	Specify the corresponding FOR.
ON---line number or label not found	Syntax error: There is a syntax error in the sequence of line number or label in the ON XX GOSUB/GOTO instruction.	Make the appropriate correction to the program according to the format.
ON---string expression not allowed	Syntax error: A string expression is used in the <expression> in the ON<expression>GOSUB/GOTO instruction.	Specify a numerical expression for <expression>.
OPEN---file name not found	Syntax error: There is no text string containing the file name for the OPEN instruction.	Specify the file name correctly.
OPEN--- INPUT/OUTPUT/APPEND not found	Syntax error: Text other than INPUT, OUTPUT, and APPEND are placed after OPEN "XXX" FOR.	Specify either INPUT, OUTPUT, or APPEND.
parameter must be numerical expression	Illegal parameter: A numerical expression is required but another data type is specified.	Specify a numerical expression.
parameter must be string expression	Illegal parameter: A string expression is required but another data type is specified.	Specify a string expression.
parameter must be variable	Illegal parameter: A variable is required, but another data type is specified.	Specify a variable.

Error	Meaning	Corrective action
port number must be constant	Illegal parameter: A port number (%X) must be an integer constant.	Specify an integer constant.
PRINT USING---';' not found	Syntax error: There is a character other than ";" in a format string in the LPRINT/PRINT USING instruction.	Change to ";."
PRINT USING---format string not found	Syntax error: There is a character other than a format character after USING in the LPRINT/PRINT USING instruction.	Specify a format string.
PRINT USING---illegal parameter	Syntax error: There is a syntax error in the sequence of display data in the LPRINT/PRINT USING instruction. (Limited to 8 data items in the compiler.)	Make the appropriate correction to the program according to the syntax.
PRINT USING---too many parameters	Compiler restricted item: There are too many display data items lined up for the LPRINT/PRINT USING instruction. (Limited to 8 data items in the compiler.)	Divide the PRINT USING statement into several PRINT USING instructions.
RESTORE---line number or label not found	Syntax error: A line number/label is required after RESTORE but something else is specified.	Specify a line number or label.
RESUME---syntax error	Syntax error: There is a syntax error in the RESUME instruction.	Make the appropriate correction to the program according to the format.
statement expected	Syntax error: An assignment statement or instruction statement is required at the beginning of a sentence, but something else (e.g., a function, constant, symbol) is specified.	Modify the program and place assignment statement or instruction at the beginning of the sentence.
STEP---string expression not allowed	Syntax error: A string expression is used as the increment value indicated by STEP in the FOR – NEXT instruction.	Make the appropriate correction to the program according to the format of the instruction.

Error	Meaning	Corrective action
string expression not allowed	Illegal parameter: An illegal string variable/expression is used.	This error occurs when a string variable/expression is specified where a numerical value/expression is required. Modify the program according to the format of the instruction.
string expression too complex	Compiler restricted item: A string expression is too complicated.	Make the expression less complicated by assigning an intermediate value of the expression to a variable, etc.
subscript out of range	Illegal parameter: • The subscript of an array is beyond the range specified by the DIM instruction. • The array dimensions do not match.	• Modify the program so that the subscript remains within the range specified by the DIM instruction. • Make the appropriate correction to the program so that the array dimensions match.
swap type mismatch	Illegal parameter: The types of two variables is not corresponding by the SWAP instruction.	The program corrected as the types of the two variables is corresponding.
syntax error	Syntax error: There is a syntax error.	Make the appropriate correction to the program according to the format of the instruction or function.
syntax error---binary operator	Syntax error: There is only one argument specified for a binary operator.	Make the appropriate correction to the program so that the calculation expression can be evaluated.
syntax error at end of statement	Syntax error: There is a reserved word, symbol, or expression at the end of a sentence.	Make the appropriate correction to the program according to the format of the instruction or function.
syntax error in expression	Syntax error: There is a syntax error in an expression.	Make the appropriate correction to the program in a correct format.
syntax error in function parameter list	Syntax error: There is a syntax error in the parameter list of the FN part of the DEF FN function.	Make the appropriate correction to the program according to the format of the function.
syntax error in parameter	Syntax error: There is a syntax error in the sequence of statement/function arguments.	Make the appropriate correction to the program according to the format of the instruction or function.

Error	Meaning	Corrective action
syntax error in string expression	Syntax error: • There is a syntax error in a string expression. • An operator that cannot be used for a string expression is used.	Make corrections to the program so that an appropriate operator can be used.
THEN/GOTO not found	Syntax error: There is no THEN/GOTO after a conditional statement in the IF instruction.	Make the appropriate correction to the program to include THEN or GOTO.
TO---string expression not allowed	Syntax error: The last value in a FOR loop indicated by TO is a string expression	Specify a numerical expression or numerical variable.
TO not found	Syntax error: There is no TO corresponding to a FOR instruction.	Make the appropriate correction to the program.
type mismatch	Illegal parameter: • Types do not match. • In an argument in an instruction/function, a value is required but a string is passed or a string is required but a value is passed.	Make the appropriate correction to the program to make sure the types match.
WEND without WHILE	Syntax error, compile restricted item: A WEND instruction without a corresponding WHILE instruction was detected.	Make modification so that the WHILE and WEND instructions match.
WHILE without WEND	Syntax error: There is no WEND corresponding to a WHILE instruction.	Make modification so that the WHILE and WEND instructions match.
XXXXX---file number not found	Syntax error: There is no file number or a non-value data type is specified at the position of the file number in statement XXXXX.	Specify a file number.
XXXXX---name too long	Syntax error: A name XXXXX of a variable or FN function is too long (maximum 15 characters).	Change the name and make it shorter.
XXXXX---ON/OFF/STOP not found	Syntax error: A statement XXXXX requires ON, OFF, or STOP.	Specify ON/OFF/STOP.

App - 10

Error	Meaning	Corrective action
XXXXX---redimensioned array	Compiler restricted item: Array change XXXXX is declared twice.	Make the appropriate correction to the program so that the array is not redefined.
XXXXX---string variable expected	Illegal parameter: A string variable is required but a different data type is specified in a statement XXXXX.	Specify a string variable.
XXXXX---undefined function	Syntax error: An undefined FN function is called.	Define the function or specify a defined function.
XXXXX---undefined label	Syntax error: A non-existent label XXXXX is referenced by the GOTO/GOSUB instruction.	Change to a correct label.
XXXXX---undefined line number	Syntax error: A non-existent line number XXXXX is referenced by the GOTO/GOSUB instruction.	Change to a correct line number.
XXXXX---undefined variable	Compiler restricted item: An undefined variable is referenced.	Examine the program and correct the undefined variable. Define the array variable enclosed within the parentheses "( )" in advance. Or, replace it with 0 if no problem will arise.
XXXXX expected'X'expected	Syntax error: XXXXX or 'X' is required, but something else is specified.	Specify correctly.
XXXXX not supported	Compiler restricted item: XXXXX uses a statement/function that is not supported by the compiler.	Make the appropriate correction to the program so that the unsupported instruction/function is not used.

App - 11

(3) Warnings

Error	Meaning	Corrective action
STOP assumed to be END	The compiler compiles the STOP instruction assuming it is the END instruction.	_
XXXXX ignored	The compiler ignores statement/function XXXXX.	_

App - 12

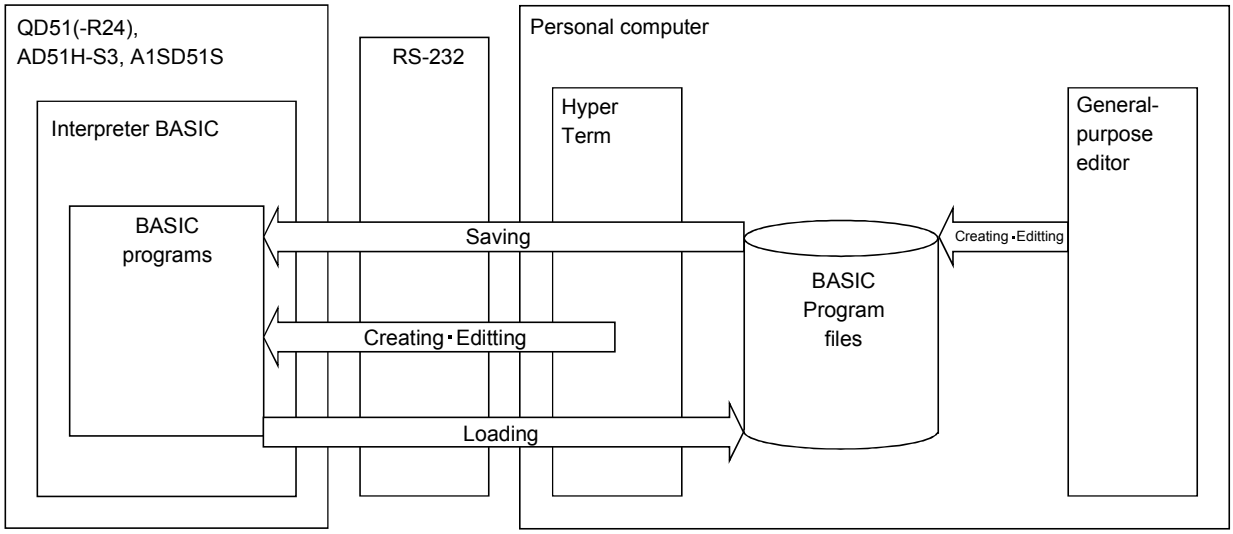
Appendix-3 Programming Operation Using HyperTerm of Microsoft Windows®

The following interpreter BASIC programming operations can be performed between the personal computer and the module using HyperTerm of Microsoft Windows® (Hyper Terminal) as a console of the AD51H-S3/A1SD51S/QD51(-R24).

(1) Saving BASIC programs between the personal computer and the module

(2) Loading BASIC programs between the personal computer and the module

(3) Creating and editing BASIC programs online ∗1



The following shows the functional comparison of HyperTerm with software package.

○: Available ×: Not available

Function		Software package	HyperTerm
Online programming	Saving of interpreter BASIC programs	○	○
	Loading of interpreter BASIC programs	○	○
	Creating and editing of interpreter BASIC programs	○	○∗1
	Access to the files on the PC side	○	×
	Output to the printer on the PC side	○	×
File maintenance	Directory display of hard disks, floppy disks, and memory cards	○	×
	File deletion of hard disks, floppy disks, and memory cards	○	×
	File copy of hard disks, floppy disks, and memory cards	○	×
	File verification of hard disks, floppy disks, and memory cards	○	×
	Format of hard disks, floppy disks, and memory cards	○	×
	Backup of memory cards	○	×
	Copy to ROM	○	×
	Setting data display	○	×
Date and time setting	Clock data setting of the PLC CPU	○	×
Compiling of BASIC programs (BASIC compiler)		○	×
Reassigning of line numbers of the file (Line numbering tool)		○	×

∗1: Available only with HyperTerm of Windows® XP.

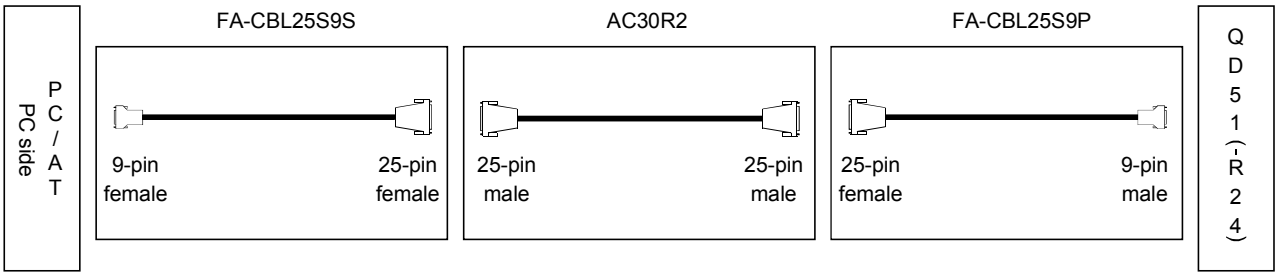
●: Operable ×: Not operable

Version of Windows® used to check operations	Windows® 98	Windows® ME	Windows® 2000	Windows® XP
	4.10.1998	4.90.3000	5.00.2195 Service Pack3	HOME Edition 2002
Version of HyperTerm	1.0.0.0	3.0.2000.0	5.0.2195.5305	5.1.2600.0
Transfer between the PC and the module (Saving)	●	●	●	●
Transfer between the PC and the module (Loading)	●	●	●	●
Basic screen operations (BASIC program creation and edit functions such as cursor key and scrolling operation)	●	×	●	●

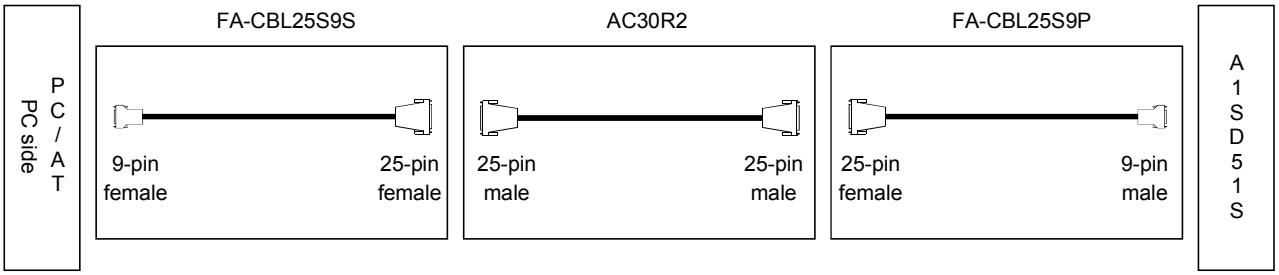
Appendix-3.1 Wiring

Cable wiring shall be as follows.

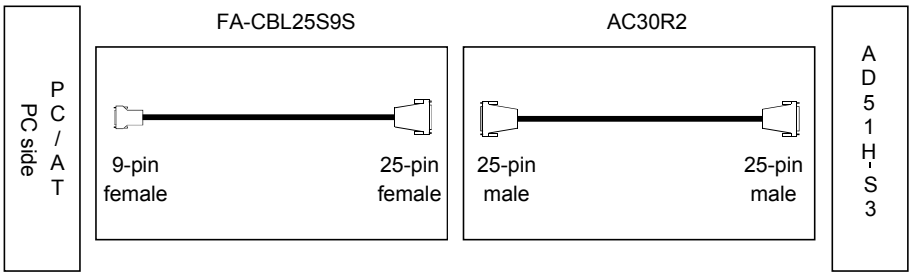
(1) For the QD51(-R24)



(2) For the A1SD51S



(3) For the AD51H-S3



App - 15

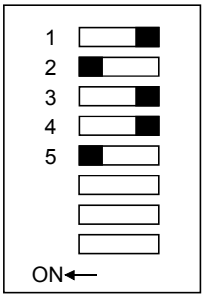
Appendix-3.2 Settings on the AD51H-S3/A1SD51S/QD51(-R24) side

Set the Console, debugger type to "VT".

(1) Settings shall be as follows when the console is connected to "CH1" on the QD51(-R24).

Switch setting for intelligent function module	Value	Remarks
Switch 1	0021H	Console: CH1 Software: Unused Console, debugger type: VT Ctrl+C setting: Enable
Switch 2	0004H	Programming mode
Switch 3 to 5	Unused	

(2) The console is connected to "VT-382" on the AD51H-S3/A1SD51S.
(For example, when the debugger is not used, mode setting switch 2 shall be as follows.)



Mode setting switch 1 is set to programming mode "4".

App - 16

Appendix-3.3 Settings on the HyperTerm side

This section explains the setting method with Windows® XP.

(1) Settings at HyperTerm start-up are as follows.

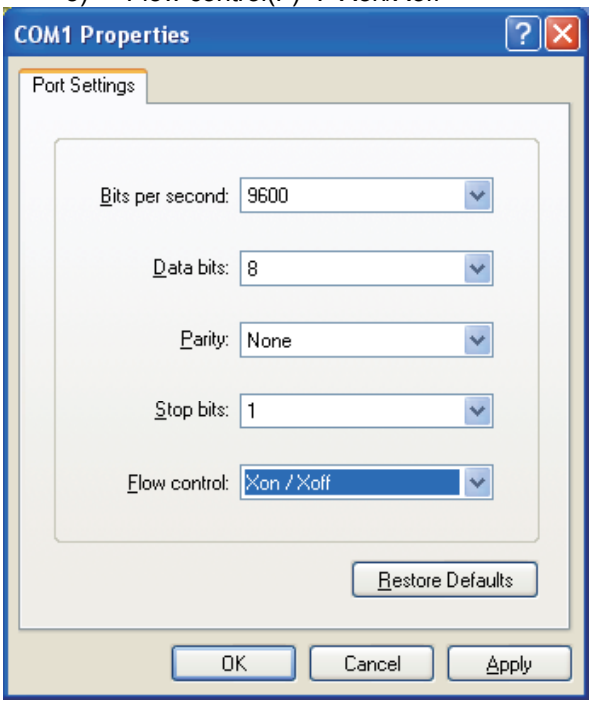
(a) "Connect To"

"Connect using(N) " : "Com?"

*When using COM1

(b) "COM? Properties" - "Port Setting"

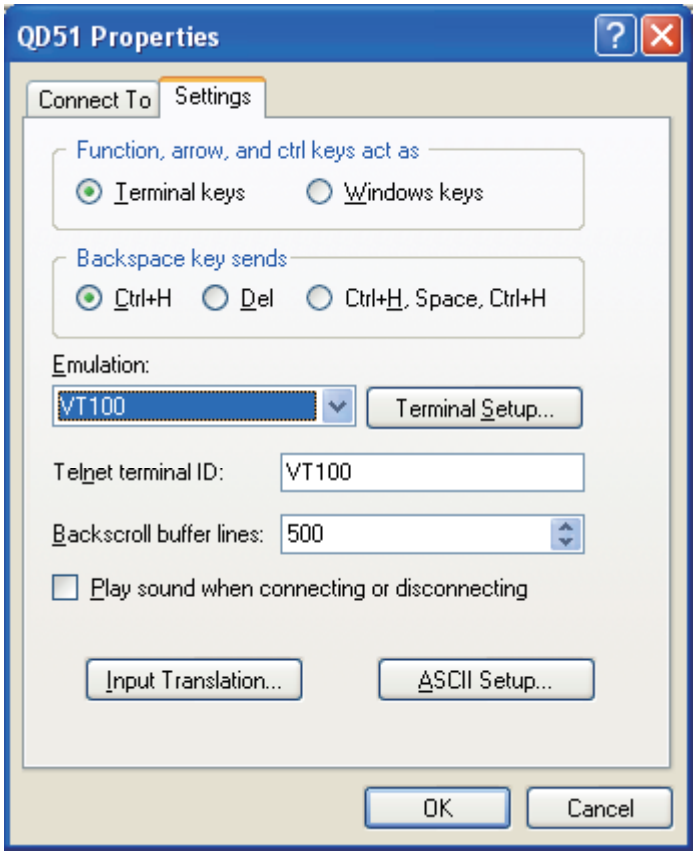
1) "Bits per second(B)" : "9600"
2) "Data bits(D)" : "8"
3) "Parity(P)" : "None"
4) "Stop bits(S)" : "1"
5) "Flow control(F)" : "Xon/Xoff"



(2) Select "File(F)" - "Properties(R)" from the menu and set the followings.

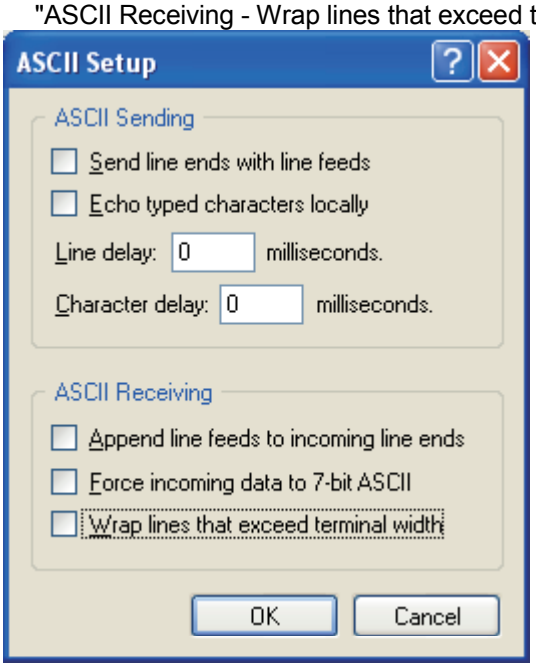
(a) "COM? Properties" - "Settings"

"Emulation(E)" : "VT100"



(b) "COM? Properties" - "Settings" - "ASCII Setup"

"ASCII Receiving - Wrap lines that exceed terminal width(W)": Unchecked



Appendix-3.4 How to save a BASIC program

Take the following steps to save a BASIC program file created with a general-purpose editor.

(1) Start interpreter BASIC of the saving-target task and set interpreter BASIC in the input status.

(2) Select 'Transfer(T)' - 'Send Text File(T)' from the HyperTerm menu. Upon selecting the sending-target BASIC program file, the corresponding file is send from HyperTerm and saved to the module. Wait until all the file contents are saved.

(3) Press Ctrl+W to redisplay the screen.

Appendix-3.5 About editing of BASIC programs

Editing of BASIC programs is possible with HyperTerm of Microsoft Windows® XP.

When editing BASIC programs with other operating systems, take the following steps.

(1) Delete a BASIC program with the NEW instruction, collect the contents of original BASIC program file with a general-purpose editor, and save it again.

Appendix-3.6 How to load a BASIC program

Take the following steps to load a BASIC program from the module to the file on the PC side.

(1) Execute the following instruction in direct mode by interpreter BASIC of the loading-target task.
WHILE INKEY$="":WEND:SAVE"COM0":WHILE 1:WEND

(2) Select 'Transfer(T)' - 'Capture Text(C)' from the HyperTerm menu. Then, start capturing after entering a file name.

(3) Press the space key or another key once to execute the SAVE instruction. The contents of the BASIC program are sent to HyperTerm as text. The sending contents are also displayed on the HyperTerm screen.

(4) Select 'Transfer(T)' - 'Capture Text(C)' - 'Stop(S)' from the HyperTerm menu upon confirming the completion of sending all contents on the screen.

(5) Press Ctrl+C to stop the instruction being executed.

(6) Press Ctrl+W to redisplay the screen.

Note that an EOF code (1aH) is added at the end of the BASIC program file loaded using capture. Delete the code using Notepad or other software.

MEMO

App - 20

WARRANTY

Please confirm the following product warranty details before using this product.

1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the sales representative or Mitsubishi Service Company.
However, if repairs are required onsite at domestic or overseas location, expenses to send an engineer will be solely at the customer's discretion. Mitsubishi shall not be held responsible for any re-commissioning, maintenance, or testing on-site that involves replacement of the failed module.

[Gratis Warranty Term]

The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place.
Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.

[Gratis Warranty Range]

(1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.
(2) Even within the gratis warranty term, repairs shall be charged for in the following cases.
1. Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
2. Failure caused by unapproved modifications, etc., to the product by the user.
3. When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
4. Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
5. Failure caused by external irresistible forces such as fires or abnormal voltages, and Failure caused by force majeure such as earthquakes, lightning, wind and water damage.
6. Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
7. Any other failure found not to be the responsibility of Mitsubishi or that admitted not to be so by the user.

2. Onerous repair term after discontinuation of production

(1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued. Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.
(2) Product supply (including repair parts) is not available after production is discontinued.

3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

4. Exclusion of loss in opportunity and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation of damages caused by any cause found not to be the responsibility of Mitsubishi, loss in opportunity, lost profits incurred to the user by Failures of Mitsubishi products, special damages and secondary damages whether foreseeable or not , compensation for accidents, and compensation for damages to products other than Mitsubishi products, replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

5. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

6. Product application

(1) In using the Mitsubishi MELSEC programmable controller, the usage conditions shall be that the application will not lead to a major accident even if any problem or fault should occur in the programmable controller device, and that backup and fail-safe functions are systematically provided outside of the device for any problem or fault.
(2) The Mitsubishi programmable controller has been designed and manufactured for applications in general industries, etc. Thus, applications in which the public could be affected such as in nuclear power plants and other power plants operated by respective power companies, and applications in which a special quality assurance system is required, such as for Railway companies or Public service purposes shall be excluded from the programmable controller applications.
In addition, applications in which human life or property that could be greatly affected, such as in aircraft, medical applications, incineration and fuel devices, manned transportation, equipment for recreation and amusement, and safety devices, shall also be excluded from the programmable controller range of applications.
However, in certain cases, some applications may be possible, providing the user consults their local Mitsubishi representative outlining the special requirements of the project, and providing that all parties concerned agree to the special circumstances, solely at the users discretion.

SH(NA)-080091-E(0612)MEE
MODEL: BASIC-P-DEBAG2-E
MODEL CODE: 13JF64

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , JAPAN