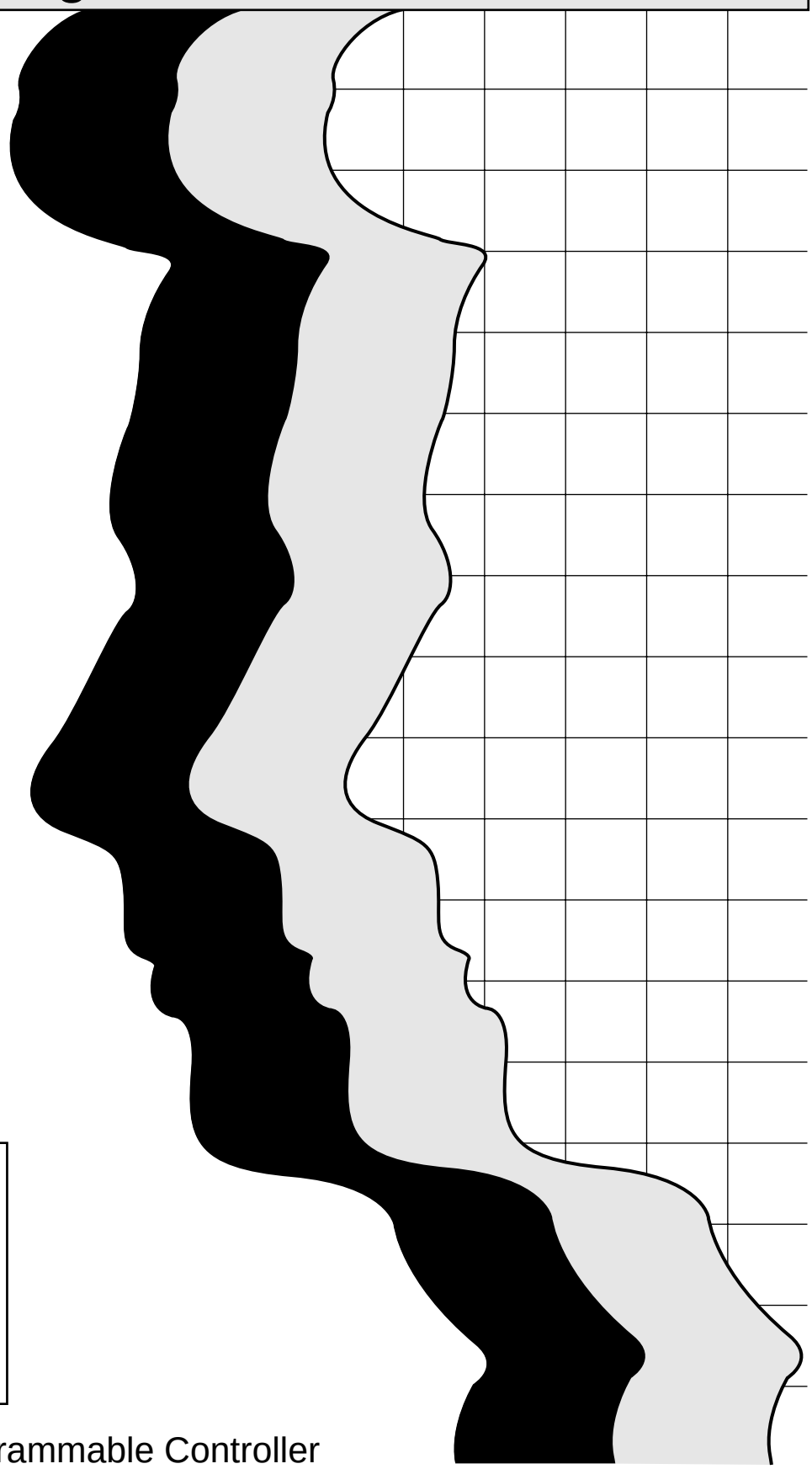
MITSUBISHI

type AD51H-BASIC (Program edit, Compile)

Programming Manual



A

MELSEC

Mitsubishi Programmable Controller

REVISIONS

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

Print Date	*Manual Number	Revision
Jun., 1995	IB (NA) 66568-A	First edition

INTRODUCTION

Thank you for choosing the Mitsubishi MELSEC-A Series of General Purpose Programmable Controllers. Please read this manual carefully so that the equipment is used to its optimum. A copy of this manual should be forwarded to the end User.

CONTENTS

Manuals

The following manuals are also relevant to the AD51H communication module.

Related Manuals

Manual Name	Manual Number
AD51H-S3 Intelligent Communication Module User's Manual Describes the system configuration when using the AD51H-S3 module, the module specifications, part nomenclature and settings, functions, and outside dimensions. (Packaged with the module)	IB-66401
A1SD51S Intelligent Communication Module User's Manual (Hardware) Describes the system configuration when using the A1SD51S module, the module specifications, part nomenclature and settings, functions, and outside dimensions. (Packaged with the module)	IB-66550
A1SD51S Intelligent Communication Module User's Manual (Detailed Information) Describes the system configuration when using the A1SD51S module, the module specifications, part nomenclature and settings, functions, and outside dimensions. (Sold separately)	IB-66551
AD51H-BASIC Programming Manual (Command) Describes the AD51H programming method, commands, error codes, etc. (Sold separately)	IB-66567
SW0IX-AD51HP Operating Manual Describes how to use the software package for the IBM PC/AT. (Packaged with the software package)	IB-66402

1. GENERAL DESCRIPTION

This programming manual describes the system and debugging commands, and compiling method, used with the AD51H communication module.

(1) System/debugging commands

The following operations can be executed by inputting commands from a console/debugger.

- BASIC program editing/debugging
- Writing BASIC programs to, and reading them from, memory cards, FDs, and HDs.
- BASIC program execution, stop, status display
- General-purpose I/O, reading/writing to internal devices
- Changing/reading multitasking settings

(2) Creation of a BASIC program with a general-purpose editor (PC/AT only)

BASIC programs can be created online by using one of the general-purpose editors available on the market.
By using the line number tool, the program created with the general-purpose editor can then be assigned line numbers.

(3) Compilation of a BASIC program (PC/AT only)

BASIC programs created using interpreter BASIC can be compiled using a compiler.
This gives an execution speed approximately 3 to 4 times faster than interpreter BASIC.

(4) Storage of the AD51H-S3 BASIC programs in a ROM (PC/AT only)

Created BASIC programs can be written to a ROM for use by an AD51H-S3.

2. STARTING UP THE COMMUNICATION MODULE AND MODE CHANGE

This section gives how the communication module mode changes (when the communication module starts up and after the comunication module starts up).

This development is related to the online programming (see Section 4) and the multitask debugging (see Section 5).

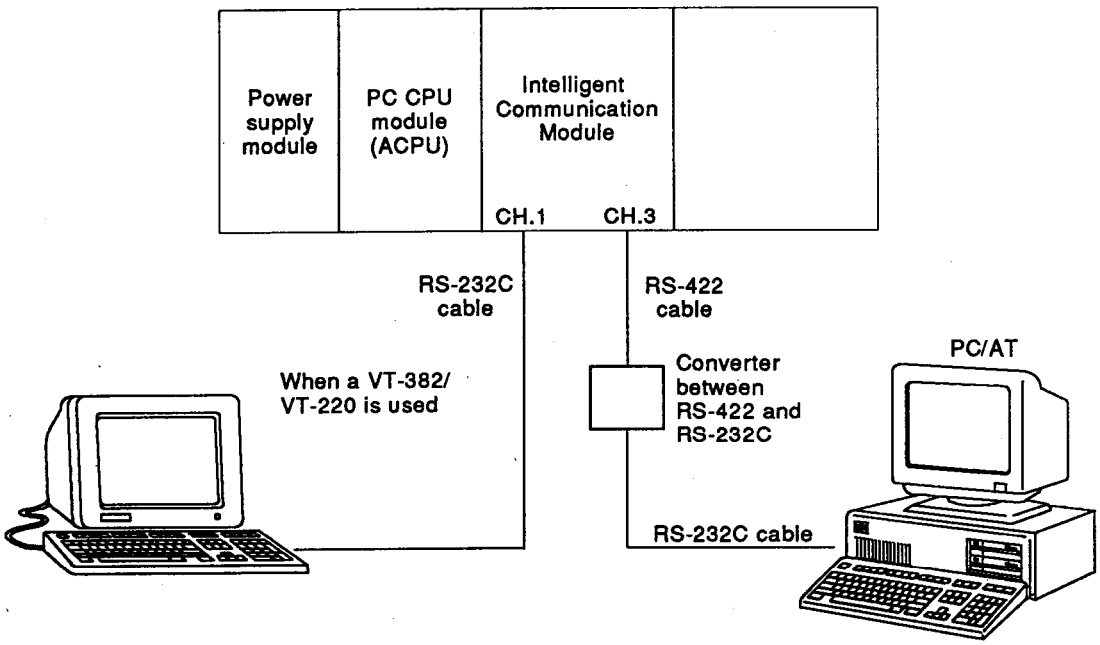
2.1 Using a PC/AT and a General-Purpose Terminal as the Console (Display Terminal) and the Debugger

This section describes how to start up the communication module when a PC/AT and a General-Purpose Terminal as the Console and the debugger.

Use either the PC/AT (connected to the communication module) or the general-purpose terminal (connected to the communication module) as the console, and the other as the debugger.

The switch settings (communication module mode setting switches SW1 to SW5) determine which machines are used as the console and the debugger. (See Section 2.1.1.)

(When a building-block type CPU module is used)



2.1.1 Preparations required to start up the communication module

This shows the preparations required before the system starts up.

For details, see the operating manuals of the used devices.

(1) Communication module

(a) Setting the communication module

Set the switches used for operating the communication module .

On how to set the switches and how to use the switches, refer to AD51H-S3 Intelligent communication module User's Manual and A1SD51S Intelligent communication module User's Manual.

The following switches must be set according to the operation mode when the communication module starts up and to how to use the PC/AT and general-purpose terminal.

1) Mode setting switch 1

Set this switch to 0 to 4 according to the communication module operation mode.

[0]/[1] : Used when executing a BASIC program that has already been debugged after the communication module starts up.

[2]/[3] : Used when the multitask debugging (see Section 5) is done after the communication module starts up. (Can be set when the required BASIC program editing and debugging, and multitask setting, have been completed.

[4] : Used when editing a BASIC program, debugging a single program, or operating in the system mode (see Section 4) after the communication module starts up.

2) Mode setting switch 2

Set the switches SW1 to SW5 so that they match the machine used as the console, the machine used as the debugger, and the used interfaces.

(Example)

Set the switches as show below when using a PC as the console, a VT-382/VT-220 (connected to CH.1 of the communication module) as the debugger:

No.	1	2	3	4	5
Setting	OFF	OFF	OFF	ON	ON

(b) Installing memory cards (AD51H-S3 only)

To store execution programs and data, install necessary memory cards in the AD51H-S3. (Up to two cards can be installed.)

To install the cards, connect a battery beforehand when necessary.

When a memory card with a write-protect tab is used, make sure that the protect is released when a BASIC program is written after the AD51H-S3 starts up or the memory card is newly used.
On how to install a memory card and how to connect a battery,refer to AD51H-3 Intelligent communication module User's Manual.

(c) Loading the communication module into a base unit

After setting the communication module and installing memory cards, load the communication module into a slot of the base unit.

(2) PC/AT

(a) Installing the software

Install the following the software packages in the PC/AT.

- Operating system : MS-DOS (Ver 3.21 or after)
- AD51H-BASIC software package: (SW1IX-AD51HPE)

(b) Connecting the communication module to the PC/AT

Use a AC30R4 cable to connect the communication module RS-422 interface (CH.3) to the PC/AT serial interface.

Connect (COM 1) using an converter between the RS-422 and RS-232C interfaces.

(3) General-purpose terminal

(a) When a VG-620 is used (*1)

1) Set the VG-620 USART mode as shown below:

- Baud rate : 9600 bps
- Data length : 8-bit
- Stop bit : 2-bit
- Parity : None

2) Connecting the communication module to the VG-620

Use an AC30R2 cable connect the communication module RS-232C interface (CH.1/CH.2) to the VG-620 RS-232C interface.

- When the general-purpose terminal is used as the console: CH.1
- When the general-purpose terminal is used as the debugger: CH.1 or 2

(Set the utilized interfaces so that they are consistent with the switches SW1 to SW5 of the communication module mode setting switch 2.)

REMARK

*1: The manual of the VG-620 gives information about how to set and connect the VG-620.

(b) When a VT-382/VT-220 is used (*1)

1) Set the VT-382 USART mode as shown below:

- Baud rate : 9600 bps
- Data length : 8-bit
- Stop bit : 2-bit
- Parity : None

2) Connecting the communication module to the VT-382/VT-220

Use an AC30R2 cable to connect the communication module RS-232C interface (CH.1/CH.2) to the VT-382/VT-220 RS-232C interface.

- When the VT382/VT-220 is used as the console: CH.1
- When the VT382/VT-220 is used as the debugger: CH.1 or 2

(Set the utilized interfaces so that they are consistent with the switches SW1 to SW5 of the communication module mode setting switch 2.)

REMARK

*1: The manual of the VT-382/VT-220 gives information about how to set and connect the VT-382/VT-220.

2.1.2 Starting up the communication module

This section describes how to start up the communication module used with a PC/AT and a general-purpose terminal.

(1) Starting up the PC/AT and the general-purpose terminal

The following gives how to start up the PC/AT and the general-purpose terminal.

(a) When starting up the PC/AT

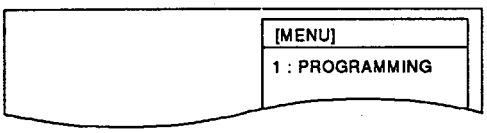
1) Power ON

Turn ON the power to the PC/AT.

2) Starting the SW1IX-AD51HPE installed in the PC/AT

Input "D51HBASE", and press the [↵] key. Then, the SW1IX-AD51HPE starts.

- The SW1IX-AD51HPE main menu appears on the screen of the PC/AT.



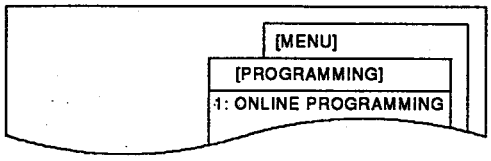
POINT

The SW1IX-A51HPE Operating Manual gives details about how to execute the following operations:

- Starting up the SW1IX-AD51HPE
- Setting the operating conditions for the PC/AT
- Setting the PC/AT to the online programming mode

3) Changing the PC/AT mode (to the online programming mode)

i) When the PC/AT screen displays the SW1IX-AD51HPE main menu, select the PROGRAMMING from the menu.



ii) Select ONLINE PROGRAMMING from the menu.

(b) When starting up the general-purpose terminal

Turn ON the power to the general-purpose terminal.

(2) Starting up the communication module

Turn ON the power to the communication module

According to the switches SW1 to SW5 of the communication module mode setting switches 1 and 2, a prompt screen appears on the PC/AT and the general-purpose terminal.

Communication Module Mode Setting Switch (1)	Communication Module Mode	Display on the Console	Display on the Terminal (used for debugging)
		(Depends on the Mode Setting Switch 2 Setting (SW1 to SW5))	
0 or 1	Execution mode (2)	Displays the data output from a BASIC program.	Displays the data output from a BASIC program.
2 or 3	Debug mode	Displays the data output from a BASIC program.	D>
4	System mode	S>	Displays the data output from a BASIC program.

(3) Starting the online programming operation/multitask debugging operation

(a) When the communication module is in the debug mode

1) Execute the debugging operation in the multitasking system according to Section 5.

2) When changing the communication module mode and continuing the operation, see Section 2.3.

(b) When the communication module is in the system mode

1) Execute operations in the system mode according to Section 4.

2) To edit and debug a BASIC program, use the START command to return the communication module to the editing mode (1), and execute the operation.

The AD51H-BASIC Programming Manual gives how to edit and debug a BASIC program.

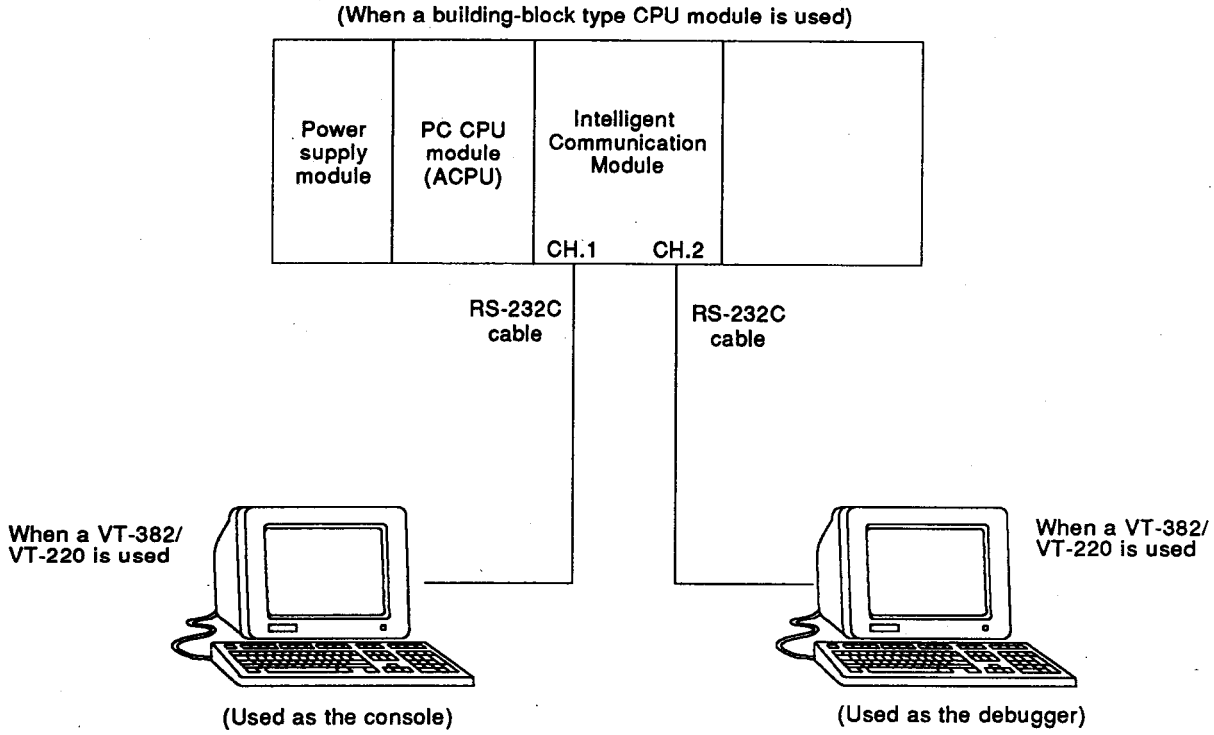
3) When changing the communication module mode and continuing the operation, see Section 2.3.

2.2 Startup when Using Two General-Purpose Terminals as the Console and the Debugger

This section gives the communication module startup procedure when using two general-purpose terminals as the console and the debugger.

When two general-purpose terminals are used:

- CH.1 corresponds to the console;
- CH.2 corresponds to the debugger.



2.2.1 Preparations required to start up the communication module

This shows the preparations required before the system starts up.

For details, see the operating manuals of the used devices.

(1) Communication module

(a) Setting the communication module

Set the switches used for operating the communication module.

On how to set the switches and how to use the switches,refer to AD51H-S3 Intelligent communication module User's Manual and A1SD51S Intelligent communication module User's Manual.

The following switches must be set according to the operation mode when the communication module starts up and to how to use the general-purpose terminals.

(Section 2.1.1 (1)-(a) gives general information.)

1) Mode setting switch 1

2) Mode setting switch 2

(Example)

Set the switches as show below when using two general-purpose terminals VT-382/VT-220 (connected to CH.1 and 2 of the communication module respectively) as the console and the debugger:

No.	1	2	3	4	5
Setting	OFF	ON	ON	OFF	ON

(b) Installing memory cards (AD51H-S3 only)

To store execution programs and data, install necessary memory cards in the AD51H-S3. (Up to two cards can be installed.)

To install the cards, connect a battery beforehand when necessary.

When a memory card with a write-protect tab is used, make sure that the protect is released when a BASIC program is written after the AD51H-S3 starts up or the memory card is newly used.

On how to install a memory card and how to connect a battery, refer to AD51H-S3 Intelligent communication module User's Manual.

(c) Loading the communication module into a base unit

After setting the communication module and installing memory cards, load the communication module into a slot of the base unit.

(2) General-purpose terminal

(a) When a VG-620 is used

When a VG-620 is used as the console or debugger for the communication module, set and connect the VG-620 according to Section 2.1.1 (3)-(a).

(b) When a VT-382/VT-220 is used

When a VT-382/VT-220 is used as the console or debugger for the communication module, set and connect the VT-382/VT-220 according to Section 2.1.1 (3)-(b).

2.2.2 Starting up the communication module

This section describes how to start up the communication module used with two general-purpose terminals.

(1) Power ON

(a) Turn ON the power to the general-purpose terminals.

(b) Then, turn ON the power to the communication module.

(2) Starting the online programming operation/multitask debugging operation

After the communication module starts up, according to the switches SW1 to SW5 of the communication module mode setting switches 1 and 2, a prompt screen appears on the PC/AT and the general-purpose terminal.
The contents of this screen are as indicated in Section 2.1.2 (2).

(a) When the communication module is in the debug mode

1) Execute the debugging operation in the multitasking system according to Section 5.

2) When changing the communication module mode and continuing the operation, see Section 2.3.

(b) When the communication module is in the system mode

1) Execute operations in the system mode according to Section 12.

2) To edit and debug a BASIC program, use the START command to return the communication module to the editing mode (1), and execute the operation.

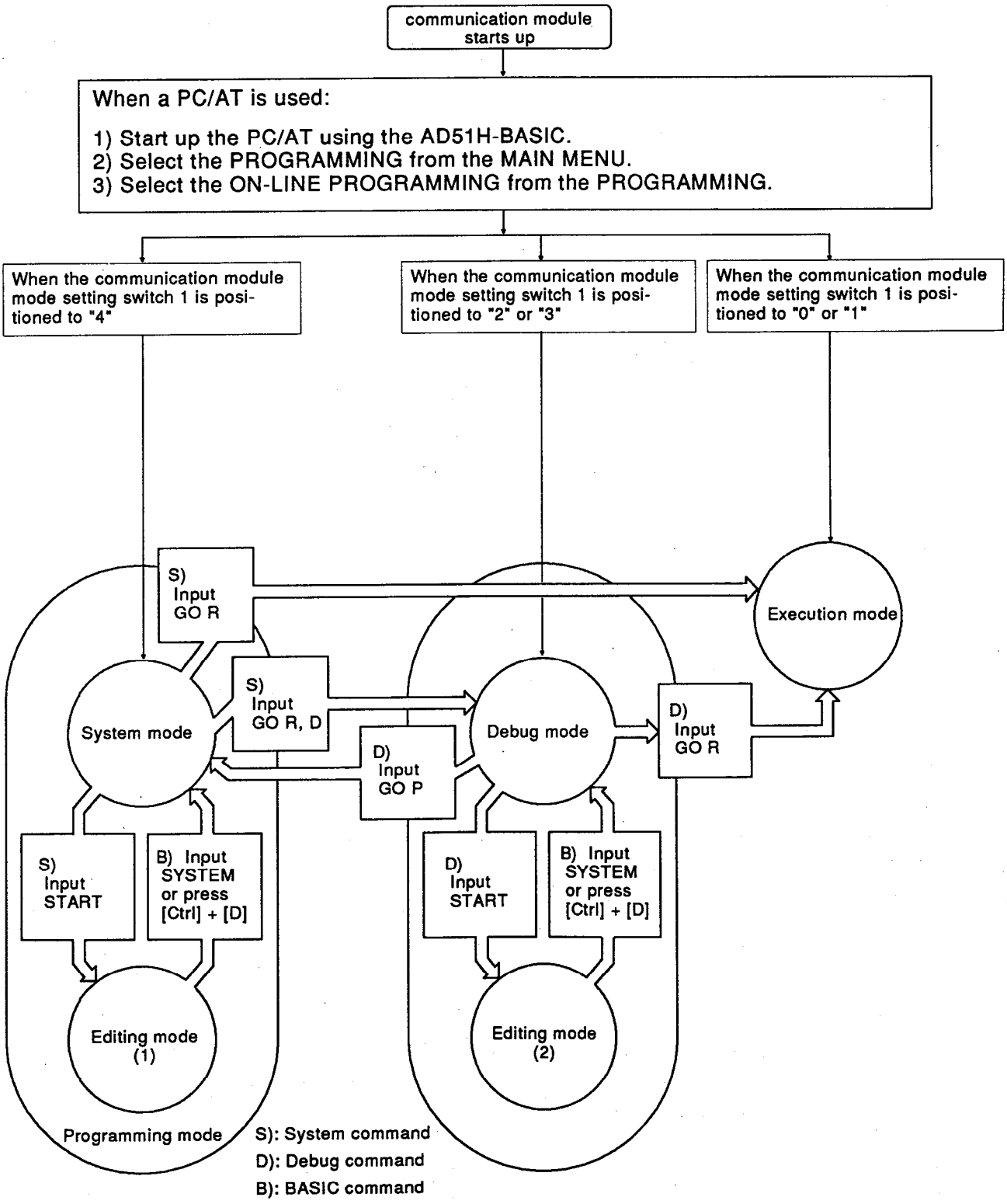
The AD51H-BASIC Programming Manual gives how to edit and debug a BASIC program.

3) When changing the communication module mode and continuing the operation, see Section 2.3.

2.3 Communication Module Mode and Mode Change

After the communication module starts up, the mode can be changed giving the system command from the console (see Section 4) or giving the debug command from the debugger (see Section 5).

This section gives the outline of how to change the communication module mode with the system command and the debug command.



(1) Programming modes

(a) In the modes, it is possible to edit/debug a BASIC program, write/read data to/from a memory card, and to set a multitask.

(b) The programming modes are divided into the system mode and the editing mode (1).

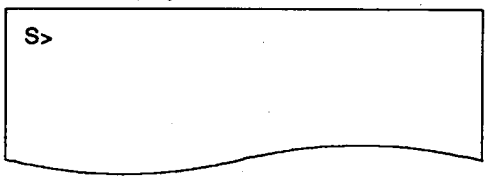
(2) System mode

(a) The communication module enters this mode when the communication module mode setting switch is positioned to "4" or when the GO command (GO P) is input from the debugger in the debug mode.

(b) The console is controlled by the operating system in the communication module.

(c) By giving the system command, the following operations for each BASIC program can be executed:

Console display



- Writes/reads a BASIC program to/from the execution program area of the memory card (installed to MEMORY CARD [1] of AD51H-S3) and the EEP-ROM of A1SD51S.
- Sets the multitask.

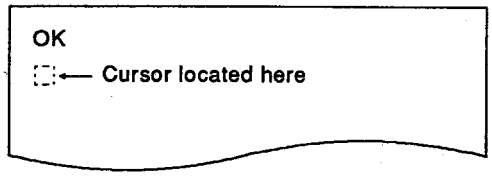
(3) Editing mode (1)

(a) The communication module enters this mode when the START command is input using the console (in the system mode).

(b) The interpreter (an operating system to analyze and execute BASIC commands) uses the console.

(c) By giving a AD51H-BASIC command or function, the following operations for each BASIC program.

Console display



- Editing and debugging
- Writes/reads a BASIC program to/from the file storage area of a memory card.

(4) Execution mode

(a) The communication module enters this mode when the communication module mode setting switch 1 is positioned to "0" or "1" or when the GO command is input from the console or debugger.

(The communication module enters this mode when the RUN keyswitch/RUN switch is positioned to RUN.)

(b) Set the multitask to execute several BASIC programs in the normal mode.

(5) Debug mode

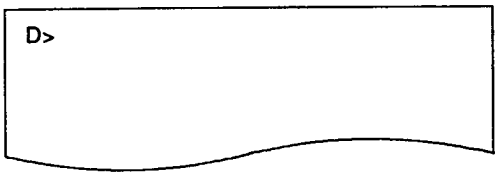
(a) The communication module enters this mode when the communication module mode setting switch 1 is positioned to "2" or "3" or when the GO command is input using the console.

(The communication module enters this mode when the RUN keyswitch/RUN switch is positioned to RUN.)

(b) The communication module debug (an operating system to analyze and execute debug commands) uses the debugger.

(c) By giving a debug command input from the debugger, it is possible to debug each BASIC program in the multitask system.

Terminal used as the debugger



- Controls a specified BASIC program.
- Inputs and outputs data to a memory or devices that can be accessed using a BASIC program.
- Changes the communication module mode.

(6) Editing mode (2)

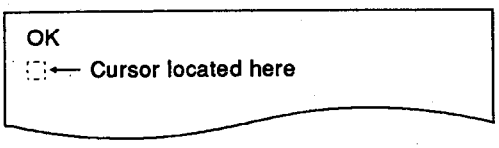
(a) The communication module enters this mode when the START command is input from the debugger

(Another task than specified using the START command continues in the multitask system.)

(b) The debugger is controlled by the interpreter.

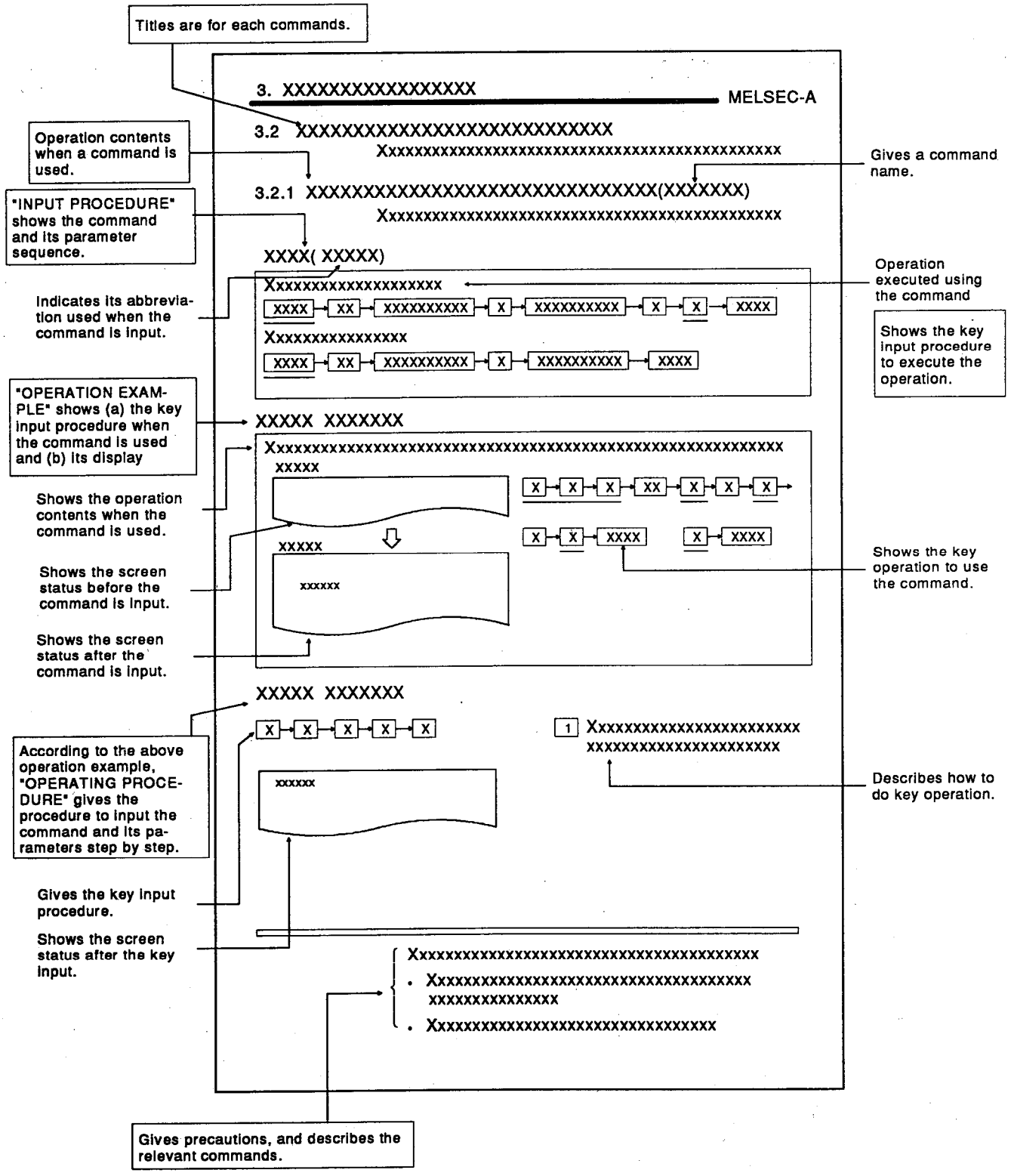
(c) By giving an AD51H-BASIC command or function, it is possible to correct a BASIC program when another BASIC program is being edited.

Terminal used as the debugger



3. COMMAND EXPLANATION FORMAT

This manual gives explanations about the commands as shown below:



4. ONLINE PROGRAMMING

Online programming refers to such operations as editing/debugging the BASIC program, and writing/reading the BASIC program to a memory card, FD, or the PC/AT HD.

(Only the BASIC programs in a task can be debugged online.)

This section tells how to use system commands when (a) editing/debugging the BASIC program, and (b) writing/reading the BASIC program to/from a memory card, FD, or the PC/AT HD using the console in the system mode.

(1) Since most of this section concerns key inputting and displays on the console, the explanations assume that all key inputting and displays refer to the console.
When key inputting and displays refer to the debugger rather than the console, the word "debugger" is always used to avoid misunderstanding.

(2) Executing the operations discussed in this section requires the following preparations (see Section 2 for details):

- Setting the AD51H switch to program online..... See Section 2
- Connecting the console ..... See Section 2

4.1 System Commands

Table 4.1 lists the system commands (input from the keyboard to the console) used for online programming.

Table 4.1 List of System Commands

Classification	System Command	Function	Reference Section	Module Availability	
				AD51H-S3	A1SD51S
Memory card control	CCOPY	Makes a copy of data from a memory card, and writes the copy to another memory card. (Sets a backup memory card.)	4.2.1	o	x
	CFORMAT	Formats a memory card (physical format).	4.2.2		
	CFORMAT?	Displays information about memory card formatting.	4.2.2		
	CRECOVER	Recovers an unusable area in the file area of a memory card.	4.9		
	FFORMAT	Formats the file area in a memory card (logical format).	4.10		
Execution program information control	MLOAD*1	Reads data from a designated BASIC task area of the memory card/EEP-ROM using the corresponding BASIC task area in the communication module.	4.3.1	o	o
	MSAVE	Writes data in a designated BASIC task area in the communication module to the corresponding BASIC task area (multitask setting is automatically done).	4.3.2		
Multitask setting control	SET	Changes the multitask setting description.	4.4.1	o	o
	SET?	Displays the multitask setting description.	4.4.2		
Mode control	START*1	Switches the communication module from the system mode to the editing mode (1). (For editing/debugging each program)	4.5.1	o	o
	GO	Switches the communication module from the system mode to the execution mode (2) or debug mode.	4.5.2		
Interpreter operation control	TKILL*1	Stops the interpreter operation in a designated BASIC task area in the communication module.	4.6	o	o
Others	EXIT	Displays the MAIN MENU screen on the console.	4.7	o	o
	HELP	Displays a list of system commands, descriptions of functions, and command input formats.	4.8		

o: Available x: Unavailable

*1 Cannot be extecuted with tasks that contain compiled BASIC programs.

4.2 Copying/Deleting Data From a Memory Card

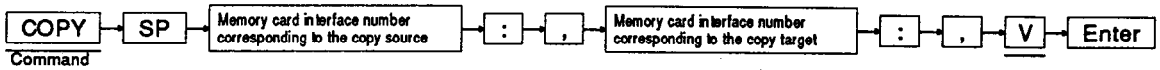
This section tells how to use system commands to copy/delete data from a memory card.

4.2.1 Copying data from a memory card and writing that data to another memory card (CCOPY command)

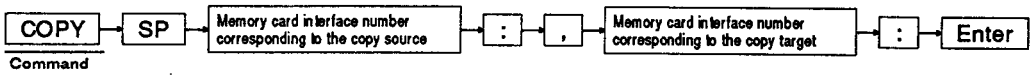
This operation backs up data by writing it to another memory card.

INPUT PROCEDURE (This command is also referred to as "CC")

To use the copy source to verify that correct copying has been done:



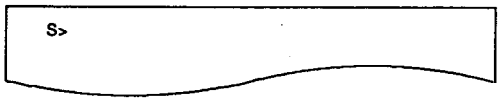
When only making a copy:



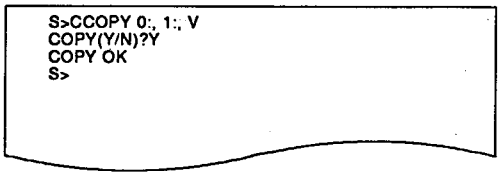
OPERATION EXAMPLE

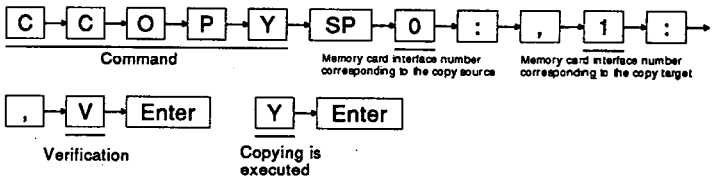
Writes a copy of data from the memory card installed in MEMORY CARD[1] to the memory card installed in MEMORY CARD[2], and verifies that copying has been done correctly.

Before the command is input

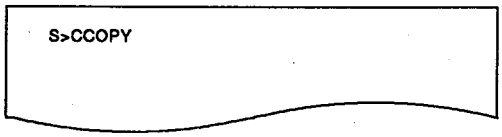


After the command is input





OPERATING PROCEDURE



1 Input the CCOPY command to write a copy of data in one memory card to another memory card.

(1) Precautions when using the CCOPY command
- Before copying, use the CFORMAT command to format the memory card in the copy target drive.
- Make sure that the copy source memory card capacity ≤ copy target memory card capacity.

SP → 0 → : → , → 1 → :

```
S>CCOPY 0:, 1:
```

2 Input the copy source memory card interface number (accompanied by a colon), followed by the copy target memory card interface number (accompanied by a colon and a comma). The numbers that can be input are "0" and "1".

0: Corresponds to the AD51H-S3 MEMORY CARD[1]
1: Corresponds to the AD51H-S3 MEMORY CARD[2]

(This example assumes that a copy of data in the memory card installed in MEMORY CARD[1] is written to the memory card installed in MEMORY CARD[2].)

, → V → Enter

```
S>CCOPY 0:, 1:, V
```

3 Designate "V" to verify that data has been correctly copied from the copy source to the copy target.

If verification is not executed, press the [Enter] key.

(This example assumes that verification is executed.)

Y → Enter

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

4 The "COPY (Y/N)?" dialog box appears.

Press the [Y] key to execute copying.

Press the [N] key to cancel the copy operation.

(The console remains in a wait state until either key is pressed.)

5 The next line shows the result of the execution.

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

When copying is executed normally, the screen shows "COPY OK".

If copying is not executed normally, an error message appears.

(This example assumes that copying is executed normally.)

6 "S>" appears after the execution result is displayed.

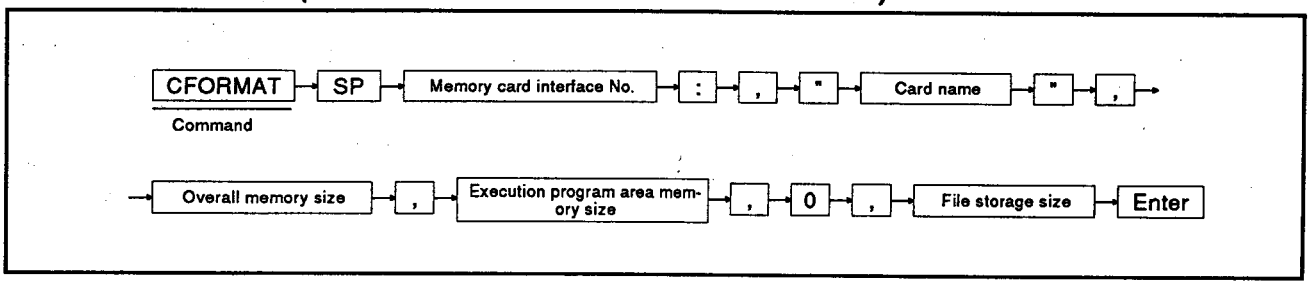
Input the necessary command.

(2) References
- Formatting a memory card: CFORMAT command..... (see Section 4.2.2)
- Displaying memory card format information: CFORMAT? command .... (see Section 4.2.3)

4.2.2 Formatting a memory card (CFORMAT command)

This operation formats a memory card installed in MEMORY CARD[1] or MEMORY CARD[2].

INPUT PROCEDURE (This command is also referred to as "CF")



OPERATION EXAMPLE

Format a 512K byte memory card installed in MEMORY CARD1 as shown below:

Before the command is input

```
S>
```

After the command is input

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

C F O R M A T SP 0 : , (Command; Memory card interface No.)

" T A S K - D T M " . (Card name)

Memory card is named

8 , (Overall memory size) ..... Sets the memory card capacity (for 8 x 64K bytes)

6 , (Execution area memory) ..... Sets the capacity of the execution program area (for 6 x 64K bytes) in the memory card

0 ,

2 Enter (File storage size) ..... Sets the capacity of the file storage area (for 2 x 64K bytes) in the memory card

Y Enter (Formatting is executed)

OPERATING PROCEDURE

C F O R M A T

1 Input the CFORMAT command to format a memory card.

```
S>CFORMAT
```

(1) Precautions when using the CFORMAT command
- Formatting a memory card deletes all data in that memory card.
- Release the write-protect tab when formatting a memory card which has a write-protect tab.
- Turn OFF the memory protect keyswitch on the AD51H-S3 when the memory card is installed in MEMORY CARD[1].

SP → 0 → : → ,

```
S>CFORMAT 0:,
```

2 Input the interface number (which corresponds to the memory card to be formatted) accompanied by a colon.

The numbers that can be input are "0" and "1".

0: Corresponds to the AD51H-S3 MEMORY CARD[1]
1: Corresponds to the AD51H-S3 MEMORY CARD[2]

This example assumes that the memory card installed in MEMORY CARD[1] will be formatted.

" → T → A → S → K → - →
D → T → M → " → ,

```
S>CFORMAT 0:, "TASK-DTM",
```

3 Input a name for the memory card using up to 16 alpha-numeric characters and symbols.

This name must start with an alphabetic character. The name must be in quotation marks.

This example assumes that the memory card is named "TASK-DTM".

8 → ,

```
S>CFORMAT 0:, "TASK-DTM", 8
```

4 Designate the overall capacity of the memory card using a number equal to or greater than one (in units of 64K bytes).

Set the capacity which is consistent with the following formula:

Overall memory size (Overall capacity) = (Execution program area memory size + File storage size)

This example assumes that a 512K byte memory card is formatted.

(8 x 64K bytes → 512K bytes)

6 → ,

```
S>CFORMAT 0:, "TASK-DTM", 8, 6,
```

5 Designate the capacity of the execution program area in the memory card by using a number from 0 to 6 (in units of 64K bytes).

The maximum capacity of the execution area is 384K bytes. This area is allocated to the operating system area (128 bytes) and the BASIC task number area (where the execution programs is stored).

(This example assumes that the execution program area capacity is set to 384K bytes.)

0 → ,

```
S>CFORMAT 0:, "TASK-DTM", 8, 6, 0,
```

6 Input "0" as dummy data.

2 → Enter

```
S>CFORMAT 0:, "TASK-DTM", 8, 6, 0, 2
```

7 Designate the capacity of the file storage area in the memory card using a number equal to or greater than one (in units of 64K bytes).

This area stores BASIC programs (that are not stored in the BASIC task area) and data files.

(This example assumes that the file storage capacity is set to 128K bytes.)

(2 x 64 bytes → 128K bytes)

Y → Enter

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

8 The "FORMAT (Y/N)?" dialog box appears.

Press the [Y] key to execute formatting.

Press the [N] key to cancel the format operation.

(The console remains in a wait state until either key is pressed.)

(This example assumes that formatting is executed.)

9 The next line shows the result of the execution.

When formatting is executed normally, the screen shows "FORMAT OK".

If formatting is not executed normally, an error message appears.

(This example assumes that formatting is executed normally.)

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

10 "S>" appears after the execution result is displayed.

Input the necessary command.

(2) Precautions when using the CFORMAT command to designate capacities
- The overall memory size (overall capacity) must be consistent with the memory card to be formatted.
  The overall memory size must be equal to the execution program area memory size plus the file storage size.
- In the execution program area of a memory card, if the entire BASIC task number area is divided into eight parts, the maximum capacity of one part is approximately 48K bytes.
- The size can be designated in decimal, hexadecimal ("&H[ ][ ][ ][ ]"), or binary ("&B[ ][ ] to [ ]").

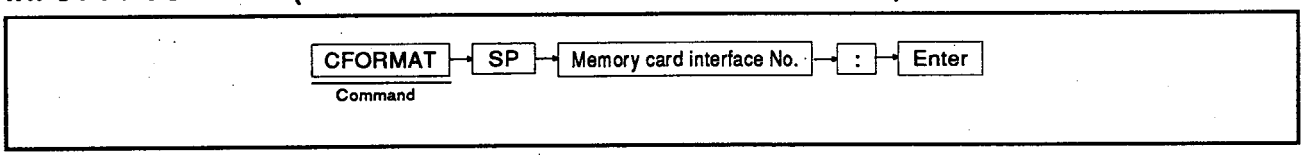
(3) Logically formatting a memory card
- The execution area is logically formatted when the SET or MSAVE command is initially used.
- Logically format the file storage area using the FFORMAT command.

(4) Reference
- Displaying memory card format information...CFORMAT? command (see Section 4.2.3)

4.2.3 Displaying memory card format information (CFORMAT? command)

This operation displays memory card formatting information installed in MEMORY CARD[1] or MEMORY CARD[2] of the AD51H-S3.

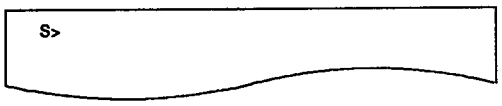
INPUT PROCEDURE (This command is also referred to as "CF?")



OPERATION EXAMPLE

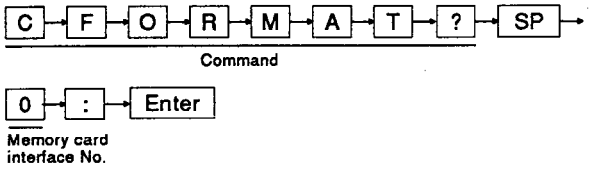
Displays format information from the memory card installed in MEMORY CARD[1].

Before the command is input

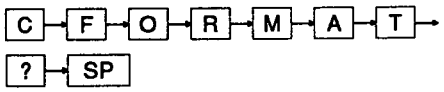


After the command is input

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

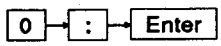


OPERATING PROCEDURE




```
S>CFORMAT ?
```

1 Input the CFORMAT? command to display memory card format information.




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

2 Input the interface number (corresponding to the memory card whose format information is displayed) accompanied by a colon.

The numbers that can be input are "0" and "1".

0: Corresponds to the AD51H-S3 MEMORY CARD[1]
1: Corresponds to the AD51H-S3 MEMORY CARD[2]

If the [Enter] key is pressed without being designated, the result is the same as if "0" were designated.

(This example assumes that format information from the memory card installed in MEMORY CARD[1] is displayed.)

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

3 Displays the command execution result.

When formatting is executed normally, the next lines show format information from the designated memory card.

If formatting is not executed normally, an error message appears.

When formatting is executed normally, the display on the left is shown.

(1)	Card Name	.....	Name of the memory card that is formatted.
(2)	Card Size	.....	Corresponds to the capacity of the whole memory card designated when the memory card was formatted. ( ) indicates the entire memory size designation when the memory card is formatted using the CFORMAT command.
(3)	Program Size	.....	Corresponds to the capacity of the execution program area when the memory card is formatted. ( ) indicates the capacity designation when the memory card is formatted using the CFORMAT command.
(4)	Canvas Size	.....	Should be ignored.
(5)	File Size	.....	Corresponds to the capacity of the file storage area designated when the memory card was formatted. ( ) indicates the capacity designation when the memory card was formatted using the CFORMAT command.

4 "S>" appears after the execution result is displayed.

Input the necessary command.

(1) Reference
- Formatting a memory card ..... CFORMAT command (see Section 4.2.2)

4.3 Writing/Reading an Execution Program

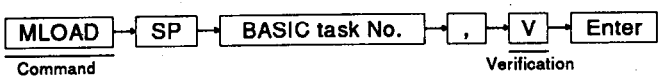
This section tells how to (a) write an execution program (stored in a communication module BASIC task number area) to a memory card/EEP-ROM, and (b) read an execution program from the memory card/EEP-ROM using a BASIC task number area. Both are done by using system commands (to control execution program information).

4.3.1 Reading an execution program stored in a memory card/EEP-ROM using the communication module (MLOAD command)

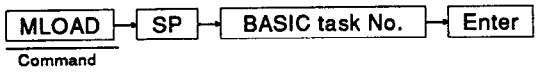
This operation reads an execution program stored in a memory card/EEP-ROM using the corresponding BASIC task number area in the communication module.

INPUT PROCEDURE (This command is also referred to as "ML")

To verify the read data with the source data after reading the execution program:



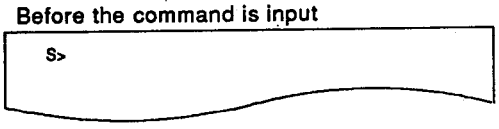
When only reading:



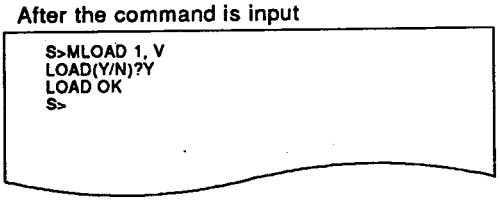
OPERATION EXAMPLE

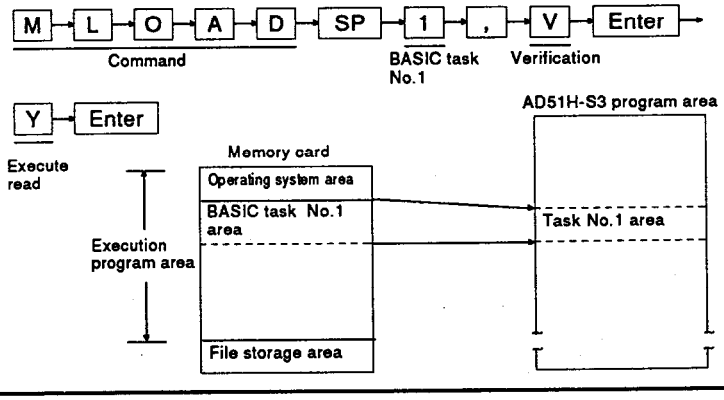
Reads an execution program (stored in the BASIC task No.1 area of a memory card) using the BASIC task No.1 area, and verifies the read program with the source program.

Before the command is input



After the command is input





(1) Memory card location
- To use the MLOAD command for a memory card, the memory card must be installed in AD51H-S3 MEMORY CARD[1].

OPERATING PROCEDURE

M → L → O → A → D → SP

```
S>MLOAD
```

1 Input the MLOAD command to read the execution program (stored in the memory card/EEP-ROM) using the communication module execution program area.

1

```
S>MLOAD 1
```

2 Designate the BASIC task No. area (AD51H-S3: 1 to 8, A1SD51: 1,2) of the program stored in the execution program area.

(This example assumes that the BASIC task No.1 program area is read.)

, → V → Enter

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

3 Designate "V" to verify the read program with the source program after reading the program.

Press the [Enter] key to read without doing verification.

(This example assumes that the read program is verified with the source program.)

(2) Precautions when using the MLOAD command
- The BASIC task No. area (designated by using the MSAVE or SET commands) must be as large as the communication module BASIC task No. area (designated using the START command).
- Make sure that the interpreter is not working in the communication module BASIC task No. area (where the execution program stored in a memory card/EEP-ROM is written). When the interpreter is working, stop the interpreter operation by using the TKILL command.

Y → Enter

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

4 The "LOAD (Y/N)?" dialog box appears.

Press the [Y] key to execute reading.

Press the [N] key to cancel the read operation.

(The console remains in a wait state until either key is pressed.)

(This example assumes that reading is executed.)

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

5 The next line shows the result of the execution.

When reading is executed normally, the screen shows "LOAD OK".

If reading is not executed normally, an error message appears.

(This example assumes that reading is executed normally.)

6 "S>" appears after the execution result is displayed.

Input the necessary command.

(2) References

- Writing an execution program (stored in the communication module) to a memory card/EPP-ROM.....MSAVE command (see Section 4.3.2)
- Changing the multitask setting/setting description.....SET command (see Section 4.4.1)
- Displaying the multitask setting description.....SET? command (see Section 4.4.2)
- Setting the communication module to the editing mode (1).....START command (see Section 4.5.1)
- Stops the interpreter operation in a specific BASIC task No. area.....TKILL command (see Section 4.6)

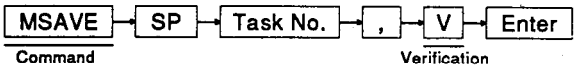
4.3.2 Writing an execution program (stored in the communication module) to a memory card/EEP-ROM (MSAVE command)

This operation writes an execution program (stored in a designated BASIC task No. area of the AD51H) to the corresponding BASIC task area of a memory card/EEP-ROM.

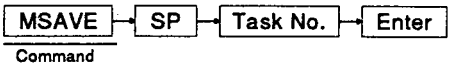
Executing this operation automatically sets multitasking in the designated task area.

INPUT PROCEDURE (This command is also referred to as "MS")

To verify the written program with the source program:



Only to write:

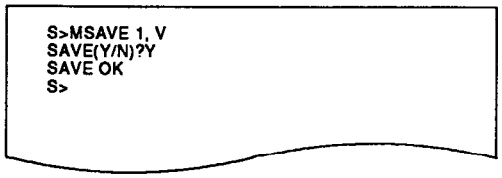


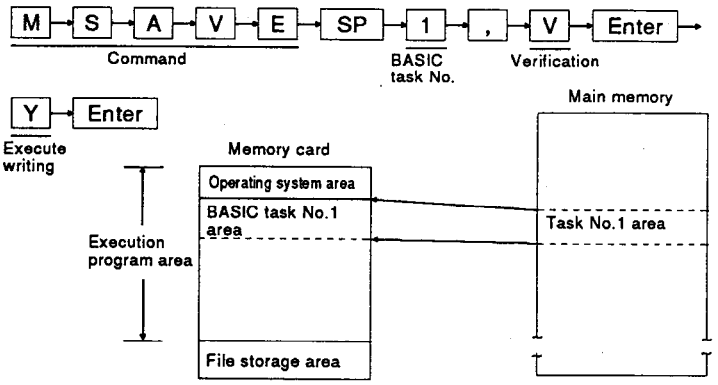
OPERATION EXAMPLE

Writes an execution program (stored in the BASIC task No. 1 area) to the BASIC task No. 1 area in a memory card, and verifies the written program with the source one.

Before the command is input

After the command is input





(1) Memory card location
- To use the MSAVE command with a memory card, the memory card must be installed in MEMORY CARD[1] of the AD51H-S3.

(2) Precautions when using the MSAVE command
- Use the START command to start the interpreter. Then (a) execute the SYSTEM command to the interpreter, or (b) press the [Ctrl] + [D] keys, and execute the MSAVE command.
- Since writing another execution program to the BASIC task No. area (to which an execution program has already been written) can cause an overwrite in the BASIC task No. area, take the following steps:
  (a) Write all execution programs to the execution area of the memory card/EEP-ROM.
  (b) Reset multitasking when appropriate.

OPERATING PROCEDURE

M → S → A → V → E → SP

```
S>MSAVE
```

1 Input the MSAVE command to write an execution program stored in the communication module to the memory card/EEP-ROM.

1

```
S>MSAVE 1
```

2 Designate the communication module BASIC task No. area (AD51H-S3: 1 to 8, A1SD51S: 1,2).

(This example assumes that the execution program is stored in the BASIC task No.1 area.)

, → V → Enter

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

3 Designate "V" to verify the written program with the source program after writing the program.

Press the [Enter] key to read without verification.

(This example assumes that the written program has been verified with the original.)

---

(3) Precautions when using the MSAVE command
- Data for the communication module BASIC task area is written as an execution program to the corresponding BASIC task area of a memory card/EEP-ROM.
- After writing the execution program, multitasking is automatically set in the designated BASIC task area. The setting description is shown below:
  The SET command explanation in this manual gives details.
  - Starting condition ..... "BOOT" attribute is set.
  - Size ..... Designated task size when the START command is executed is set.
  - Execution sequence..... Nothing is set.

Y → Enter

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

4 The "SAVE (Y/N)?" dialog box appears.

Press the [Y] key to execute writing.

Press the [N] key to cancel the write operation.

(The console remains in a wait state until either key is pressed.)

(This example assumes that writing is executed.)

5 The next line shows the result of the execution.

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

When writing is executed normally, the screen shows "SAVE OK".

If writing is not executed normally, an error message appears.

(This example assumes that the MSAVE command is executed normally.)

6 "S>" appears after the execution result is displayed.

Input the necessary command.

(4) References

- Reading an execution program (stored in a memory card/EEP-ROM) using the main memory
  .....MLOAD command (see Section 4.3.1)
- Changing the multitask setting/setting description .....SET command (see Section 4.4.1)
- Displaying the multitask setting description.....SET? command (see Section 4.4.2)
- Setting the communication module to the editing mode (1)
  .....START command (see Section 4.5.1)

4.4 Setting/Changing/Displaying Multitasking Descriptions

This section tells how to set/change/display multitasking descriptions using system commands to control the multitask setting.

The multitask setting is used to set the starting conditions when starting up the communication module and executing several programs during multitasking.

The multitask setting contains the following items designated by using the MSAVE or SET commands:

(1) Starting condition

Sets the condition when the BASIC program is stored in the BASIC task No. area.

(a) START

- After power to the communication module is turned ON or the communication module is reset, an execution program (stored in the memory card/EEP-ROM execution program area) is read using the communication module execution program area, and program execution is started.

(b) BOOT

- When the communication module is started up, an execution program stored in the memory card/EEP-ROM BASIC task No. area is read by using the communication module execution program area.
- If the program being executed gives a ZSTART command, then the designated BASIC program will start.

(c) IT

- When the communication module is started up, an execution program stored in the memory card/EEP-ROM BASIC task No. area is read by using the communication module execution program area.
- If a PC CPU turns ON an output signal (such as the start task number designation flag or task start signal) for the communication module, then the designated BASIC program will start.

(d) ON

- After starting up the communication module, if the BASIC program being executed gives a ZSTART command, then the designated program will be read from the memory card file storage area and the execution will start.

(e) OFF

- Invalidates the multitask setting for a task No. area.

  If this is executed for a task No. area, then the BASIC program cannot be executed in that area.

(2) Task size

Designates the size of the BASIC task area (16K, 32K, 48K, or 64K bytes).

(3) Execution sequence

After the communication module starts up, if several programs are installed in the corresponding task No. areas and those programs are executed, designate which program will be executed first.

If an execution program is written to the memory card/EEP-ROM execution program area used for multitasking, then the multitasking is automatically set in that area.

This section tells how to set/change/confirm the multitask setting.

(1) Changing the task size in the multitask setting
- To change the task size in the multitask setting in order to enlarge the corresponding task No. area, take the following steps:
  (a) Save all the communication module execurtion programs using the SAVE command.
  (b) Reset each task size so that all execution programs (max.8) can be written using the appropriate area designated when the memory card was formatted.

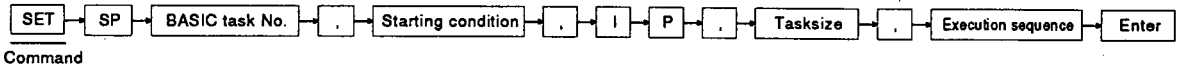
(2) Section 4.3.2 gives details about the MSAVE command.
  (c) Reset the communication module.
  (d) Read the execution program with LOAD command and execute the MSAVE command.

4.4.1 Setting/Changing the multitask (SET command)

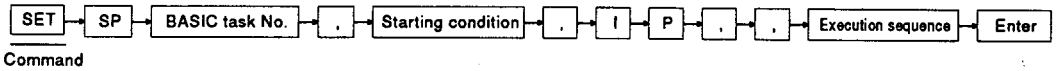
This operation sets multitasking for a task No. area and changes the multitask setting.

INPUT PROCEDURE (This command is also referred to as "S")

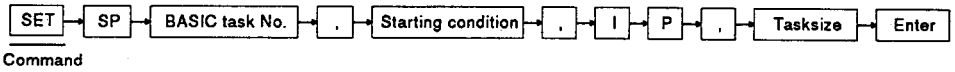
To set/change the starting condition, size and execution sequence:



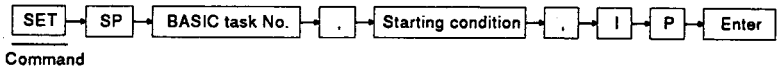
To change the starting condition and execution sequence:



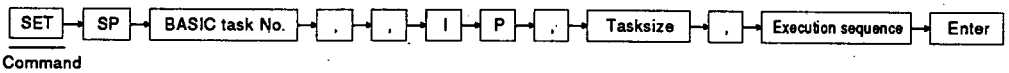
To change the starting condition and size:



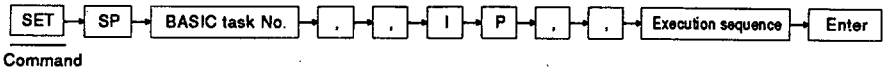
To change the starting condition:



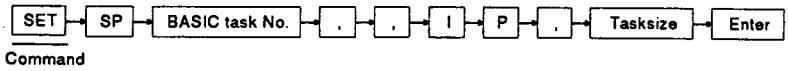
To change the size and execution sequence:



To change the execution sequence:



To change the size:



OPERATION EXAMPLE

Sets multitasking for the BASIC task No.1 area.

Before the command is input

```
S>
```

S → E → T → SP → 1 → , → S → T → A → R → T

Command: SET — Task No.: 1 — Starting condition: START

After the command is input

```
S>SET 1, START, IP, 48, 2
SET OK
S>
```

, → I → P

, → 4 → 8

Task size: 48

, → 2 → Enter

Execution sequence: 2

OPERATING PROCEDURE

S → E → T → SP

```
S>SET
```

1 Input the SET command to set/change multitasking.

1

```
S>SET 1
```

2 Designate the BASIC task No. area (AD51H-S3: 1 to 8, A1SD51S: 1, 2) for which multitasking is set.

(This example assumes that multitasking is set/changed for the communication module BASIC task No.1 area.)

, → S → T → A → R → T

```
S>SET 1, START
```

3 To set/change the starting condition for the communication module BASIC task No. area, select one of the following items:

1) START
2) BOOT
3) IT
4) ON
5) OFF

If the starting condition is not designated, input a comma (,). In this case, the previously designated starting condition will be used.

(This example assumes that START is selected.)

(1) Section 4.4 gives details about starting conditions.

, → I → P

```
S>SET 1, START, IP
```

4 Designate "IP" as the type of execution program.

, → 4 → 8

```
S>SET 1, START, IP, 48
```

5 Input "16", "32", "48", or "64" to set/change the corresponding BASIC task area.

If the task size is not designated, input a comma (,).

In this case, the previously designated task size will be used.

(This example assumes that the task size is set to 48K bytes.)

, → 2 → Enter

```
S>SET 1, START, IP, 48, 2
```

6 When the communication module is started, input a number from 1 to 8 to set/change the execution sequence of the multitask programs for which "START" is designated. (When "1" is designated, the corresponding program is given the highest priority.)

If several task areas have the same number, the tasks stored in those areas are processing in ascending order of the task numbers.

When the execution sequence is not designated, press the [Enter] key.

In this case, the previously designated execution sequence is used.

(This example assumes that "2" is designated for the execution sequence.)

7 The next line shows the result of the execution.

When the SET command is executed normally, the screen shows "SET OK". If the SET command is not executed normally, an error message appears.

(This example assumes that the SET command is executed normally.)

```
S>SET 1, START, IP, 48, 2
SET OK
S>
```

8 "S>" appears after the execution result is displayed.

Input the necessary command.

---

(2) Size designation
- The size can be designated in decimal, hexadecimal ("&H[ ][ ][ ][ ]"), or binary ("&B[ ][ ] to [ ]").

(3) References
- Writing BASIC task area information stored in the communication module to a memory card/EEP-ROM ..... MSAVE command (see Section 4.3.2)
- Displaying the multitask setting description ..... SET? command (see Section 4.4.2)
- Setting the communication module to editing mode (1) ..... START command (see Section 4.5.1)

4.4.2 Displaying the multitask setting description (SET? command)

This operation displays the multitask setting description for each BASIC task No. area of the communication module.

INPUT PROCEDURE (This command is also referred to as "S?")

To designate one of the BASIC task No. areas:

SET? (Command) → SP → Basic Task No. → , →

Designate the memory location display → , → Device to be display → Enter

To designate all BASIC task No. areas:

SET? (Command) → Enter

OPERATION EXAMPLE

Displays the multitask setting description for the BASIC task No.1 area of the communication module.

Before the command is input

```
S>
```

S → E → T → ? → SP → 1 → , → L → , → R → Enter

Command: S E T ?; BASIC Task No.: 1; Memory location display: L; DRAM: R

After the command is 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
```

OPERATING PROCEDURE

S → E → T → ?

```
S>SET?
```

1 Input the SET? command to display the multitask setting description.

SP → 1 → Enter

```
S>SET? 1
```

2 Designate the BASIC task No. area (AD51H-S3: 1 to 8, A1SD51S: 1,2) whose setting description will be displayed.

When all areas have been designated, press the [Enter] key.

(This example assumes that No.1 has been designated.)

[,] → [L]

```
S>SET? 1,L
```

3 When booting from a DRAM, input L to display the allocation of each task.

Input a comma (,) to cancel the display.

The following items are displayed:

1) Start No.
2) Size
3) BASIC task No.

[,] → [R] → [Enter]

```
S>SET? 1,L,R
```

4 Input devices (U/R) which display the multitask setting.

U: Displays the user's ROM multitask setting.

R: Displays the multitask setting currently booted from the DRAM.

```
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
```

5 The execution result of the command is displayed.

When a command is executed normally, the multitask setting information about the designated task No. area and the allocation of each task are displayed beginning with the following line.

If the command is not executed normally, an error message, etc. are displayed in the following line.

The following is displayed when a command is executed normally. (In the example on the left, the setting description of the BASIC task No. 1 area are displayed.)

The SET? command explanation gives details about the displayed description.

1) Task No. ..... Displays the task No. of the corresponding task No. area.

2) Type..... Displays the IP/CP that was input right after the start condition by using the SET? command.

3) Size..... Displays the corresponding task No. area size.
Corresponds to the task size set by the SET? command.

4) Start Condition... Used to execute the BASIC program in a designated area.
Corresponds to the start condition set by the SET? command.

5) Start No. ..... Indicates the execution sequence when the start condition 4) is set to START.
Corresponds to the execution sequence set by the SET? command.
If the start condition is not "START", "-" will be displayed at the position corresponding to this item, since it is ignored.

6) Location..... Displays the location in memory where the task has been allocated. (When the type is CP.)

6 "S" appears after the execution result is displayed.

Input any necessary command.

(2) References

- Writing BASIC task area information..... MSAVE command (see Section 4.3.2)
  (stored in the communication module) to a memory card/EEP-ROM
- Setting/changing the multitask setting ..... SET command (see Section 4.4.1)
  description
- Setting the communication module to edit mode (1)
  ..... START command (see Section 4.5.1)

4.5 Changing the Communication Module Mode

This section tells how to use the system commands (used to control the mode) to change the communication module mode.

4.5.1 Setting the communication module to the editing mode (1) (START command)

This operation edits/debugs a program.

INPUT PROCEDURE (This command is also referred to as "ST")

To set/change the task area size:

START → SP → BASIC task No. → , → Task size → Enter
(START: Command)

To not change the task area size:

START → SP → BASIC task No. → Enter
(START: Command)

OPERATION EXAMPLE 1

Starts editing a new BASIC program in the BASIC task 1 area.
Sets the task 1 area to 48K bytes.

Before the command is input

```
S>
```

S → T → A → R → T → SP → 1 → , → 4 → 8 → Enter
(START: Command; 1: Task No.; 4 8: Task size)

After the command is input

```
S>START 1, 48
```

```
OK
[]
```
Cursor located here

If the interpreter was not operating during the execution, the following message appears before "OK":
"AD51H-BASIC ON-LINE PROGRAMMING Ver [ ]"

OPERATION EXAMPLE 2

Starts editing an existing BASIC program in the BASIC task 1 area.
Changes the task 1 area size to 64K bytes.

Before the command is input

```
S>TKILL 1:
KILL OK
S>
```

S → T → A → R → T → SP → 1 → , → 6 → 4 → Enter
(START: Command; 1: Task No.; 6 4: Task size)

After the command is input

```
S>TKILL 1:
KILL OK
S>START 1, 64
```

```
OK
```

If the interpreter was not operating during the execution, the following message appears before "OK":
"AD51H-BASIC ON-LINE PROGRAMMING Ver [ ]"

OPERATING PROCEDURE

S → T → A → R → T → SP

```
S>START
```

1 Input the START command to set the communication module to the editing mode (1).

1

```
S>START 1
```

2 Designate the task No. area(AD51H-S3: 1 to 8, A1SD51S: 1,2) in which the BASIC program to be edited/debugged will be stored.

The task number can be omitted.

Omitting the task number is regarded as doing the following designation:

1) When the START command is initially input, the designation is "1".
2) If the START command was already used, the previous task number is used.

(This example assumes that the BASIC program is edited/debugged in the task No.1 area.)

, → 4 → 8 → Enter

```
S>START 1, 48
```

3 Input "16", "32", "48", or "64" to set/change the task area size (in units of 1K byte).

After completing editing/debugging, giving the MSAVE command (a) writes the data in the BASIC task area to the memory card/EEP-ROM, and (b) automatically sets the task size.

When designating a BASIC task No. area for which multitasking is not set, be sure to input the task size.

Even when designating a BASIC task area for which multitasking is already set, be sure to input the task size if the task size needs to be changed.

If the set task is not changed, just press the [Enter] key.

(This example assumes that the BASIC task 1 area size is set/changed to 48K bytes.)

---

(1) Size designation
- The size can be designated in decimal, hexadecimal ("&H[ ][ ][ ][ ]"), or binary ("&B[ ][ ] to [ ]").

```
AD51H-BASIC ON-LINE PROGRAMMING Ver[ ]
OK
```

or

```
OK
```

4 The next line shows the result of the execution.

When the START command is executed normally, the screen enters the state shown on the left.

Start editing/debugging the program.

The AD51H-BASIC Programming Manual tells how to edit/debug the BASIC program.

If the START command is not executed normally, an error message appears.

The example on the upper left shows the description displayed if the interpreter was not started before the execution.

The example on the lower left shows the description displayed when the interpreter was already started before the execution.

```
SYSTEM
```

⇩

```
S>
```

5 To return the communication module from the edit mode (1) to the system mode after editing/debugging is completed, execute one of the following:

1) Execute the BASIC SYSTEM command.
   - Stops executing the BASIC program.
   - Closes the open file and the communications line.
2) Press the [Ctrl] + [D] keys.
   - Stops executing the BASIC program.
   - Leaves the open file and the communications line as they are.
   - Can resume (or continue) executing the program using the CONT command if the BASIC program was not changed when the communication module was reset to the editing mode using the START command.

---

(2) Precautions when using the START command
- When the task size must be changed, use the TKILL command to stop operation of the interpreter in the utilized task No. area in order to edit/debug the BASIC program in the following task areas:
  1) A task No. area for which the multitask is set
  2) A task No. area that already contains the BASIC program

  In addition, when the task size is enlarged, take the following steps to write the BASIC program to the memory card/EEP-ROM execution area (using the MSAVE command) after editing/debugging a program:
  1) Write all execution programs to the execution area of the memory card.
  2) Reset multitasking so that all appropriate execution programs (max.8) can be written using the area designated when the memory card was formatted.

(3) Debugging the BASIC program after executing the START command
- In the editing mode (1), debug the program as shown in the programming manual.
- The debug commands shown in Section 5 cannot be used.

(4) References
- Writing the BASIC task area information stored in the main memory to a memory card ..... MSAVE command (see Section 4.3.2)
- Changing the communication module mode ..... GO command (see Section 4.5.2)
- Stopping the interpreter operation in a designated task No. area ..... TKILL command (see Section 4.6)

4.5.2 Setting the communication module to the execution/system mode (GO command)

This operation switches the communication module between the system mode and the execution/debug/execution (2).

To give the command to debug the BASIC multitask program, set the communication module to the debug mode. (see Section 5)

To start executing the BASIC program by setting multitasking, set the communication module to the execution mode (2) to start

When the communication module is returned to the system mode, giving the system command enables the BASIC program to be edited/debugged.

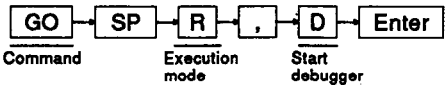
The following table gives the mode and debug start designation when the GO command is input and the state of the console and debugger after the GO command is executed.

<table>
<tr><th>Mode Designation</th><th>Debug Start Designation (YES/NO)</th><th>Console State</th><th>Debugger Terminal State</th><th>Remarks</th></tr>
<tr><td rowspan="2">R
(Execution mode (1))</td><td>YES
(To the debug mode)</td><td rowspan="2">Clears the displayed description.
Sets the terminal to be used for running the BASIC program.</td><td>Starts the debugger, clears the screen, and displays "D>" on the screen.
Enables the debug commands to be input.</td><td rowspan="2">Setting multitasking installs the BASIC program in the corresponding task No. area, starting the execution.</td></tr>
<tr><td>NO
(To the execution mode (2))</td><td rowspan="2">Displayed descriptions remain as they are.
Becomes a general-purpose port used for running the BASIC program.</td></tr>
<tr><td>P
(System mode during programming)</td><td>Cannot be designated.</td><td>Clears the displayed description, and displays "S>".
Enables the system commands to be input.</td><td>Stops executing BASIC programs in each task No. area.</td></tr>
</table>

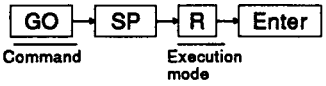
(1) BASIC program state when the GO command is executed
- Designating the execution mode (1) starts executing the BASIC program in the same way when the communication module is started up by setting the communication module mode switch (1) to "0", "1", or "3".
- Designating the programming mode stops the execution of all programs in task areas. In this case, since the task area memory state is not changed, all BASIC programs remain as they are.

INPUT PROCEDURE (No command abbreviation)

To set the communication module to the debug mode:



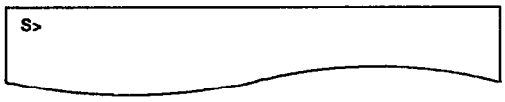
To set the communication module to the execution mode:

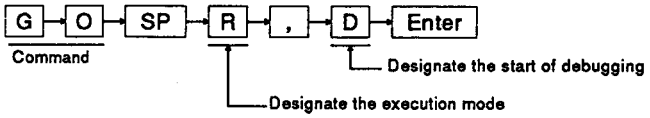


OPERATION EXAMPLE

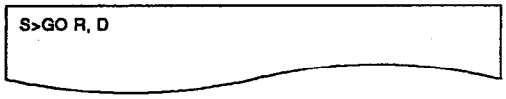
Sets the communication module to the debug mode:

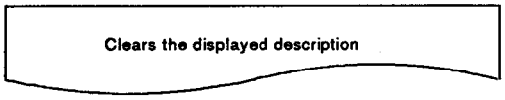
Before the command is input



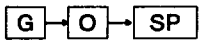


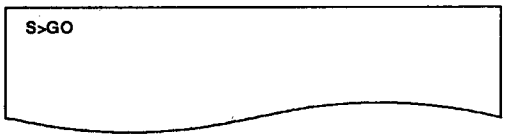
After the command is input



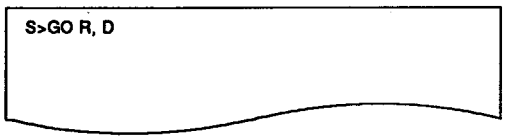


OPERATING PROCEDURE





1 Input the GO command to change the communication module mode.



2 Designate the mode.

Input "R" to set the communication module to the execution mode.

Input "R, D" to set the communication module to the debug mode.

(This example assumes that the communication module is set to the debug mode.)

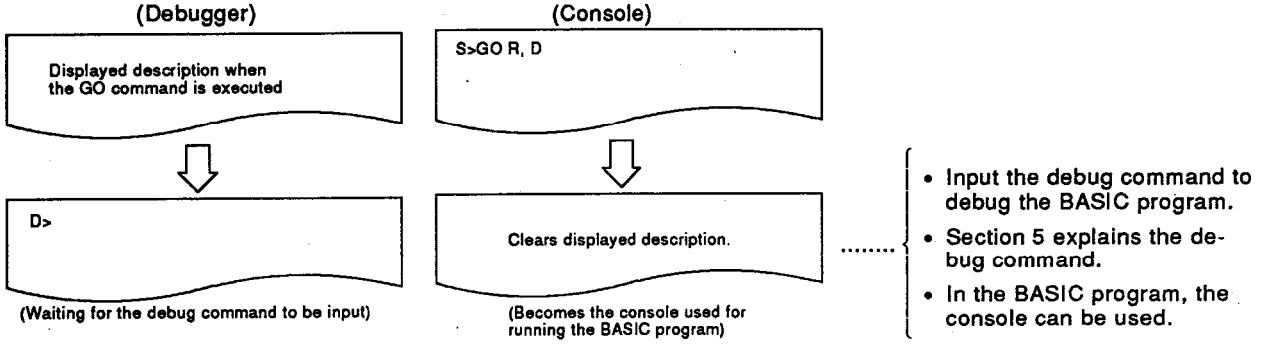
3 The next line shows the result of the execution.

When the GO command is executed normally, the screen enters the state shown below:

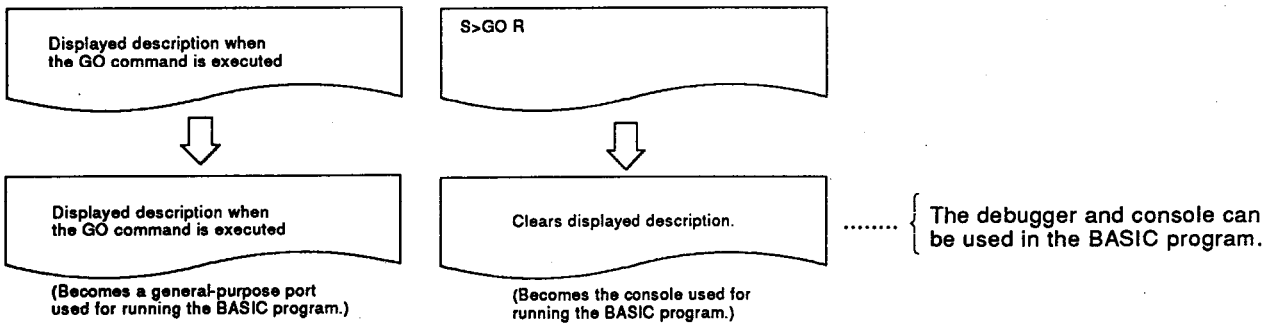
If the GO command is not executed normally, an error message appears.

The following example shows the displayed description when the GO command is executed normally:

1) When the debug mode is designated:



2) When the execution mode is designated:



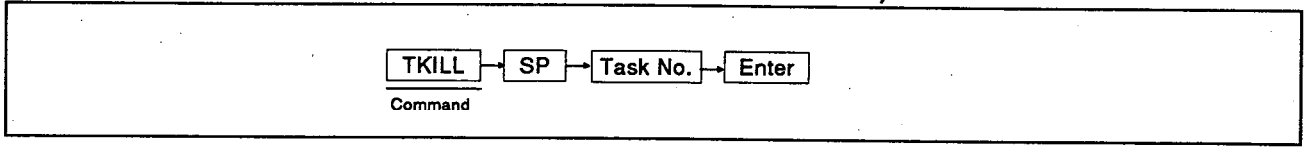
(2) Communication module mode change
See the communication module mode change chart in Section 2.3.

(3) Reference
• Displaying the MAIN MENU on the console screen.....EXIT command (see Section 4.7)

4.6 Stopping the Interpreter Operation in a Designated Task No. Area (TKILL Command)

This section tells how to stop the interpreter operation in a designated task No. area using the system command (TKILL) to control the interpreter operation.

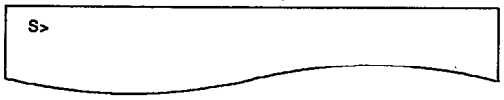
INPUT PROCEDURE (This command is also referred to as "TK")

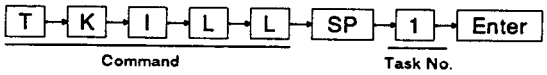


OPERATION EXAMPLE

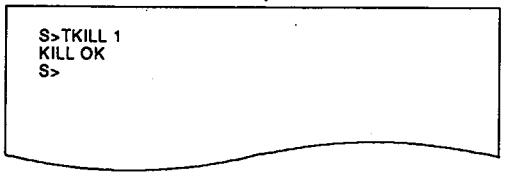
Stops the interpreter operation in the BASIC task No.1 area.

Before the command is input

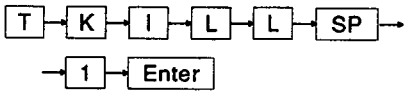


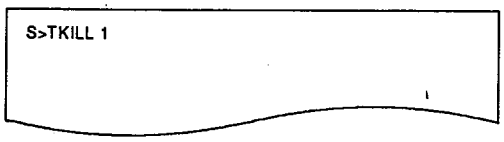


After the command is input



OPERATING PROCEDURE





1 To stop operating the interpreter, input the TKILL command accompanied by the corresponding task number (AD51H-S3: 1 to 8, A1SD51S: 1,2).

(This example assumes that the interpreter operating in the task No.1 area has been stopped.)

```
S>TKILL 1
KILL OK
S>
```

2 The next line shows the result of the execution.

When the TKILL command is executed normally, the screen shows "TKILL OK".

If the TKILL command is not executed normally, an error message appears.

(This example assumes that the TKILL command is executed normally.)

3 "S>" appears after the execution result is displayed.

Input the necessary command.

(1) When the TKILL command should be used

To execute the following operations and stop the interpreter operation.

1) Setting the communication module to the system mode and changing a task size using the START/SET command.
2) Setting the communication module to the system mode and reading an execution program in a designated memory card/EEP-ROM BASIC task No. area using the corresponding task No. area in the communication module.

(2) References

- Setting the communication module to the editing mode (1) ..... START command (see Section 4.5.1)
- Changing the communication module mode ..... GO command (see Section 4.5.2)

4.7 Displaying the MAIN MENU on the Console Screen (EXIT Command)

This section tells how use the EXIT command to display the MAIN MENU on the console screen when a PC/AT is used as the console.

When a VG-620 or a VG-382/VT-220 is used as the console, pressing any key redisplays "S>" after the EXIT command is input.

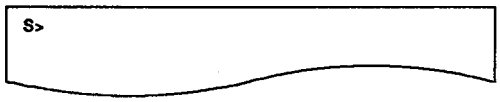
INPUT PROCEDURE (This command is also referred to as "E")

EXIT → Enter
Command

OPERATION EXAMPLE

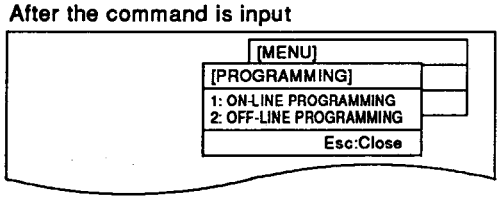
Displays the MAIN MENU on the console screen.

Before the command is input



E → X → I → T → Enter
Command

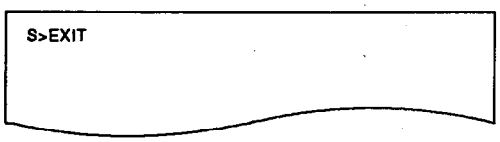
After the command is input



OPERATING PROCEDURE

E → X → I → T → Enter

1 Input the EXIT command to display the MAIN MENU.



(1) BASIC program state when the EXIT command is executed
Even when the EXIT command is executed, BASIC programs in the BASIC task No. areas will be executed.

(2) Precautions when the command is input
To display the MAIN MENU to edit the BASIC program in a task No. area, stop the BASIC program before the EXIT command is input
(Execution of the BASIC program can influence the system control.)

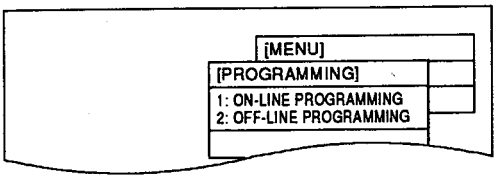
2 Do the corresponding operation after displaying the result of the execution.

When the display is executed normally, the console screen enters the following state:

If the display is not executed normally, an error message appears.

(This example assumes that the display is executed normally.)

1) When a PC/AT is used as the console:



The console screen displays the MAIN MENU.

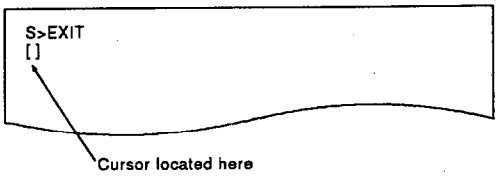
Select an operation from the MAIN MENU.

The SW1IX-AD51HPE Operating Manual tells how to use the MAIN MENU.

2) When a VG-620 or a VT-382/VT-220 is used as the console:

The console enters the state of waiting for key input.

Press any key to display "S>", and input a system command.



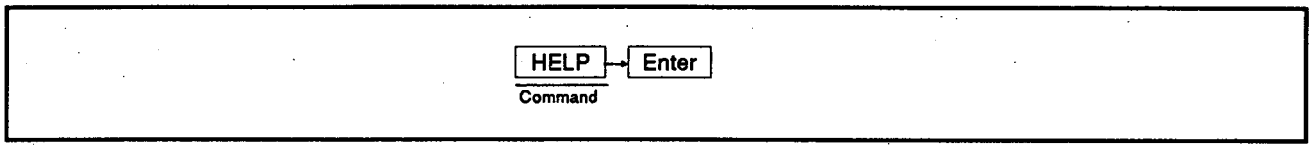
(2) Communication module mode changes
See the communication module mode change chart in Section 2.3.

(3) Reference
Changing the communication module mode ..... GO command (see Section 4.5.2)

4.8 Confirming the System Command Input Procedure (HELP Command)

This section tells how to use the HELP command when displaying a command input procedure to confirm the input procedure.

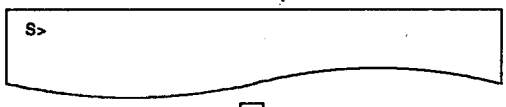
INPUT PROCEDURE (This command is also referred to as "H")

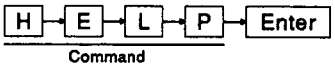


OPERATION EXAMPLE

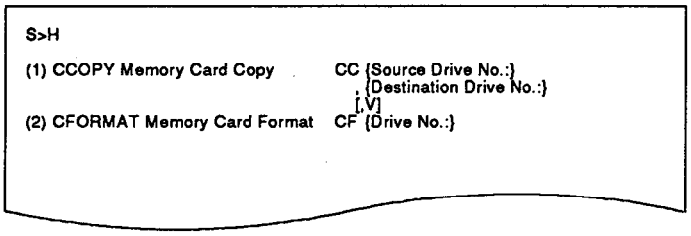
Displays the system command input procedure.

Before the command is input

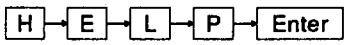




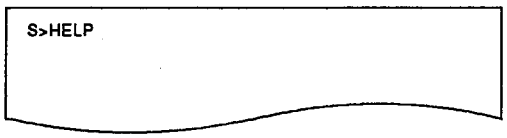
After the command is input



OPERATING PROCEDURE



1 Input the HELP command to display the system command input procedure.



2 The result of the execution is displayed.

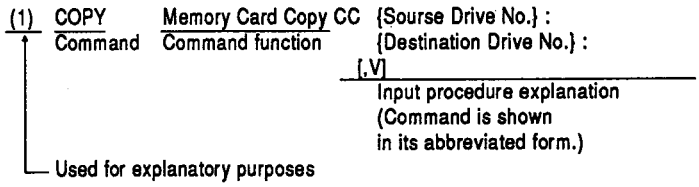
When the HELP command is executed normally, the subsequent lines show nine kinds of system command functions and the input procedure.

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

When three kinds of system command functions and the input procedure (as shown on the next page) are displayed, press any key other than the [Esc] key.

Pressing the [Esc] key stops the HELP function.

(Example)



If the HELP command is not executed normally, an error message appears.

3 "S>" appears after the execution result is displayed.

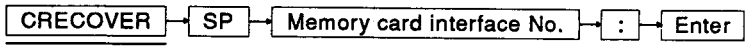
Input the necessary command.

(1) Display description of a command input procedure
A space after the command requires pressing the [SP] key one time.
Since braces ("{" and "}") are only used to indicate the beginning and end of a command argument, they don't need to be actually input.
Since brackets ("[" and "]") are used only to indicate an "omissible part", they don't need to be actually input.

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

This section tells how to search and recover an unusable area in the file area of a memory card in a designated drive.

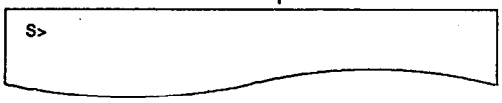
INPUT PROCEDURE (This command is also referred to as "CR")

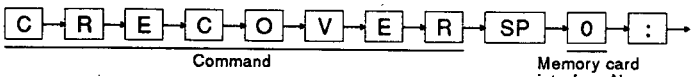


OPERATION EXAMPLE

Recovering an unusable file area in the memory card installed in MEMORY CARD[1].

Before the command is input





Enter

After the command is input

```
S>CRECOVER 0:
RECOVER (Y/N) ? Y
RECOVER OK
S>
```

OPERATING PROCEDURE

C → R → E → C → O → V → E → R → SP

```
S>CRECOVER
```

1 Input the CRECOVER command to recover a file area in the memory card.

0 → : → Enter

```
S>CRECOVER 0:
```

2 Input the interface number of the memory card with the file area to be recovered and a colon (:).

The numbers that can be input are "0" and "1".

0: Corresponds to MEMORY CARD[1] of the AD51H-S3

1: Corresponds to MEMORY CARD[2] of the AD51H-S3

(This example assumes that the memory card installed in MEMORY CARD[1] is designated.)

Y → Enter

```
S>CRECOVER 0:
RECOVER (Y/N) ? Y
```

3 The "RECOVER (Y/N)?" dialog box is displayed.

Press the [Y] key to recover.

Press the [N] key to cancel the recover operation.

(The console remains in a wait state until a system command is input.)

(This example assumes that the recover operation is executed.)

```
S>CRECOVER 0:
RECOVER (Y/N) ? Y
RECOVER OK
S>
```

4 The next line displays the result of the execution.

When the command is executed normally, the screen displays "RECOVER OK".

If the recover operation is not executed normally, an error message appears.

(This example assumes that the recover operation is executed normally.)

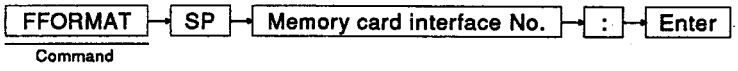
5 "S>" appears after the execution result is displayed.

Input any necessary command.

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

This section tells how to format (logical format) the file area in a memory card installed in MEMORY CARD[1] or MEMORY CARD[2] of the AD51H-S3.

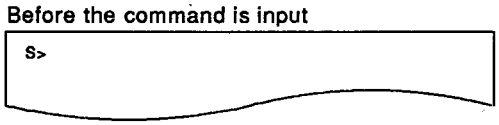
INPUT PROCEDURE (This command is also referred to as "FFM")

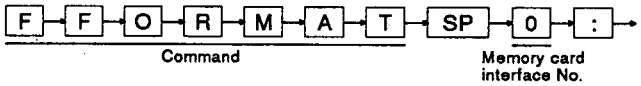


OPERATION EXAMPLE

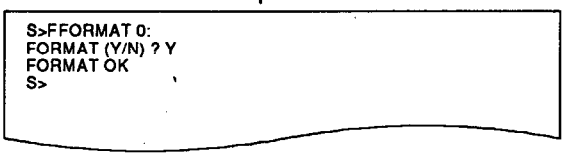
Formatting the memory card installed in MEMORY CARD[1] of the communication module.

Before the command is input

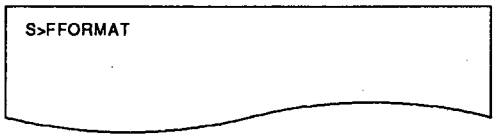




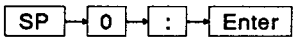
After the command is input

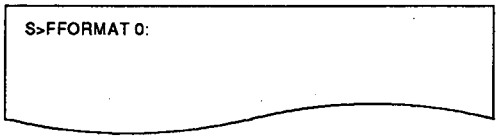


OPERATING PROCEDURE



1 Input the FFORMAT command to format the memory card.





2 Input the interface number which corresponds to the memory card to be formatted and a colon (:).

The numbers that can be input are "0" and "1".

0: Corresponds to MEMORY CARD[1] of the AD51H-S3.

1: Corresponds to MEMORY CARD[2] of the AD51H-S3

(This example assumes that the memory card installed in MEMORY CARD[1] will be formatted.)

(1) Precautions when using the FFORMAT command.
- Formatting a memory card deletes all data in that memory card.
- Set the write-protect function for unprotected when formatting a memory card which has a write-protect function.
- Turn OFF the memory protect keyswitch on the AD51H-S3 when the memory card installed in MEMORY CARD[1] is to be formatted.

Y → Enter

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

3 The "FORMAT (Y/N)?" dialog box is displayed.

Press the [Y] key to execute formatting.

Press the [N] key to cancel the format operation.

(This example assumes that the format operation is executed.)

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

4 The next line displays the result of the execution.

When the command is executed normally, the screen displays "FORMAT OK".

If the recover operation is not executed normally, an error message appears.

(This example assumes that the format operation is executed normally.)

5 "S>" appears after the execution result is displayed.

Input any necessary command.

(2) Reference
Formatting a memory card(physical format)..... CFORMAT command (Section 4.2.2)

5. MULTITASK DEBUGGING

Multitask debugging is used to (a) find an error in a program, and (b) correct that error during multitasking.

This section tells how to input a debug command from the debugger to debug all programs when BASIC programs are executed by multitasking.

(1) Since most of this section concerns key inputting and displays on the debugger, the explanations assume that all key inputting and displays refer to the debugger.
When key inputting and displays refer to the console rather than the debugger, the word "console" is always used to avoid misunderstanding.

(2) Executing the operations discussed in this section requires the following preparations:
- Setting the communication module switch to program online ..... see Section 2
- Connecting the debugger ..... see Section 2
- Creating BASIC programs and debugging them ..... see Section 4
- Storing programs in memory ..... see Section 4.3.2
- Setting multitasking ..... see Section 4.4.1

(3) Precautions when inputting debug commands
If the debugging system (operating system to analyze a debug command and execute it) is not in a state to execute an input command, the execution of that command will be suspended until the debugging enters an executable state.
After displaying "D>" again, input the necessary command.

5.1 Debug Commands

Table 5.1 lists the debug commands used for multitask debugging.

Table 5.1 List of Debug Commands

Classification	Debug Command	Function		Reference Section	Module Availability	
					AD51H-S3	A1SD51S
Task control	TSTATUS*1	Displays the state of BASIC programs in the designated task area.		5.2.1	o	o
	TRUN*1	Starts executing a BASIC program in the designated task area.		5.2.2		
	TSTOP*1	Stops executing a BASIC program in the designated task area.		5.2.3		
	TCONTINUE*1	Restarts execution of a stopped BASIC program in the designated task area.		5.2.4		
	T?*1	Displays the value of a designated variable in the BASIC program (existing in the designated task area).		5.2.5		
	TLET*1	Assigns a value to a designated variable in the BASIC program (existing in the designated task area).		5.2.6		
Memory access control	MREAD	Displays the range of addresses that can be shared by BASIC programs.	•Buffer •Main memory •Extension relay (EM) •Extension register (ED)	5.3.1	o	o
	MWRITE	Writes a value to a designated address that can be shared by BASIC programs.		5.3.2		
	B@	Displays internal device bit data that can be shared by BASIC programs.	•General-purpose input (X) •General-purpose output (Y) •Extension relay (EM)	5.3.3		
	B@	Writes bit data that can be shared by BASIC programs to an internal device.		5.3.4		
	W@	Displays internal device word data that can be shared by BASIC programs.	•Extension register (ED)	5.3.5		
	W@	Writes word data in that can be shared by BASIC programs to an internal device.		5.3.6		
OS information confirmation	ZSTATUS	Displays the operating states of the event, message port, and source (that can be shared by BASIC programs).		5.4.1 5.4.2 5.4.3	o	o
Mode control	START*1	Switch the communication module from the debug mode to editing mode (2). (To edit a program during multitasking)		5.5.1	o	o
	GO	Switch the communication module from the debug mode to the system mode or execution mode (2); or vice versa.		5.5.2		
Others	EXIT	Displays the MAIN MENU on the debugger.		5.6	o	o
	HELP	Displays the debug command list, functions, and command input procedures.		5.7		

*1 Cannot be executed with tasks that contain compiled BASIC programs.

o: Available x: Unavailable

5.2 Controlling BASIC Program Operations

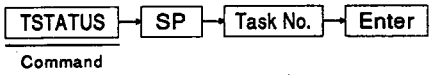
This section tells how to use the debug commands (to control a task) when controlling BASIC programs.

5.2.1 Displaying the state of a designated program (TSTATUS command)

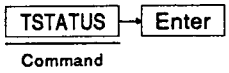
This operation displays the state of a BASIC program in the designated task area.

INPUT PROCEDURE (This command is also referred as "TS")

To designate one task area:



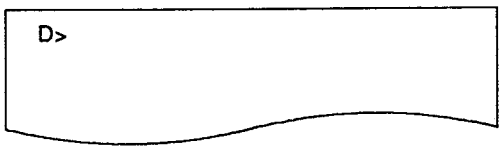
To designate all task areas:

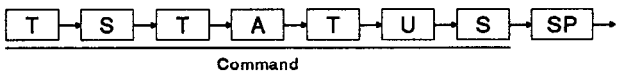


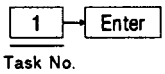
OPERATION EXAMPLE

Displays the status of a BASIC program in the task No.1 area.

Before the command is input



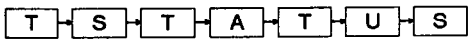




After the command is input

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

OPERATING PROCEDURE




```
D>TSTATUS
```

1 Input the TSTATUS command to display the state of a BASIC program.

SP → 1 → Enter

```
D>TSTATUS 1
```

2 Input the task number (AD51H-S3: 1 to 8, A1SD51S: 1, 2) to display the state.

When all task areas are designated, press the [Enter] key.

(This example assumes that task 1 is designated.)

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

3 The execution result is displayed.

When the display is not executed normally, the next line shows "TSTATUS: Error" accompanied by the error code.

When the display is executed normally, the contents are shown below:

(This example assumes that the task No.1 area status is displayed.)

1) TASK NO : Designated task number

2) STATUS : BASIC program state

DORMANT : Indicates that the printer has not yet been started up in the designated area.

RUN : Indicates that a BASIC program is running.

WAIT : Indicates that a BASIC program is in a wait state.

STOP : Indicates that a BASIC program is not being executed. (*1)

3) PRIORITY : Current priority of a BASIC program.

If the STATUS is DORMANT, "0" is displayed.

4) STEP NO : Step number currently being executed.

If the STATUS is STOP, "0" is displayed.

4 "D>" appears after the execution result is displayed.

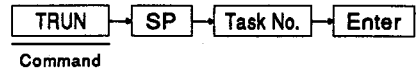
Input the necessary command.

*1 If a designated program is stopped by using the TSTOP debug command, the debugger will be in the STOP state.

5.2.2 Starting the execution of a designated BASIC program (TRUN command)

This operation starts executing a BASIC program in the designated task area.

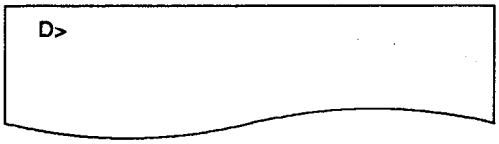
INPUT PROCEDURE (This command is also referred to as "TR")

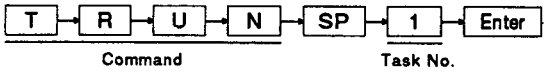


OPERATION EXAMPLE

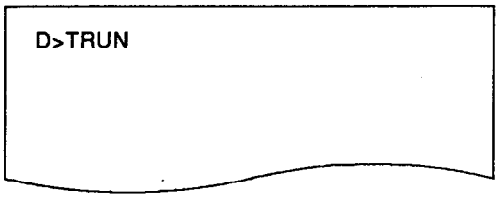
Starts executing a BASIC program in the task No.1 area.

Before the command is input

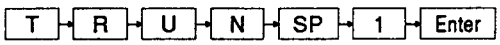


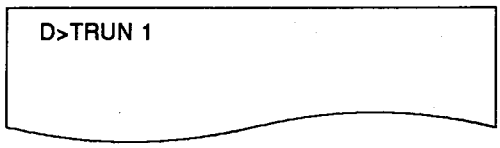


After the command is input



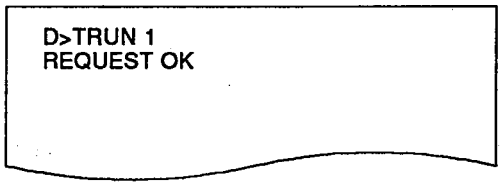
OPERATING PROCEDURE





1 Input the TRUN command and task number (AD51H-S3: 1 to 8, A1SD51S: 1, 2) to start executing a BASIC program.

(This example assumes that task No.1 is designated.)



2 The next line shows the execution result.

When the TRUN command is executed normally, the screen shows "REQUEST OK".

If the TRUN command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that the TRUN command is executed normally.)

3 "D>" appears after the execution result is displayed.
Input the necessary command.

(1) Precautions when inputting a command

- If a task number which corresponds to an area containing a program being executed is designated, then an error will occur.
  If a task number which corresponds to an area which contains no BASIC program is designated, then an error will occur.

(2) Reference

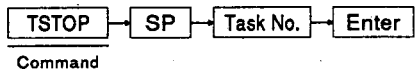
- Stopping the execution of a designated BASIC program ..... TSTOP command (see Section 5.2.3)

5.2.3 Stopping the execution of a designated BASIC program (TSTOP command)

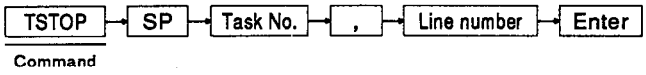
This operation stops the execution of a designated BASIC program in the designated task No. area.

INPUT PROCEDURE (This command is also referred to as "TP")

To stop the execution immediately:



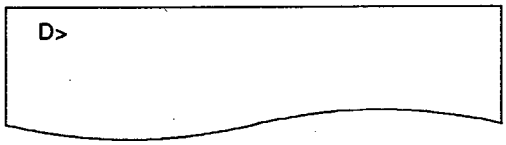
To stop the execution on a designated line:

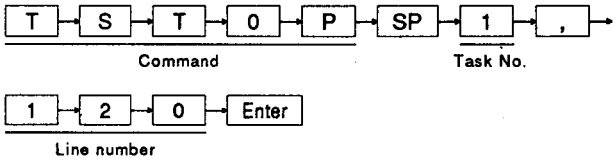


OPERATION EXAMPLE

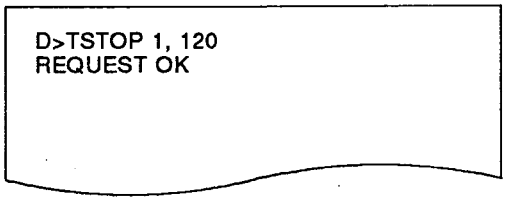
Stops the execution of a designated BASIC program (being executed in the task No.1 area) on line 120.

Before the command is input

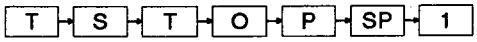


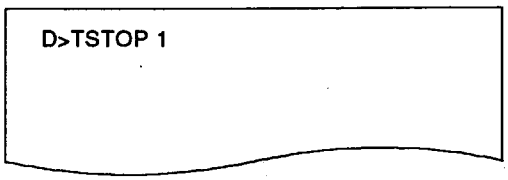


After the command is input



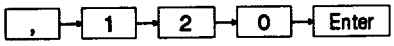
OPERATING PROCEDURE





1 Input the TSTOP command and task number (AD51H-S3: 1 to 8, A1SD51S: 1, 2) to stop the execution of a BASIC program.

(This example assumes that task No.1 is designated.)




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

2 Input the line number (where the execution will be stopped) in decimal.

When the program is stopped immediately, press the [Enter] key.

When designating "65535" or "–1" as the line number before this command is input, the next execution stop command is canceled.

(This example assumes that the execution is stopped on line 120.)

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

3 The next line shows the result of the stop command execution.

When the command is executed normally, the screen shows "REQUEST OK" along with the following stop message:

BREAK : Task No. [ ] Line No. [ ]

Line number where the execution is stopped

Task number of the stopped program

When designating "65535" or "–1" as the line number, giving the TSTOP command normally displays "BREAK Cancel: Task No.[ ]".

If the TSTOP command contains an error, an error message accompanied by the error code appears.

(This example assumes that the TSTOP command is executed normally.)

When the TSTOP command is executed normally, the corresponding program enters the STOP state.

Execution of the program can be resumed by giving the TCONTINUE command.

4 "D>" appears after the execution result is displayed. Input the necessary command.

(1) Precautions when inputting a TSTOP command

- When designating a line number, this number must exist in the program. Input this number in decimal. If a line number that does not exist in the program is input, the program cannot be stopped. In this case, designate "65535" or "–1", and re-input this command.
- A BASIC program can be stopped only at one position.
- When a BASIC program is stopped, make sure that operation does not also stop the system control.

(2) References

- Confirming the current BASIC program status ..... TSTATUS command (see Section 5.2.1)
- Retrying execution of a program from the first line ..... TRUN command (see Section 5.2.2)
- Resuming the execution from the interrupted line ..... TCONTINUE command (see Section 5.2.4)
- Confirming a variable ..... T? command (see Section 5.2.5)
- Assigning a value to a variable ..... TLET command (see Section 5.2.6)

(3) TSTOP command operations

- Stops the execution of a program at the step preceding the designated line when the TSTOP command along with a line number is input.

  If a TSTOP command without a line number is input, the program will not be executed. In this case, after the interpreter completes the command being executed when the [Enter] key was input, the execution stops.

  Therefore, if a line containing several commands (a multistatement) is input, any commands following the command being executed when the [Enter] key was input will not be executed.

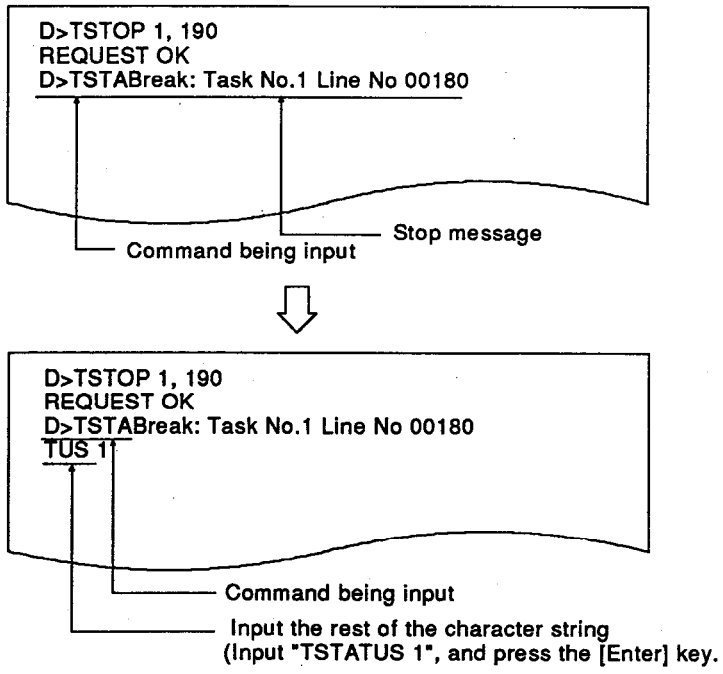
(4) Precautions when inputting the TSTOP command to stop a BASIC program

- When stopping the execution of a BASIC program, the debug (OS) displays the stop message on the line where the cursor is located.

  If a command is input when the debug displays the stop message, the message and the input command overlap on the screen.

  If this happens, continue inputting, since the input command is valid.

(Example) When the stop message appears during the TSTATUS command input



(5) How to stop the interpreter operation in the designated task No. area

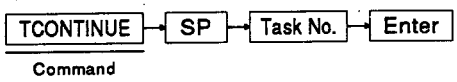
- When stopping the interpreter operation in the designated task No. area, create a program containing an END command.

5.2.4 Resuming a stopped BASIC program (TCONTINUE command)

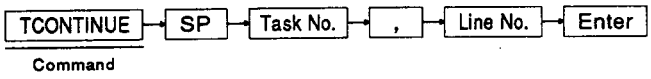
This operation resumes the execution of the BASIC program (in the designated task No. area) stopped by the TSTOP command.

INPUT PROCEDURE (This command is also referred to as "TC")

To resume the execution from the step following the last executed line:



To resume the execution after a command is stopped:

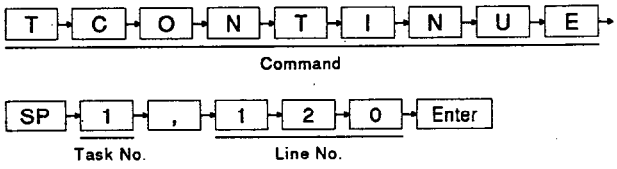


OPERATION EXAMPLE

Resumes execution of a BASIC program (in the task No.1 area) from line 120.

Before the command is input

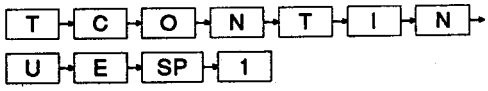
```
D>TCONTINUE 1, 120
```



After the command is input

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

OPERATING PROCEDURE




```
D>TCONTINUE 1
```

1 Input the TCONTINUE command and a task number (AD51H-S3: 1 to 8, A1SD51S: 1, 2) to resume a BASIC program stopped by the TSTOP command.
(This example assumes that task No.1 is designated.)

[,] → [1] → [2] → [0] → [Enter]

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

2 Input the line number (where the execution will be re-started) in decimal.

If the program is resumed from the step following the last number already executed, press the [Enter] key.

(This example assumes that line 120 is designated.)

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

3 The next line shows the result of the stop command execution.

When the command is executed normally, the screen shows "TCONTINUE: REQUEST OK".

If the command is not executed normally, "TCONTINUE: Error" accompanied by the error code appears.

(This example assumes that the command is executed normally.)

When the command is executed normally, the corresponding program enters the RUN state.

4 "D>" appears after the execution result is displayed.

Input the necessary command.

---

(1) Precautions when inputting a TCONTINUE command

- The TCONTINUE command can only be executed to a BASIC program stopped by the TSTOP command.

  If the command is executed to a program other rhan the program stopped by the TSTOP commsnd,an error will occur.

  When the communication module is set to editing mode (2) by designating a task No. area that contains a stopped BASIC program,this program cannot be resumes using the TCONTINUE command even if the communication module is returned to the debug mode using the SYSTEM command.

  The state of a BASIC program can be confilrmed by using ths TSTATUS command.

(2) Precautions when designating a line number

- Any designated line number must exist in the program.

  Input this number in decimal.

  If a line number that does not exist in the program is input,the pprogram will be resumed from the step following the last executed line.

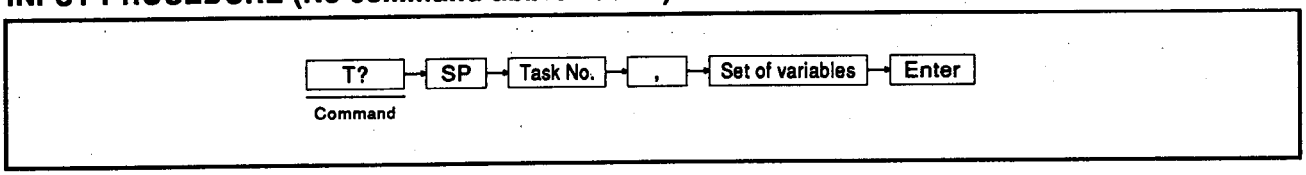
(3) References

- Confirming the current BASIC program status ..... TSTATUS command (see Section 5.2.1)
- Stopping the execution of a designated BASIC program..... TSTOP command (see Section 5.2.3)
- Resuming the execution from the first line..... TRUN command (see Section 5.2.2)

5.2.5 Displaying the value of a designated variable in a designated BASIC program (T? command)

This procedure displays designated variable values used in the BASIC program stored in designated task No. area.

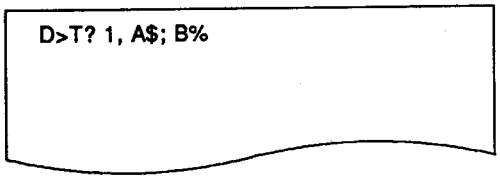
INPUT PROCEDURE (No command abbreviation)

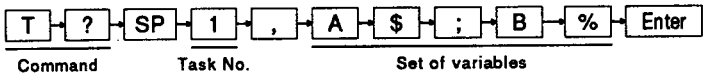


OPERATION EXAMPLE

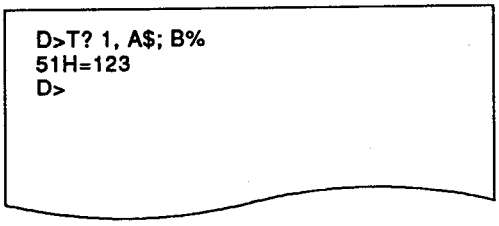
To display values of variables A$ and B% used in the BASIC program stored in task No.1 area with which execution is currently suspended:

Before the command is input

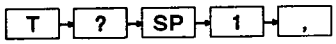


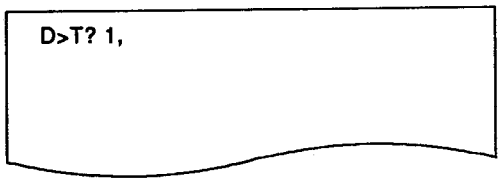


After the command is input



OPERATING PROCEDURE

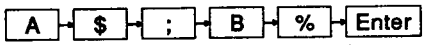


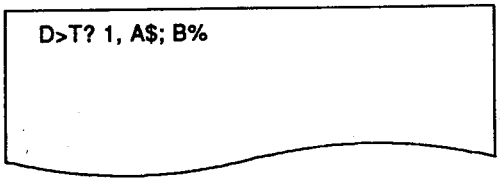


1 Input the T? command and a task number (AD51H-S3: 1 to 8, A1SD51S: 1, 2).

The task number must correspond to the area where a designated program exists.

(This example assumes that task No.1 is designated.)





2 Input the names of variables whose values are to be displayed.

Like the PRINT command, the T? command can be used with a numerical and character string expression.

By using commas (,) and semicolons (;), several variables can be displayed.

(This example assumes that the values of A$ and B% are displayed.)

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

3 The next line shows the result of the stop command execution.

When the command is executed normally, the screen shows "T?:" accompanied by the values of the variables.

If the command is not executed normally, a "T?: Error" message accompanied by the error code appears.

(This example assumes that the command is executed normally, which means that A$ and B% store "51H=" and "123" respectively.)

4 "D>" appears after the execution result is displayed.

Input the necessary command.

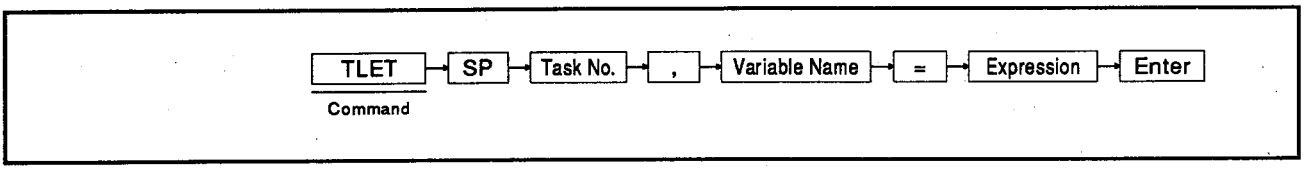
(1) Precautions when inputting commands
- If a BASIC program (that is set to DORMANT) is designated, then an error will occur.
- Designate the T? command along with its parameters in a line.
  The display caused by the T? command can consist of up to 1024 characters.
- Mitsubishi recommends that the BASIC program to be designated should be in the STOP state when the T? command is input.

(2) References
- Confirming the current BASIC program status ..... TSTATUS command (see Section 5.2.1)
- Stopping the execution of a designated BASIC program ..... TSTOP command (see Section 5.2.3)
- Resuming the execution of a stopped BASIC program ..... TCONTINUE command (see Section 5.2.4)
- Assigning a value to a variable ..... TLET command (see Section 5.2.6)

5.2.6 Assigning a value to the designated value in the BASIC program (TLET command)

This operation assigns a value to the designated value in the BASIC program.

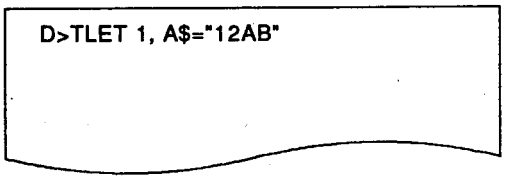
INPUT PROCEDURE (This command is also referred to as "TL")

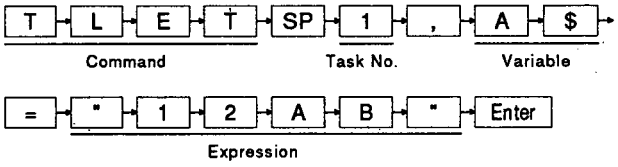


OPERATION EXAMPLE

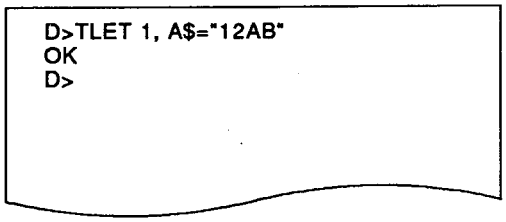
To assign values to variables A$ and B% used in the BASIC program stored in a task No. area with which execution is currently suspended:

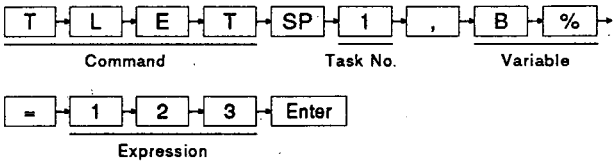
Before the command is input



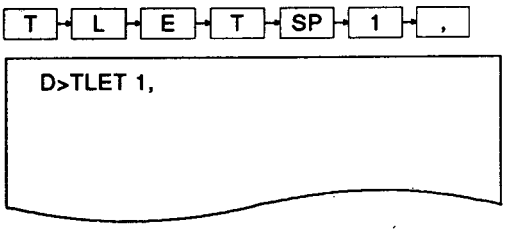


After the command is input



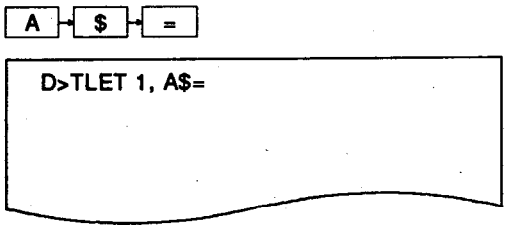


OPERATING PROCEDURE



1 Input the TLET command and a task number corresponds to the area where the program exists(AD51H-S3: 1 to 8, A1SD51S: 1, 2).

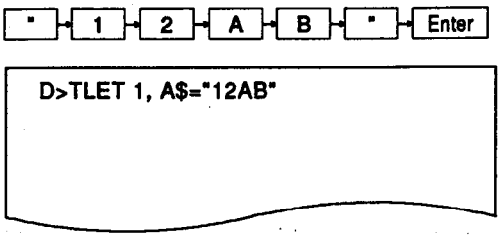
(This example assumes that task No.1 is designated.)



2 Designate the name of the variable to which a value is assigned.

Like the LET command, the TLET command can be used with an array variable name (ex. C(0), D$(1%), etc.).

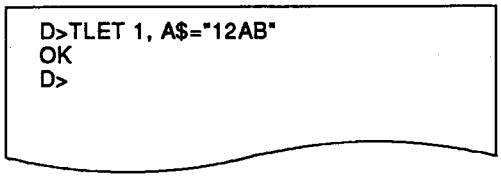
(This example assumes that the character variable A$ is designated.)



3 Input the value to be assigned to the variable.

Like the LET command, the TLET command can be used with a mathematical expression or character expression.

(This example assumes that the character constant of "12AB" is assigned to the A$ character variable.)



4 The next line shows the execution result.

When the TLET command is executed normally, the screen shows "OK".

If the TLET command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that the command is executed normally.)

5 "D>" appears after the execution result is displayed.

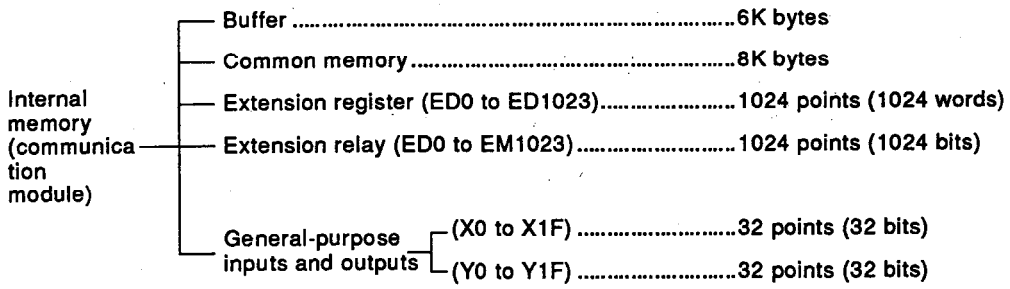
Input the necessary command.

(1) Precautions when inputting the TLET command
- If a BASIC program that is set to DORMANT is designated, an error will occur.
- Mitsubishi recommends that the BASIC program to be designated should be in the STOP state when the TLET command is input.

(2) References
- Confirming the current BASIC program status..... TSTATUS command (see Section 5.2.1)
- Stopping the execution of a designated BASIC program..... TSTOP command (see Section 5.2.3)
- Resuming the execution of a stopped BASIC program..... TCONTINUE command (see Section 5.2.4)
- Confirming a variable value..... T? command (see Section 5.2.5)

5.3 Reading/Writing from/to the Internal Memory

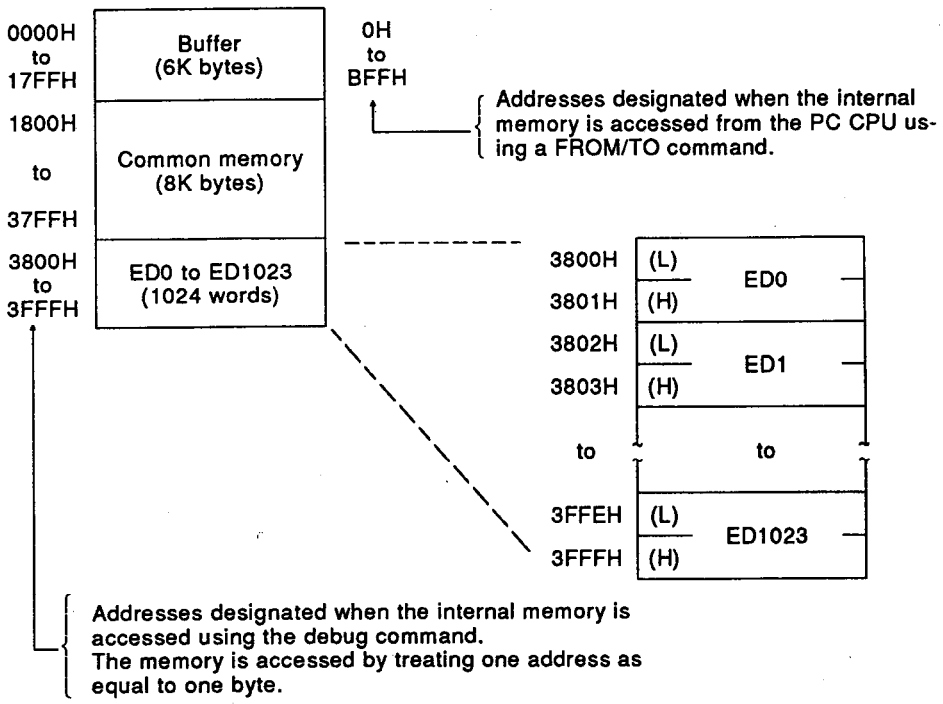
This section tells how to use debug commands when reading/writing from/to the internal memory.



The MREAD and MWRITE commands discussed in this section need designated addresses when the internal memory is accessed.

The correspondence of the addresses used with these commands to the internal memory is shown below:

Access the internal memory within the address ranges.

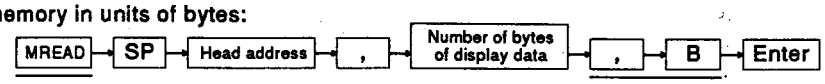


5.3.1 Displaying values in the buffer, common memory, and extension register ED (MREAD command)

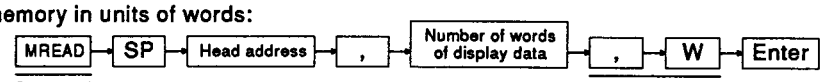
This procedure displays data stored in a designated memory (buffer, common memory, or ED).

INPUT PROCEDURE (This command is also referred to as "MR")

To display values in memory in units of bytes:



To display values in memory in units of words:



OPERATION EXAMPLE

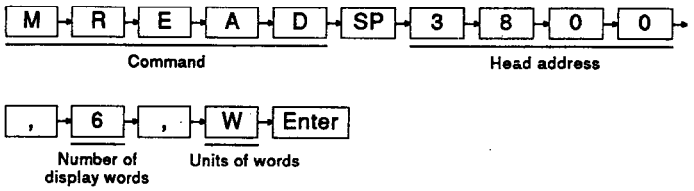
Displays values in extension registers ED0 to ED5 in units of words.

Before the command is input

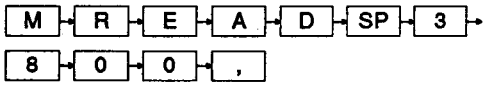
```
D>
```

After the command is input

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



OPERATING PROCEDURE




```
D>MREAD 3800,
```

1 Input the MREAD command and a head address (with which the display begins) using up to four digits in hexadecimal (0 to 3FFF).

Section 5.3 gives details about the correspondence of addresses to the device memory.

(This example assumes that address 3800H in ED0 is designated.)

When displaying data in units of words, set the first of the lower digits to a even number.

If the first of the lower digits is an odd number, the data in the designated address will not be displayed.

---

(1) Precautions when designating the number of bytes (words) of display data

When designating the number of bytes (words) of display data, the following conditions must be satisfied:

Address number + Number of display words/Number of display words −1 < 3FFFH

If a part of the memory above address 3FFFH is designated, the data in all addresses up to 3FFFH will be displayed.

6 → , → W → Enter

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

2 Designate the number of bytes (words) of display memory and the display type.

When designating "B" (can be omitted) as the display type, input the number of words of data in the designated memory range to be displayed.

When designating "W" as the display type, input the number of bytes of data in the designated memory range to be displayed.
Input the number of bytes (words) in hexadecimal.

When designating the number in units of bytes:
1H < (Number of bytes) < 4000H

When designating the number in units of words:
1H < (Number of words) < 2000H

(This example assumes that six words are designated.)

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

3 The next line shows the execution result.

When the MREAD command is executed normally, the data in the designated memory range is set in designated units.

When units of bytes is displayed, each line shows data in addresses [ ][ ][ ]0 to [ ][ ][ ]F (for 16 addresses) as shown below:

However, if the designated address ends with a number other than "0 (n)", spaces are placed for that address on the screen.

[ ][ ][ ][ ] : [ ][ ] [ ][ ] [ ][ ] [ ][ ] to [ ][ ] [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

- [ ][ ][ ][ ] : Indicates the address number.
- [ ][ ] [ ][ ] [ ][ ] [ ][ ] to [ ][ ]: Displays data for 16 addresses (address by address) in hexadecimal, dividing the data into sets of two columns.
- [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]: Display characters corresponding to each address. In the case of 0H to 1FH, "." is displayed.

When displaying data in units of words, each line displays the data for 16 addresses as shown below:

[ ][ ][ ][ ] : [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] to [ ][ ][ ][ ]

- [ ][ ][ ][ ] : Indicates the address number.
- [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ] [ ][ ][ ][ ]: Displays data for 16 addresses (address by address) in hexadecimal, dividing the data into sets of two addresses.

If the MREAD command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that the MREAD command is executed normally.)

4 "D>" appears after the execution result is displayed.

Input the necessary command.

(2) Operation when more than 16 lines of data is displayed
The screen can display 16 lines of data (max.).
If more than 16 lines of data must be displayed, pressing any key but the [Esc] key displays the additional data.
Pressing the [Esc] key clears the display.

(3) References
Writing a value to the designated memory ..... MWRITE command (see Section 5.3.2)
Confirming word information in extension register ED
..... W@ command (see Section 5.3.5)
Writing word information in extension register ED ..... W@ command (see Section 5.3.6)

5.3.2 Writing values to the buffer, common memory, or extension register (ED) (MWRITE command)

This operation writes values to the buffer, common memory, or extension register (ED).

INPUT PROCEDURE (This command is also referred to as "MW")

To write a value to the memory in units of bytes (one address):

MWRITE → SP → Head address → , → B → Enter

MWRITE: Command
B: Units of bytes (can be omitted)

To write a value to the memory in units of words (two addresses):

MWRITE → SP → Head address → , → W → Enter

MWRITE: Command
W: Units of words

OPERATION EXAMPLE

To write values (0AH, 14H, and 1EH) in word units to ED0 to ED2:

Before the command is input

```
D>
```

After the command is input

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

M → W → R → I → T → E → SP → 3 → 8 → 0 → 0
(Command: MWRITE; Head address: 3800)

, → W → Enter
(Write type: W)

0 → 0 → 0 → A → Enter
Writes a value to addresses 3800H to 3801H (ED0).

0 → 0 → 1 → 4 → Enter
Writes a value to addresses 3802H to 3803H (ED1).

0 → 0 → 1 → E → Enter
Writes a value to addresses 3804H to 3805H (ED2).

. → Enter
Write completed

OPERATING PROCEDURE

M → W → R → I → T → E → SP → 3 → 8 → 0 → 0 → ,

```
D>MWRITE 3800,
```

1 Input the MWRITE command and a memory address (with which the written value begins) using up to four digits in hexadecimal (0 to 3FFF).

Section 5.3 gives details about the correspondence of the device memory to the addresses.

(This example assumes that address 3800H in ED0 is designated.)

W → Enter

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

2 Designate the write value type.

Designate "B" (can be omitted) as the write value type, when writing the value in units of bytes.

Designate "W" as the write value type, when writing the value in units of words.

(This example assumes that the value are written in units of words.)

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

3 The next line shows the execution result.

When the MWRITE command is executed normally, the screen shows the designated addresses and their values as hexadecimal numbers in designated units.

When byte units are designated, the display is as shown below.

Input the value to be written using a one- or two-digit hexadecimal number. (Inputting only significant digits is valid.)

[0] → [0] → [0] → [A] → [Enter]

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

[ ][ ][ ][ ] : [ ][ ] [ ][ ]

- [ ][ ][ ][ ] — Input the value (for one address)
- [ ][ ] — Displays the memory value corresponding to the address using a hexadecimal number.
- [ ][ ] — Indicates the address number in hexadecimal

When word units are designated, the display is as shown below:

Input the value to be written using up to four digits in hexadecimal.

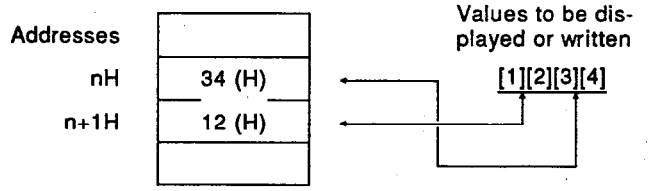
(Inputting only significant digits is valid.)

[0] → [0] → [1] → [4] → [Enter]

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

[ ][ ][ ][ ] : [ ][ ][ ][ ] : [ ][ ][ ][ ]

- [ ][ ][ ][ ] — Indicates the address number
- [ ][ ][ ][ ] — Displays memory values for two addresses using up to four digits in hexadecimal.
- [ ][ ][ ][ ] — Input the values (for two address) in hexadecimal.



[0] → [0] → [1] → [E] → [Enter]

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

Use the following keys to write values:

- [0] to [9], [A] to [F]: Used to input a value
- [.]: Used to end the write operation
- [\]: Used to move the address backwards.
- [Enter]: Used for inputting (designated using the above keys) or when not changing the current memory value

[.] → [Enter]

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

If the MWRITE command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that the value is written to memory addresses 3800H to 3805H (ED0 to ED2).)

4 "D>" appears after the execution result is displayed.
Input the necessary command.

(1) Processing when the device memory range is exceeded
- If a value is written to a device memory above 3FFFH, the MWRITE command is automatically stopped.

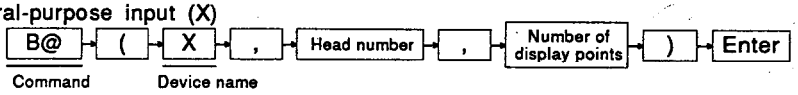
(2) References
- Confirming a value to the designated memory .... MREAD command (see Section 5.3.1)
- Confirming word information in extension register ED .....W@ command (see Section 5.3.5)
- Writing word information to extension register ED .....W@ command (see Section 5.3.6)

5.3.3 Displaying general-purpose input (X)/output (Y), or extension relay (EM) bit data (B@ command)

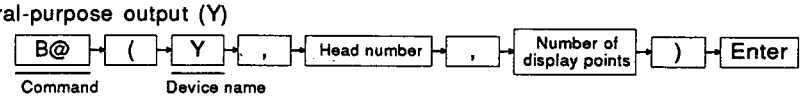
This operation displays bit data for general-purpose input (X)/output (Y) devices (used for communicating with a PC CPU), or an extension relay EM (used for data communications in a BASIC program).

INPUT PROCEDURE (No command abbreviation)

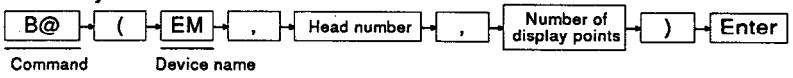
To display bit data for general-purpose input (X) devices:



To display bit data for general-purpose output (Y) devices:



To display bit data for an extension relay EM:

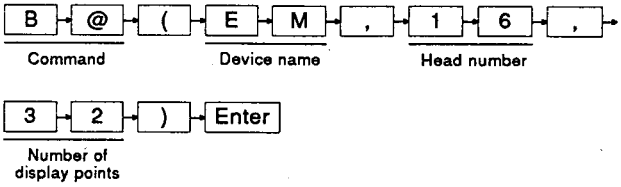


OPERATION EXAMPLE

To display bit data of EM16 to EM47:

Before the command is input

```
D>
```



After the command is input

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

(1) General-purpose I/O devices used for communications between a PC CPU and the communication module.
Those devices handle (a) bit data between a sequence program in a PC CPU and a BASIC program in the communication module, and (b) bit data controlled by the operating systems.

B → @ → (→ E → M → ,

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

1 Input the B@ command and the device name.

Input the device name as shown below:

X : When a general-purpose input is designated (PC CPU ← communication module)

Y : When a general-purpose output is designated (PC CPU → communication module)

EM : When an extension relay is designated

(This example assumes that an extension relay is designated.)

1 → 6 → ,

```
D>B@ (EM, 16,
```

2 Input the head number (with which the bit data display will begin).

When "X" or "Y" is designated, input the head number using a one- or two-digit hexadecimal number.

When "EM" is designated, input the head number using up to four digits in decimal.

X/Y: 0 to 1F, EM: 0 to 1023

(This example assumes that EM16 is designated as the head number.)

3 → 2 →) → Enter

```
D>B@ (EM, 16, 32)
```

3 Input the number of display points (bits) corresponding to the device range to be displayed in decimal or hexadecimal.

X/Y: 1(1H) ≤ "Number of display points" 32 ≤ (20H)

EM: 1(1H) ≤ "Number of display points" 1024 ≤ (400H)

(This example assumes that 32 points are designated.)

4 The next line shows the execution result.

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

When the B@ command is executed normally, the screen shows bit data in the designated device ranged.

When "X" or "Y" is designated:

Each line shows bit data of devices [ ][ ][ ]0 to [ ][ ][ ]F for 16 points as shown below:

When "EM" is designated:

Each line shows bit data of devices [multiple of 16] to [next multiple of 16] for 16 points as shown below:

"0" and "1" indicate OFF and ON respectively.

However, if the number designated as "X" or "Y" is not "0"(n), or the number designated as "EM" is not a multiple of 16, spaces are placed in the display columns corresponding to devices 0 or a multiple of 16 to (designated number – 1).

[ ][ ]: [ ][ ] to [ ][ ] – [ ][ ] to [ ][ ]

- [ ][ ]: Number; Device name
- [ ][ ] to [ ][ ]: Displays bit data for eight bits from the number shown on the left using "0" or "1".
- [ ][ ] to [ ][ ]: Displays bit data for the next eight bits using "0" or "1".

If the B@ command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that the B@ command is executed normally.)

5 "D>" appears after the execution result is displayed.

Input the necessary command.

(1) Precautions when designating the number of display points
- When designating the number of display points, the following conditions must be satisfied:
  X/Y ..... Number + Number of display points −1 ≤ 1F (H)
  EM ..... Number + Number of display points −1 ≤ 1023
- If the designated number is outside the device range, device data will be displayed until the last device number is reached.

(2) Operation for displaying more than 16 lines of data
- The screen can display 16 lines of data (max.).
  If more than 16 lines of data must be displayed, pressing any key but the [Esc] key displays the additional data.
- Pressing the [Esc] key clears the display.

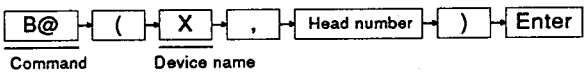
(3) Reference
- Writing bit data to extension relay EM ..... B@ command (see Section 5.3.4)

5.3.4 Writing bit data to general-purpose input signal (X) and extension relay (EM) (B@ command)

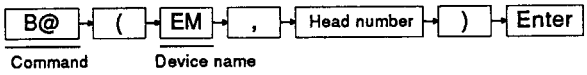
This operation writes bit data to general-purpose input signal (X) (output to a PC CPU) or to extension relay (EM) (used for data communications between BASIC programs).

INPUT PROCEDURE (No command abbreviation)

To write bit data to general-purpose signal (X):



To write bit data to extension relay (EM):



OPERATION EXAMPLE

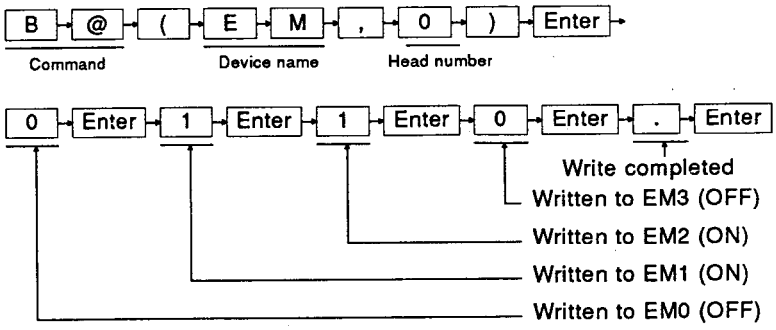
Writes bit data (0, 1, 1, 0) to devices EM0 to EM3.

Before the command is input

```
D>
```

After the command is input

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



OPERATING PROCEDURE


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

1 Input the B@ command and device name.

Input the device name as shown below:

X : Name when a general-purpose input signal is designated (PC CPU ← communication module)

EM : Name when an extension relay (EM) is designated

(This example assumes that the extension relay (EM) is designated.)

(1) General-purpose I/O devices used for communications between an PC CPU and the communication module

- Those devices handle (a) bit data between a sequence program in a PC CPU and a BASIC program in the communication module, and (b) bit data controlled by the operating systems.

(2) Precautions when using the B@ command

- To operate the communication module normally, do not write bit data to general-purpose input signal devices X0B to X0F.

0 →) → Enter

```
D>B@ (EM, 0)
EM0000 : 0 0
```

2 Input the device number with which the write operation will begin.

When "X" is designated, input the head number using a one- or two-digit hexadecimal number.

When "EM" is designated, input the head number using up to four digits in decimal.

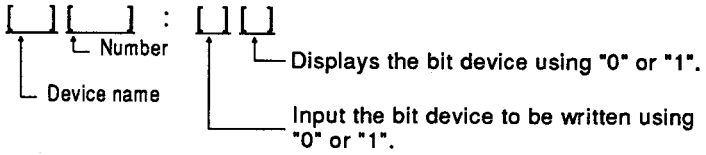
X: 0 to 1F, EM: 0 to 1023

(This example assumes that EM0 is designated.)

3 The next line shows the execution result.

When the B@ command is executed normally, the screen shows bit data corresponding to the specification using "0" and "1".

Input the bit data to be written using "0" and "1".



Use the following keys to write bit data:

[0] : Used to turn OFF the corresponding bit

[1] : Used to turn ON the corresponding bit

[\] : Used to move the corresponding bit backwards

[.] : Used to end the write operation

If the B@ command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that the B@ command is executed normally.)

4 "D>" appears after the execution result is displayed.

Input the necessary command.

(1) Processing when the device memory range is exceeded

- When the device memory range in which bit data is written is exceeded, the B@ command is automatically stopped.

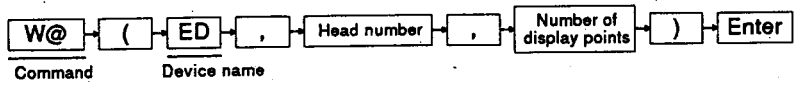
(2) Reference

- Confirming word data in extension relay (EM) ..... B@ command (see Section 5.3.3)

5.3.5 Displaying word data in extension register (ED) (W@ command)

This operation displays word devices in extension register (ED) that are used for data communications between BASIC programs.

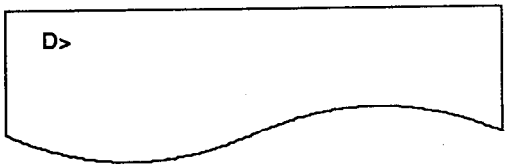
INPUT PROCEDURE (No command abbreviation)



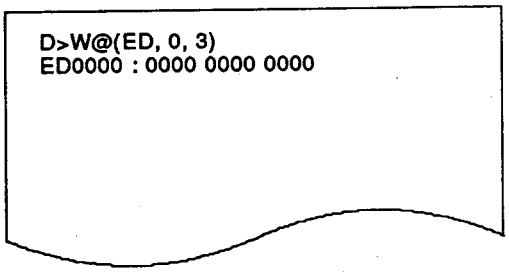
OPERATION EXAMPLE

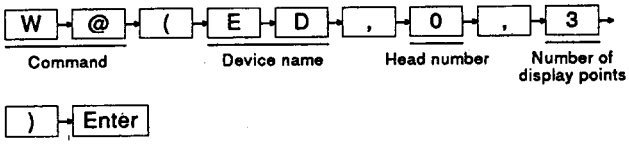
Displays word data (values) in devices ED0 to ED2.

Before the command is input

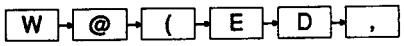


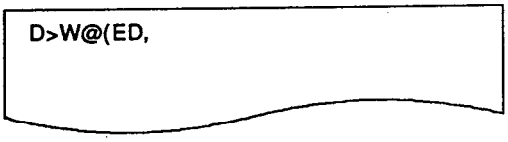
After the command is input



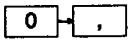


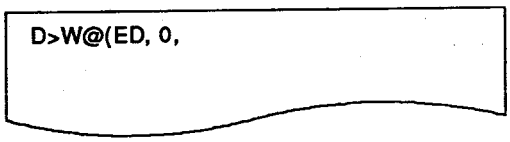
OPERATING PROCEDURE





1 Input the W@ command and the internal device name.





2 Input the head number (with which the word display will begin) using up to four digits (0 to 1023) in decimal.

(This example assumes that ED0 is designated.)

3 →) → Enter

3 Input the number of display points (words) in decimal.

ED: 1 ≤ Number of display points ≤ 1024

(This example assumes that three points are designated.)

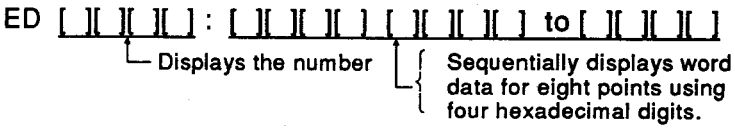
```
D>W@(ED, 0, 3)
ED0000 : 0000 0000 0000
```

4 The next line shows the execution result.

When the W@ command is executed normally, the screen shows word data corresponding to the designated range.

Each line shows bit data of devices (a multiple of eight) to (the next multiple of eight) for 8 points in hexadecimal as shown below:

However, when the designated number is not a multiple of eight (n), spaces are placed in the display columns corresponding to devices (a multiple of eight) to (designated number – 1).



If the W@ command is not executed normally, "W@: Error" accompanied by the error code appears.

(This example assumes that the W@ command is executed normally.)

5 "D>" appears after the execution result is displayed.

Input the necessary command.

(1) Precautions when designating the number of display points
- When designating the number of display points, the following condition must be satisfied:
  ED: Head number + Number of display points –1 ≤ 1023
- If the designated number exceeds ED1023, device data will be displayed up to ED1023.

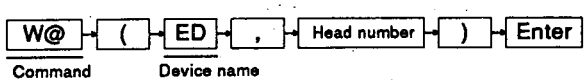
(2) Operation when more than 16 lines of data is displayed
- The screen can display 16 lines of data (max.).
  If more than 16 lines of data must be displayed, pressing any key but the [Esc] key displays the additional data.
- Pressing the [Esc] key clears the display.

(3) References
- Confirming values in the designated memory (requires the address specification) ..... MREAD command (see Section 5.3.1)
- Writing values to the designated memory (requires the address specification) .....MWRITE command (see Section 5.3.2)
- Writing word data in extension register (ED) .....W@ command (see Section 5.3.6)

5.3.6 Writing word data to extension register (ED) (W@ command)

This operation writes word data to extension register (ED) that are used for data communications between BASIC programs.

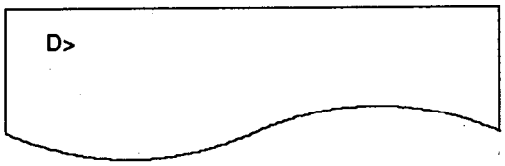
INPUT PROCEDURE (No command abbreviation)



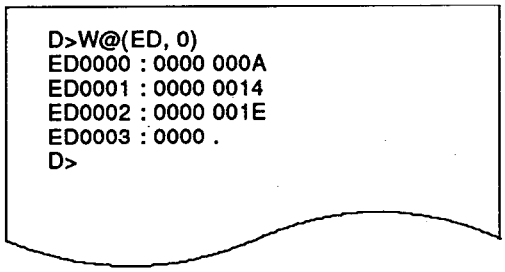
OPERATION EXAMPLE

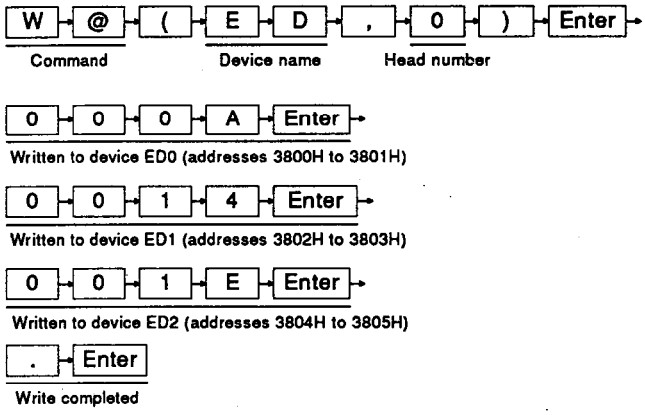
Writes word data (0AH, 14H, 1EH) to devices ED0 to ED2.

Before the command is input

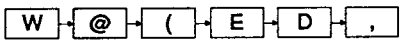


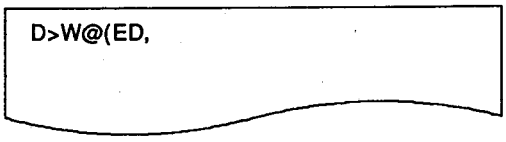
After the command is input





OPERATING PROCEDURE





1 Input the W@ command and the internal device ED name.

(1) Writing to internal devices ED
Word data can be written to internal devices ED using the MWRITE command as well.

0 →) → Enter

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

2 Input the ED number (with which word data writing begins) using up to four digits (0 to 1023) in decimal.

(This example assumes that ED0 is designated.)

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

3 The next line shows the execution result.

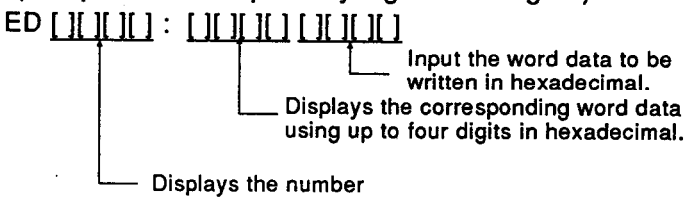
When the W@ command is executed normally, the screen shows the designated device number and word data as shown below:

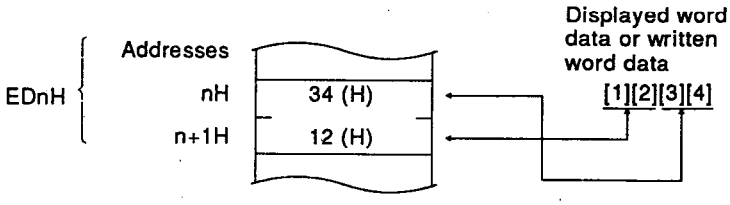
Input word data (to be written) using up to four digits in hexadecimal.

(It is possible to input only significant digits.)

0 → 0 → 0 → A → Enter

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





0 → 0 → 1 → 4 → Enter

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

Use the following keys to write bit data:

[0] to [9], [A] to [F] : Used to input word data

[\] : Used to move the corresponding number backwards

[.] : Used to end the write operation

[Enter] : Used to input when the above-mentioned keys are used. Used to leave the current data as it is.

0 → 0 → 1 → E → Enter

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

If the W@ command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that word data is written to devices ED0 to ED2 (Addresses 3800H to 3805H).)

. → Enter

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

4 "D>" appears after the execution result is displayed.

Input the necessary command.

(1) Operation when the designated number exceeds ED1023

- If the designated number exceeds ED1023, device data will be displayed up to ED1023

(2) References

Confirming values in the designated memory (requires the address specification) ..... MREAD command (see Section 5.3.1)

Writing values to the designated memory (requires the address specification) ..... MWRITE command (see Section 5.3.2)

Writing word data to extension register (EM) ..... W@ command (see Section 5.3.5)

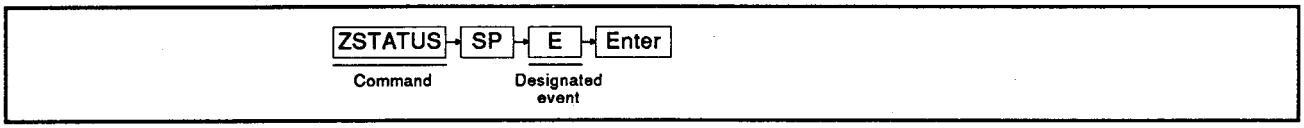
5.4 Confirming the State of Events, Message Ports, and Source Numbers

This section tells how to use the commands to confirm the current states of events, message ports, and source numbers shared by BASIC programs.

5.4.1 Displaying event declaration states (valid/invalid) (ZSTATUS command)

This operation displays the current declaration states (valid/invalid) of events (shared by BASIC programs) for each event.

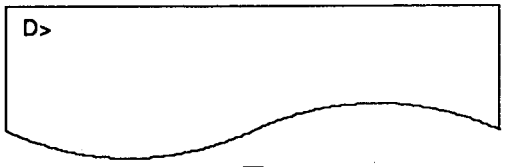
INPUT PROCEDURE (This command is also referred to as "ZS")

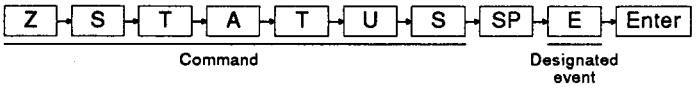


OPERATION EXAMPLE

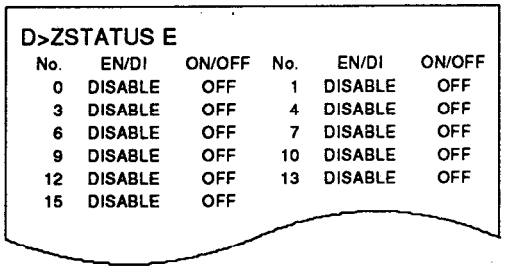
Displays the current declaration states (valid/invalid) for each event.

Before the command is input

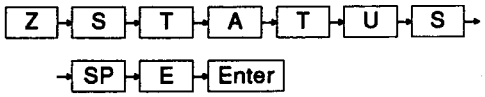




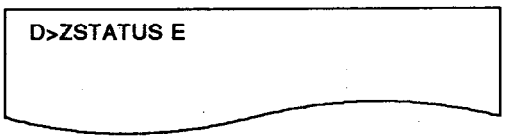
After the command is input



OPERATING PROCEDURE



1 Input the ZSTATUS command and "E" (to designate the display data type as an event)



(1) How to use events
Events are used with the following commands:
(The AD51H-BASIC Programming Manual (Command) gives details.)

- Defining an event ..... DEF ZEVENT command
- Declaring an event valid or invalid ..... ZEVENT command
- Generating an event ..... ZSIGNAL command
- Waiting to generate an event ..... ZWAIT EVENT command

```
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 execution results are displayed.

When the B@ command is executed normally, the next lines show whether each event (0 to 63) is valid or invalid.

The display contents are given in the following table:

EN/DI	ON/OFF	Meaning
ENABLE	ON	Event (corresponding to the number) is already defined and declared as valid.
ENABLE	OFF	Event (corresponding to the number) is already defined, but not declared.
DISABLE	ON	Event (corresponding to the number) is already defined and declared as invalid.
DISABLE	OFF	Event (corresponding to the number) is not defined.

If the B@ command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that the B@ command is executed normally.)

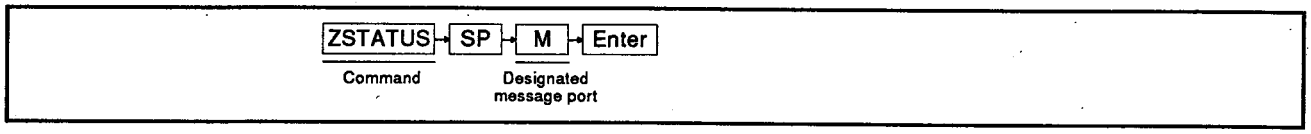
3 "D>" appears after the execution result is displayed.

Input the necessary command.

5.4.2 Displaying the state of a message transmitted to a message port shared by BASIC programs (ZSTATUS command)

This operation displays the state of transmission of a message at each message port which is shared by BASIC programs.

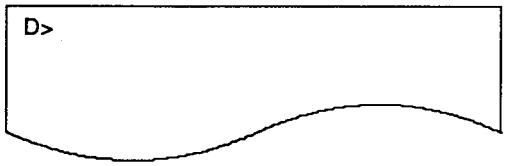
INPUT PROCEDURE (This command is also referred to "ZS")

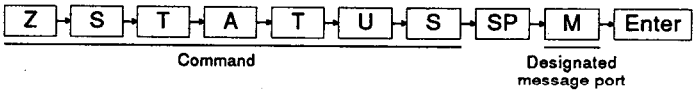


OPERATION EXAMPLE

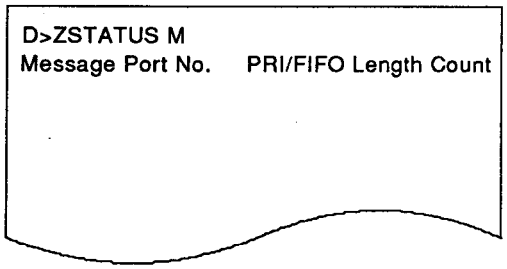
To display the current state of message transmission:

Before the command is input

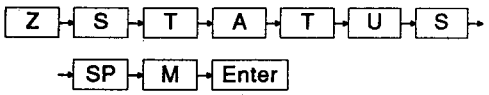




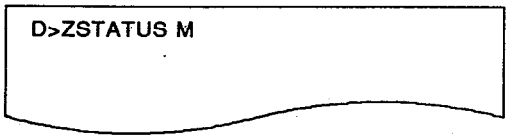
After the command is input



OPERATING PROCEDURE



1 Input the ZSTATUS command and "M" to display the transmission state of the message port.



(1) Message transmission via a message port
Messages can be transmitted/received between BASIC programs by defining the message port in the BASIC programs.
The AD51H-BASIC Programming Manual (Command) gives details.
All related commands begin with "ZMESSAGE".

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

2 The execution results are displayed.

When the ZSTATUS command is executed, the next lines show the transmission states (information about unreceived messages) for each message port.

The display contents are shown below:

Message Port N : Message port number defined by the user.

PRI/FLSO : Shows the type of corresponding message port as follows:

PRI : "FIFO" designated when defining the port.

FIFO : "FIFO" not designated when defining the port.

Length : Byte length when defining the corresponding message port.

Count : Number of unreceived messages that were transmitted to the corresponding port.

If the ZSTATUS command is not executed normally, the next line shows an error message along with the error code.

(This example assumes that the ZSTATUS command is executed normally.)

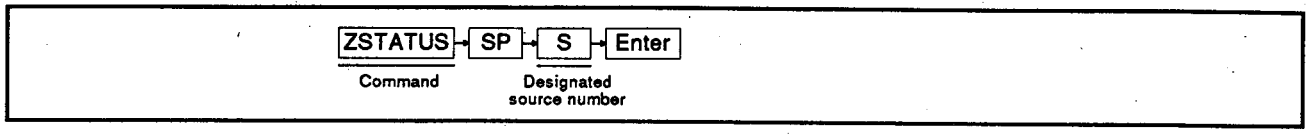
3 "D>" appears after the execution result is displayed.

Input the necessary command.

5.4.3 Displaying the reserve/release states of source numbers used for exclusive control (ZSTATUS command)

This operation displays the reserve/release states of source numbers used for exclusive control of memory and external devices.

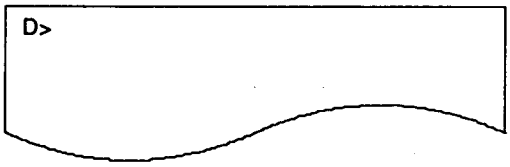
INPUT PROCEDURE (This command is also referred to as "ZS")

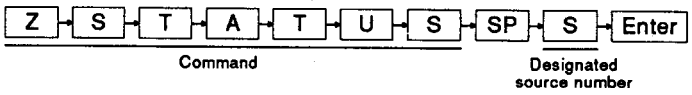


OPERATION EXAMPLE

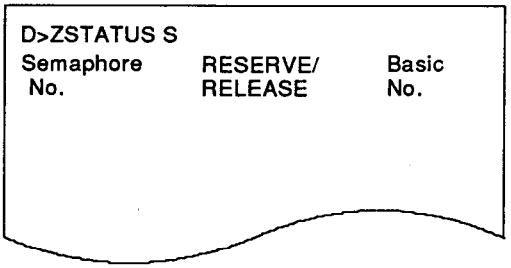
Displays the reserve/release state of the current source number.

Before the command is input

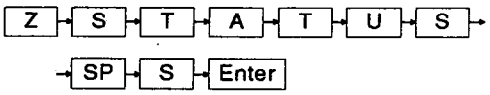




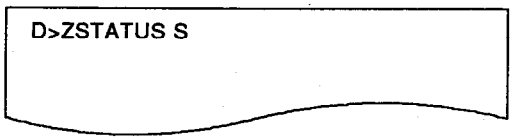
After the command is input



OPERATING PROCEDURE



1 Input the ZSTATUS command and "S" to display the state of the source number.



(1) Exclusively controlling a source by reserving/releasing its source number
When executing several BASIC programs at the same time, the BASIC programs uses the following commands exclusively to control the sources.
The AD51H-BASIC Programming Manual (Command) gives details.
Reserving a source number ..... ZRESERVE command
Releasing a source number..... ZRELEASE command

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

2 The execution results are displayed.

When the ZSTATUS command is executed normally, the next lines show the reserve/release states for source numbers 0 to 31.

The display contents are shown below:

Semaphore No. : Source number

RESERVE/RELEASE: Indicates the reserve/release state corresponding to the source.

RESERVE : Indicates the source is in the reserve state.

RELEASE : Indicates the source is in the release state.

Basic No. area : Corresponding source No.

If the ZSTATUS command is not executed normally, an error message accompanied by the error code appears.

(This example assumes that the ZSTATUS command is executed normally.)

3 "D>" appears after the execution result is displayed.

Input the necessary command.

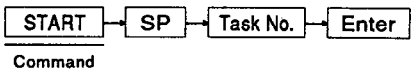
5.5 Changing the Communication Module Mode

This section tells how to use debug commands to change the mode of the communication module.

5.5.1 Setting the communication module to editing mode (2) (START command)

This operation sets the communication module to the editing mode to edit (create, change) another program using the designated task area when several BASIC programs are executed.

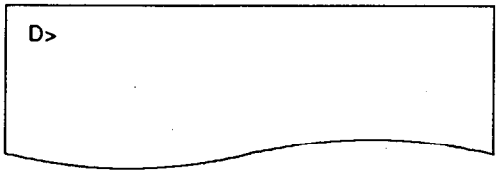
INPUT PROCEDURE (This command is also referred to as "ST")

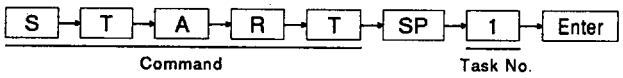


OPERATION EXAMPLE

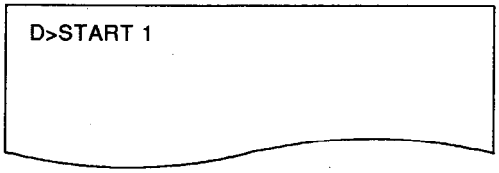
Sets the communication module to editing mode (2) to modify a BASIC program (whose execution is stopped) in the task No.1 area.

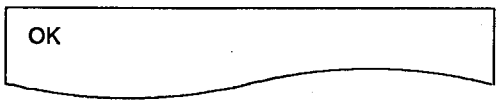
Before the command is input





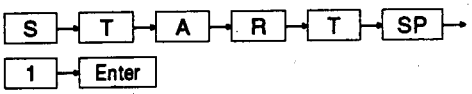
After the command is input

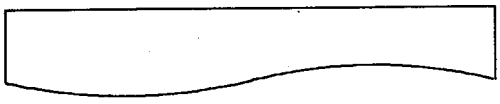




When the interpreter was not started, the following message appears before "OK".
"AD51H-BASIC ON-LINE PROGRAMMING Ver [ ]"

OPERATING PROCEDURE





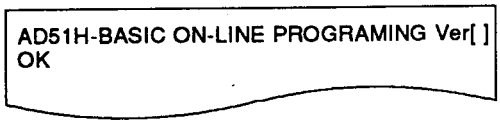
1 Input the START command and a task number (corresponding to the program to be edited) (AD51H-S3: 1 to 8, A1SD51S: 1, 2) .

The task number can be omitted.

When the task number is omitted, the START command is designated as shown below:

1) When the START command is initially input, the task number is automatically set to 1.
2) When the START command is not initially input, the previously-designated task number is used.

(This example assumes that task No.1 is designated.)



2 The execution results are displayed.

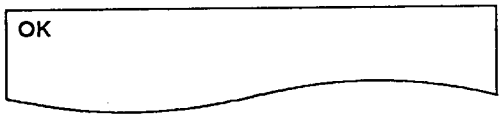
When the START command is executed normally, the screen shows the display contents indicated on the left.

Thereafter, start editing the BASIC program.

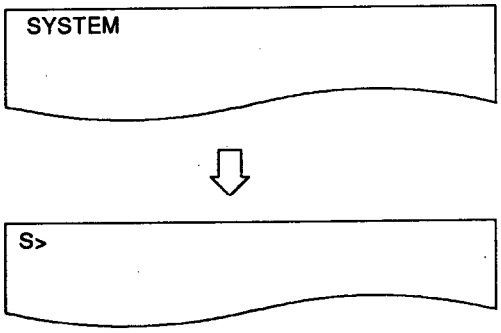
The programming manual tells how to edit a BASIC program.

If the START command is not executed normally, an error message accompanied by the error code appears.

The display contents when the interpreter was not started are shown on the upper left.



The display contents when the interpreter was started are shown on the lower left.



3 After completing the BASIC program, do either 1) or 2) below when returning the communication module from editing mode (2) to the debug mode:

1) Execute the SYSTEM command.
   - Stops the BASIC program execution.
   - Closes the open files and communications line.
2) Press the [Ctrl] + [D] keys.
   - Stops the BASIC program execution.
   - Leaves the open files and communications line as they are.
   - If the BASIC program was not modified, the execution of the program can be resumed using the TCONTINUE debug command.
     When resetting the communication module to the editing mode, the execution can be resumed using the CONTINUE command.

(1) Precautions when using the START command

- When a BASIC program is being executed in the task No. area used for editing another program, stop the program using the TSTOP command.

(2) Other BASIC program operations when the START command is executed

- When several BASIC programs are being executed, even if an edit operation in a task No. area is started, programs in other task No. areas will continue to be executed.

(3) Necessary measures for changing designated task sizes

- Set the communication module to editing mode (1) and do the following:

  1) Return the communication module to the debug mode using the SYSTEM command.
  2) Stop the BASIC programs in the task No. areas using the TSTOP command so that the operation does not also stop the system control.
  3) Use the GO command to set the communication module to the system mode.
  4) Use the TKILL system command to end the interpreter operation in the task No. areas.
  5) Use the START command to set the communication module to editing mode (1). After giving the START command, change the task size, and edit the program.

Section 2.3 gives the mode change chart.

(4) Reference

- Changing the communication module mode .....GO command (see Section 5.5.2)

5.5.2 Setting the communication module to the system mode, execution mode (2), or debug mode (GO command)

This operation switches the debug mode to the system mode or execution mode (2), and vice versa.

By setting the communication module to the system mode, it is possible to give system commands to the console to edit/debug a BASIC program (exists in a task area)(see Section 4).

(1) When the communication module switches to execution mode (2), setting multitasking starts the execution of a BASIC program.

Returning the communication module to the debug mode restarts the debug operation.

Setting multitasking starts the execution of the BASIC program.

The following table gives the relationship between the mode/debug start specifications (when the GO command is input) and the console/debugger states (after the GO command is input):

<table>
<tr><th>Mode Setting</th><th>Debug Start Specification</th><th>Console State</th><th>Debug Terminal State</th><th>Remarks</th></tr>
<tr><td rowspan="2">R
(Execution Mode (1))</td><td>YES
(to the debug mode)</td><td rowspan="2">Clears the display contents.
Set to the console used for running the BASIC program.</td><td>Restarts the debug, clears the display contents, and displays "D>".
In this state, the debug command can be input.</td><td rowspan="2">Setting multitasking reloads the BASIC program to the corresponding task No. area and starts the execution.</td></tr>
<tr><td>NO
(to execution mode (2))</td><td rowspan="2">The display contents remains unchanged.</td></tr>
<tr><td>P
[Programming in the system mode]</td><td>Cannot be designated.</td><td>Clears the display contents, and displays "S>".
The system commands can be input.</td><td>Stops the execution of BASIC programs in the task No. areas.</td></tr>
</table>

(1) BASIC program states when the GO command is executed

- When execution mode (1) is designated, if mode switch (1) is set to 0 to 3, the BASIC program starts in the same way when the communication module starts up.
- When the system mode is designated, the execution of all BASIC programs in the task No. areas is stopped.

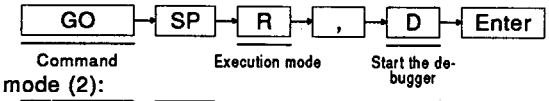
  (All BASIC programs but the program in the DORMANT state enter the STOP state)

  Since all task No. areas (in the main memory) conditions are unchanged, the BASIC programs in the task No. areas remain as they are.

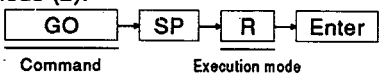
  By changing the communication module from the system mode to editing mode (1), it is possible to edit/debug a BASIC program in a task No. area.

INPUT PROCEDURE (No command abbreviation)

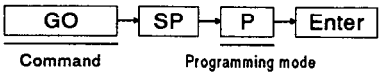
To reset the communication module to the debug mode:



To set the communication module to execution mode (2):



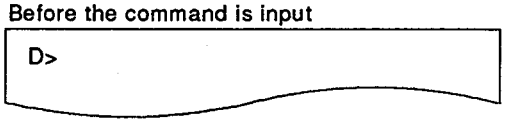
To set the communication module to the system mode (one of the programming modes):

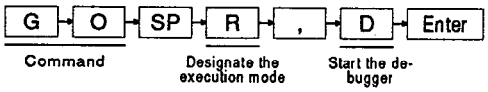


OPERATION EXAMPLE

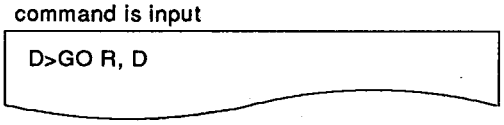
Returns the communication module to the debug mode.

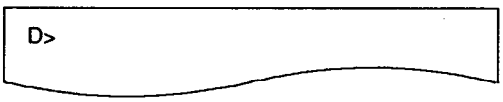
Before the command is input



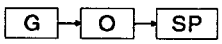


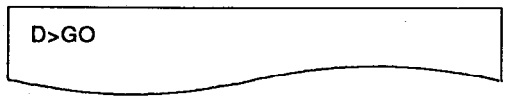
After the command is input



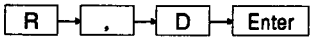


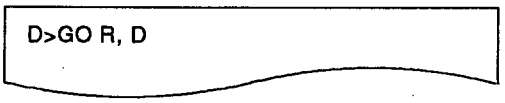
OPERATING PROCEDURE





1 Input the GO command to change the communication module command.





2 Designate the mode.
Input "R" to set the communication module to execution mode (2) or the debug mode.
Input "P" to set the communication module to the system mode.
When "R" is designated, to return the communication module from execution mode (2) to the debug mode, input "D" following the "R".
(This example assumes that the communication module is returned to the debug mode.)

(2) Precautions when changing the mode
When switching the communication module from the debug mode to another mode, Mitsubishi recommends that the execution of BASIC programs should be stopped (see the

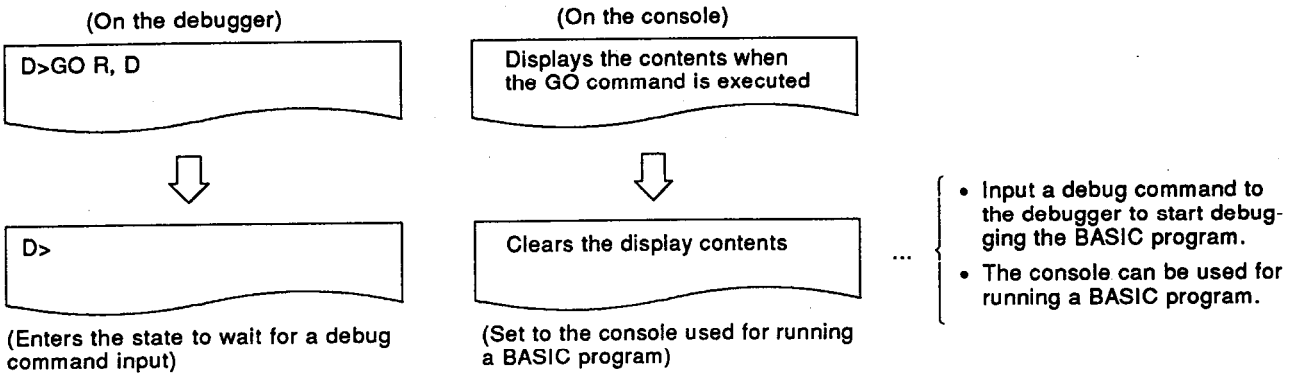
3 The result of the GO command execution is displayed.

When the command is executed normally, the screen shows the following:

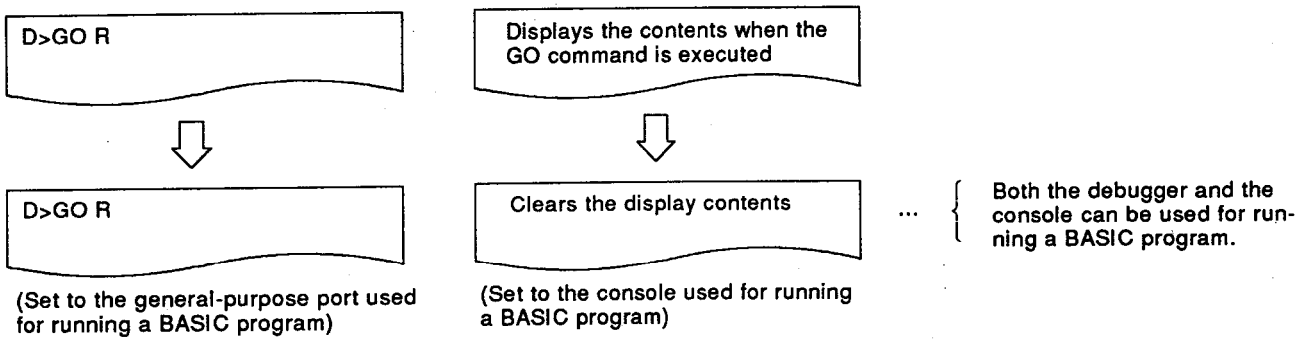
If the GO command is not executed normally, an error message accompanied by the error code appears.

The following example gives the display contents when the GO command is executed normally:

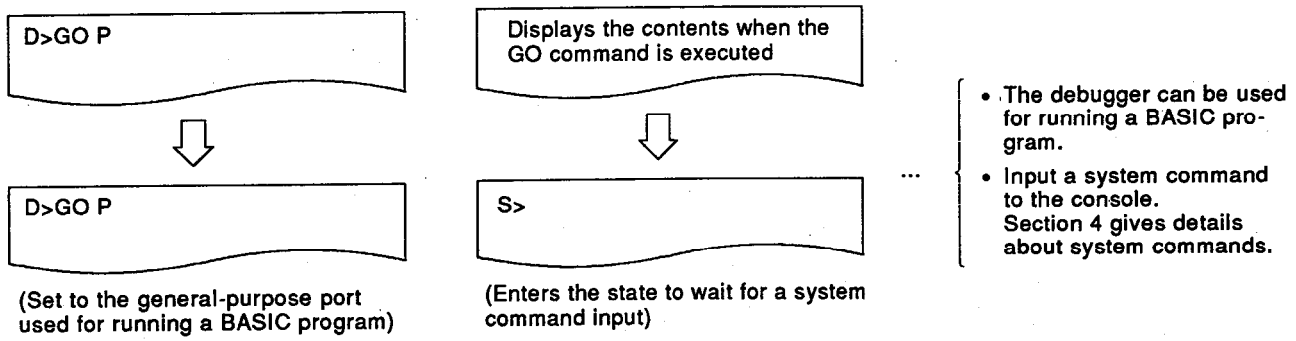
(1) When returning the communication module to the debug mode:



(2) When setting the communication module to execution mode (2):



(3) When setting the communication module to the system mode:



(3) Communication module mode changes
- Section 2.3 gives the communication module mode change chart.

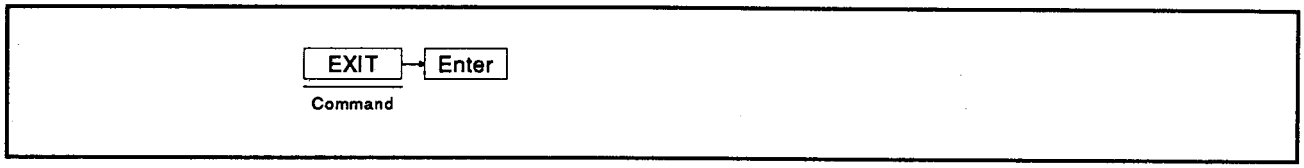
(4) Reference
- Displaying the MAIN MENU on the debugger .....EXIT command (see Section 5.6)

5.6 Displaying the MAIN MENU on the Debugger (EXIT Command)

This section tells how to use the EXIT command to display the MAIN MENU (contained in the AD51H-BASIC software package) when a PC/AT is used as the debugger,

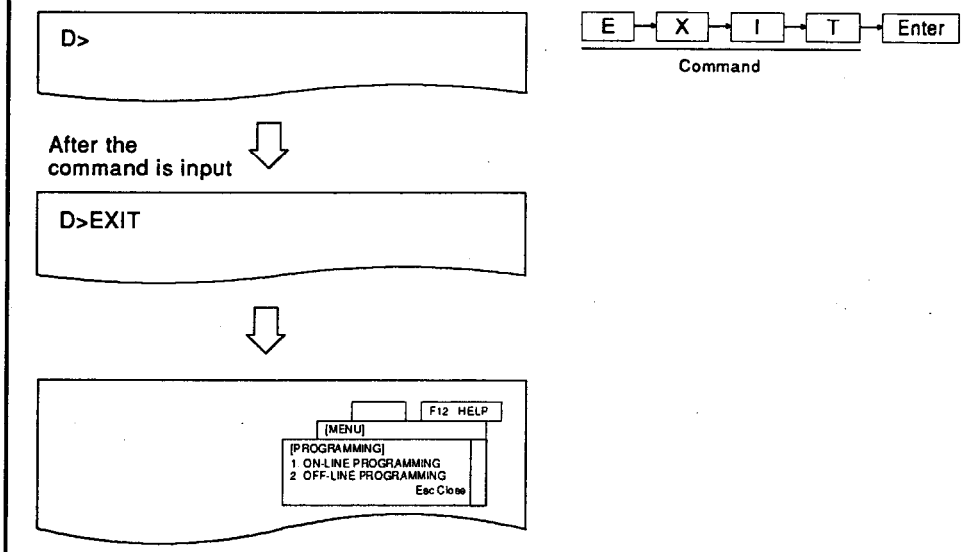
When a VG-620 or a VT-382/VT-220 is used as the debugger, pressing any key redisplays "D>" after the EXIT command is input.

INPUT PROCEDURE (This command is also referred to as "E")

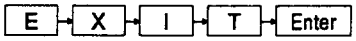


OPERATION EXAMPLE

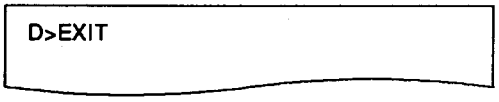
Displays the MAIN MENU on the PC/AT used as the debugger.
Before the command is input



OPERATING PROCEDURE



1 Input the EXIT command to display the MAIN MENU.



(1) BASIC program states when the EXIT command is executed
Even if the EXIT command is executed, execution of BASIC programs in the task No. area continues.

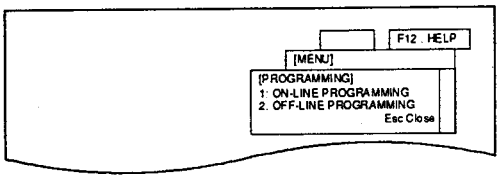
(2) Precautions when inputting the TSTOP command
When a BASIC program in a task No. area is edited using the menu screens, give the TSTOP command to stop the execution of the BASIC programs to prevent the system control from being stopped.

2 After the execution result is displayed, do the corresponding operation.

When the EXIT command is executed normally, the screen shows the following contents:

If the EXIT command is not executed normally, an error message accompanied by the error code appears.

[When a PC/AT is used]



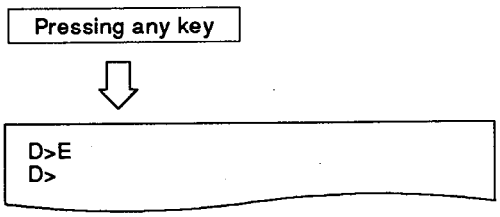
The following display contents appear when the command is executed normally:

1) When a PC/AT is used as the debugger:

The MAIN MENU appears on the debugger. Select one of the items from the MAIN MENU.

The SW1IX-AD51HPE AD51H-BASIC Operating Manual gives details about menu-driven operations.

[When a VG-620 or a VT-382/VT-220 is used]



2) When a VG-620 or a VT-382/VT-220 is used as the debugger:

The debugger enters the state to wait for a key input.

Pressing any key displays "D>". Input a debug command.

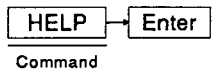
(3) Communication module mode changes
- Section 2.3 gives the communication module mode change chart.

(4) References
- Stopping the execution of a designated BASIC program ..... TSTOP command (see Section 5.2.3)
- Changing the communication module mode ..... GO command (see Section 5.5.2)

5.7 Confirming the Input Procedure for Debug Commands (HELP Command)

This section tells how to use the HELP command to display on the debugger the input procedure for debug commands.

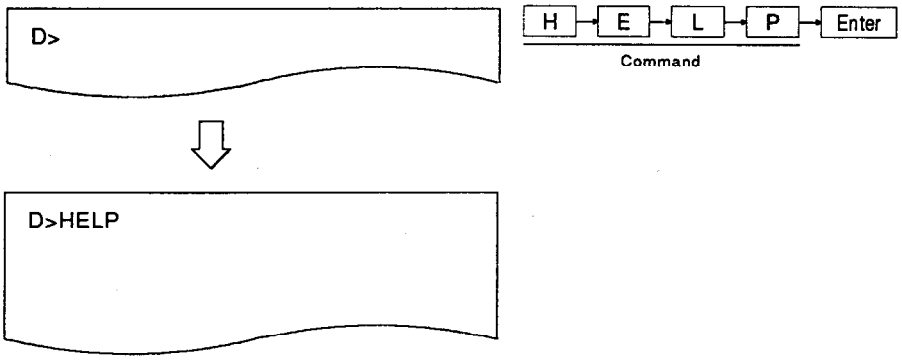
INPUT PROCEDURE (This command is also referred to as "H")



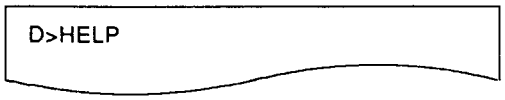
OPERATION EXAMPLE

Displays the input procedure for debug commands.

Before the command is input



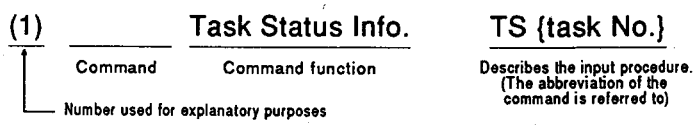
OPERATING PROCEDURE



1 Input the HELP command to display the input procedure for debug commands.

2 The result of the command execution is displayed.
When the HELP command is executed normally, the next lines show the types, functions, and input procedure for debug commands.

(Example)



If the HELP command is not executed normally, the next line shows an error message.

3 After displaying the result of the command execution, "D>" appears.
Input the necessary command.

(1) Description of the command input procedures

A space (located following a command) for one column indicates pressing the [SP] (space) key.

A pair of parentheses "(" and ")" indicates inputting parentheses.

A pair of braces "{" and "}" indicates an argument. However, they do not actually have to be input.

A pair of brackets ("[" and "]") indicates "can be omitted". However, they do not actually have to be input.

6. CREATING BASIC PROGRAMS USING A GENERAL-PURPOSE EDITOR

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

Before using a general-purpose editor, read this section to familiarize yourself with the restrictions that apply.
When using the SW1IX-AD51HPE type software package, it is not necessary to read this section.

6.1 Difference between a General-Purpose Editor and the Software Package

The communication module can use BASIC programs created either by using the SW1IX-AD51HPE type software package or a general-purpose editor.

The differences between a general-purpose editor and the software package from the point of view of BASIC program creation are as follows.

General-purpose editor ..... BASIC programs can be edited offline.
Operation can not be checked while creating a program.

Software package ..... BASIC programs can be created either offline or online.
In online programming, the program can be run and the operation checked during editing.

6.2 Operation Flow when Creating a BASIC Program Using a General-Purpose Editor

The procedure from the creation of a BASIC program with a general-purpose editor to its execution is presented below.

(1) When using a PC/AT

(a) Select DOS from the basic utility menu.

(b) Start up the general-purpose editor. ..... For the method used to start up each general-purpose editor, see Section 7.4.

(c) Create the BASIC programusing the general-purpose editor.

(d) Save the BASIC program. ..... Program saving destination and extension file name:
C: \AD51H\USR\[ ][ ][ ].BAS

(e) Quit the general-purpose editor.

(f) Start up the line number tool and assign line numbers to the program. ..... Input DRENUM name to assign the line numbers.

(g) Start up the software package.

(h) The program is executed using the interpreter and its operation is checked.

6.3 Items Required for Program Creation

In order to create a program using a general-purpose editor, the editor software must be procured.
Some representative general-purpose editors are listed below.

- Representative general-purpose editors

  MIFES ..... from Megasoft

  FINAL ..... from ASP

  EDLIN ..... from the Microsoft Corporation

> For details on how to install the general-purpose editor in your HD, refer to the instruction manual for the general-purpose editor.

6.4 Starting up the General-Purpose Editor

This section explains how to start up the general-purpose editor.

6.4.1 Starting up MIFES

This is the procedure for starting up MIFES.
For detailed information on MIFES operations, refer to the MIFES instruction manual.

(1) Creating a new program

Input the following at the MS-DOS command line:

MI[Enter] or MI file name [Enter]

If you start up with MI[Enter], input the file name after MIFES has started up.
The file will be saved under the specified file name.
Specify the storage destination for a PC/AT file as "C:\AD51H\USR\USRfile name".

(2) Modifying a program

Input the following at the MS-DOS command line:

MI[Enter] or MI file name [Enter]

If you start up with MI[Enter], input the file name after MIFES has started up.
The file will be saved under the selected file name.

6.4.2 Starting up FINAL

This is the procedure for starting up FINAL.
For detailed information on FINAL operations, refer to the FINAL instruction manual.

(1) Creating a new program

Input the following at the MS-DOS command line:

FE[Enter] or FE file name [Enter]

If you start up with FE[Enter], input the file name after FINAL has started up.
The file will be saved under the specified file name.
Specify the storage destination for a PC/AT file as "C:\AD51H\USR\file name".

(2) Modifying a program

Input the following at the MS-DOS command line:

FE[Enter] or FE file name [Enter]

If you start up with FE[Enter], input the file name after FINAL has started up.
The file will be saved under the selected file name.

6.4.3 Starting up EDLIN

This is the procedure for starting up EDLIN.
For detailed information on EDLIN operations, refer to the EDLIN instruction manual.

(1) Creating a new program

Input the following at the MS-DOS command line:

EDLIN file name [Enter]

The file will be saved under the specified file name.
Specify the storage destination for a PC/AT file as "C:\AD51H\USR\file name".

6.5 Notes on Using a General-Purpose Editor

The points that should be observed when using a general-purpose editor are listed below.

(1) Processing at the end of program lines

Always enter CR (&H0D) and LF (&H0A) at the end of a program line. (When using a general-purpose editor, these are automatically entered on pressing the Return key and Enter key respectively.)

(2) BASIC program file end processing

Enter E0F (&H1A) at the end of a basic program file.
(When using a general-purpose editor, an E0F code is usually input automatically.)

(3) Control codes in the program

If control codes are included in the program, it will not operate as a normal program.

(4) Designation of the PRINT command

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

(5) Assigning line numbers

Assign line numbers at the head of each line, in ascending order.

(6) Number of characters in one line

The number of characters in one line should not exceed 254.
("One line" means everything up to the end of a line.)

6.6 Assigning Line Numbers with the Line Number Tool

This section explains how to start up the line number tool and lists the points to note when using it.
The line number tool is included in the SW1IX-AD51HPE type software package.

6.6.1 Starting up the line number tool

To assign line numbers to or modify a program created using a general-purpose editor, the line number tool must be started up.

The following is the entry used to start up the line number tool and a guide to setting each of the options.

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

-s XXX	Input the new starting line number in place of XXX. If no setting is made, "10" is automatically set.
-t XXX	Input the old starting line number in place of XXX. If no setting is made, the head line number is automatically set.
-i XXX	Input the increment value place of XXX. If no setting is made, "10" is automatically set.
-e XXX	Input the line number for the end of the alteration in place of XXX. If no setting is made, the final line is set.
Source file name	Specify the BASIC source file name. If no extension file name is specified, it assumed to be ".BAS".
Output file name	Specify the name of the output file that results after outputting the line numbers. If no specification is made, the output file name is generated by changing the source extension file name to ".BAS". If the main file name of the output file is specified but the extension file name is not, it assumed to be ".BAS". The extension file name of the source file is changed to ".OLD".

POINT

Always use lower case letters for option settings.

DRENUM -s 10 -t 100 TEST.BAS

Lower case

An example of the flow when line numbers are assigned to a program created using a general-purpose editor is presented below.

(a) Create a program using the general-purpose editor. (Program with no line numbers assigned)

Source file name: C:\AD51\USR\TEST.BAS

```
Branch at the ' condition
INPUT "X =" ; X
IF X>=0 AND X<=10 GOTO *OK ELSE *ERROR
*OK
PRINT "Within range 0 to 10"
END
*ERROR
PRINT "Outside range"
END
```

(b) Save the program

(c) Quit the general-purpose editor.

(d) Check that the program has been created.

(e) Start up the line number tool.

```
C: \>DRENUM TEST. BAS

C: \>c:

C: \>cd \ad51h\system\drenum TEST. BAS

C: \AD51H\USR>
C: \AD51H\USR>
```

(f) Completion of line number assignment ..... Line number assignment is completed.

(g) Check the program ..... The source program is saved with the extension file name ".OLD".

```
10 Branch at the ' condition
20 INPUT "X = " ; X
30 IF X>=0 AND X<=10 GOTO *OK ELSE *ERROR
40 *OK
50 PRINT "Within range 0 to 10"
60  END
70 *ERROR
80 PRINT "Outside range"
90 END
```

6.6.2 Notes on the line number tool

The points that should be observed when using the line nunber tool are listed below.

(1) Processing of same file numbers

If the source file name and output file name are the same, processing is executed after changing the extension file name of the source file name to ".OLD". If a file with the extension file name changed to ".OLD" already exists, it is overwritten.

(2) Number of characters in one line

If the number of characters in a line of the source file, or in a line after line numbers have been assigned, exceeds 254, an error occurs.

(3) Program syntax errors

If a syntax error occurs, the line number may not be correctly assigned to the line in which the error occurred.

(4) Cases where line numbers are not reassigned

Note that line numbers will not be reassigned in the following cases.

(a) Line numbers of other programs

Line number of line from which program execution is to start when using the CHAIN command

Example:

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

However, line numbers specified for the DELETE option are reassigned.

(b) Line numbers of commands that cannot be used in the program

Line numbers of AUTO, DELETE, LIST, LLIST, MERGE, RENUM commands.

Example

```
LIST 100
     ↑
Not reassigned
```

(c) Processing in the event of a line number tool error

If processing is forcibly suspended during execution of line number tool operations, or if file reading/writing or renaming fails due to an I/O error, the source file is processed in the following way.

- If processing has not yet reached the stage of renaming the file, the source file remains unchanged.
- If the source file has already been renamed, it remains as it is.

(d) Work files used by the line number tool

The file names shown below are those of files temporarily created by the line number tool and should therefore not be used by the user.

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

7. CREATING BASIC PROGRAMS USING A COMPILER

This section explains how to create a BASIC program using a compiler.

> When using interpreter BASIC, it is not necessary to read this section.
> Use compiler BASIC after reading this section and confirming restricted items, etc.

7.1 Differences between Compiler BASIC and Interpreter BASIC

The communication module can use both compiler BASIC and interpreter BASIC.

The difference in execution between compiler BASIC and interpreter BASIC is as follows:

Compiler BASIC ..... Translates programming language into machine language before the program is run, and then a communication module executes the machine language program.

Interpreter BASIC ..... Translates programming language into machine language and runs it at the same time while a communication module is executing the program.

Neither is objectively better or worse: both have advantages and disadvantages as shown below.

	Advantages	Disadvantages
Compiler BASIC	• Fast execution .	• Debugging is difficult. • Many detailed restrictions.
Interpreter BASIC	• Debugging is easy.	• Slower execution.

Select compiler or interpreter according to the purpose to be fulfilled.

7.2 Flow when Creating a Program Using a Compiler

This section how to create a BASIC program, how to translate it using a compiler, and how to execute it using a communication module.

1) Start the software package.

   ..... Start the SW1IX-AD51HPE type software package.

2) Create a BASIC program by online programming.

3) Execute it by using an interpreter, and confirm the operation.

   ..... Confirm whether it operates in the restricted range of a compiler.

4) Store the program to the hard disk by using the SAVE instruction.

   ..... Store the program to the C drive of the hard disk (designated drive number 3).

5) Terminate the software package.

   ..... The screen returns to the DOS command line.

6) Compiler execution

   ..... Input and compile DBC <file name>.

7) Confirm the executable program (~.EXE) file.

   C:\AD51H\USR>DIR/W

   ..... Confirm whether the ~.EXE file has been created after executing the compilation.

8) Restart the software package.

   ..... Start the SW1IX-AD51HPE type software package.

9) Select online programming from the menu and transfer to the system mode.

10) Use the MSAVE command to save it to the execution area of the memory card.

    ..... MSAVE <task ID>,, file name

11) Use the SET command to change the start condition to START.

    ..... Change the start condition from BOOT to START.

12) Execution

    ..... To execute in the programming mode, input GO R[,D].

    To execute in the execution mode, set the mode setting switch 1 to 0 or 1, and reset the communication module.

7.3 Necessary Items for Compiling

An assembler and a linker are necessary for compiling programs created by a communication module. But, since they are not included in the SW1IX-AD51HPE software package, they must be purchased separately.

The recommended assembler and linker are shown below.

- Mitsubishi recommends the following:

Microsoft Macro Assembler Ver.4.0 or 5.1 (supporting AX)

Since linkers are not sold separately, use a linker (Microsoft linker) after Version 3.5 such as the linker contained in MASM or MS-DOS.

7.4 Registering Assemblers and Linkers to a Hard Disk

This section explains how to register assemblers and linkers to a hard disk.

Only those files (MASM.EXE, LINK.EXE) that are necessary for a compilation by using a communication module are copied.

1) Start

2) Insert the Microsoft MASM assembler disk in drive A.

3) Use the MS-DOS COPY command to copy MASM.EXE to the hard disk.

C:\>COPY A:\MASM.EXE C:\[Enter]

4) After copying is completed, take out the floppy disk, and insert the MASM utility disk in drive A.

5) Use the MS-DOS COPY command to copy LINK.EXE to the hard disk.

C:\>COPY A:\LINK.EXE C:\[Enter]

6) After copying is completed, take out the floppy disk.

7) Completed

7.5 Starting the Compiler

It is necessary to start the compiler to compile a created program. Compilation can be executed by inputting DBC<file name> name on MS-DOS.

How to set the compiler format and the available options are shown below.

DBC [-4] [-v] [-w-] [-d] Source file name [.BAS] [Loading file name.EXE]

Source file name	BASIC source file name. If an extension has not been designated, the source file name extension is considered to be '.BAS'. If there is a system name, designate the system name and file name.
Loading file name.EXE	Compiled loading file name. If a file name has not been designated, the source file name extension is replaced by '.EXE', and the file name becomes a loading file name.
-4	Designated when using Ver. 4.0. If this has not been designated, Ver 5.1 is considered to be used.
-v	States during compiling are fully displayed.
-w-	Warnings are not displayed.
-d	A program is compiled to check the following errors when executing an operation. (For debugging) • When adding, subtracting, or multiplying an integer, errors are checked for. If an overflow is detected, an Overflow error is generated. • When executing an operation, the ranges of the values of the array's subscripts are checked. If a value is out of range, a Subscript out of range error is generated. However, if this option is designated, the size of the execution program is enlarged, and the execution speed decreases.

POINT

Always use lower case letters for option settings.

DBC -v -w TEST.BAS

Lower case

The following example shows the flow for compiling (file name: COMP.EXE) a program (file name: INTER.BAS) created by an interpreter.

1) C:\>DBC -v INTER.BAS COMP.EXE [Enter]

.....Start compilation. Designate -v option.

C:\>c:

C:\>cd\ad51h\usr

2) C:\AD51H\USR>c:\ad51h\system\dbc -Lc:\ad51h\system -v INTER.BAS COMP.EXE

BASIC COMPILER Ver 1.0

.....Start the compiler.

masm $1.asm;

Microsoft (R) macro Assembler Version 5.10

Copyright (C) Microsoft Corp 1981, 1988. All right reserved.

xxxxx Bytes symbol free

0 Warning Errors

0 Severe Errors

3) Link c:\ad51h\system\dbb.obj $1,comp,/map,c:\ad51h\system\dbc.1ib

.....Start a linker.

Microsoft (R) Segmented-Executable Linker Version 5.01.20

Copyright (C) Microsoft Corp 1984-1988. All rights reserved.

Definitions File[NUL.DEF]:[Enter]

.....Input the ENTER key.
This is not always displayed.

LINK: warning L4021: no stack segment

.....Ignore this error.

C:\AD51H\USR>

4) C:\AD51H\USR> .....Compilation completed.

7.6 Precautions when Compiling

(1) Compiling

Always use a Microsoft Macro Assembler when compiling, since pro grams cannot be compiled by using any other assembler.

(2) Workpiece filed created by the compiler

Since, the following file names are files created by the BASIC compiler, the user cannot utilize them.

$n.ASM	assembler source file "n" indicates the numerical.
$n.OBJ	object file
BC.TMP	workpiece file
BASIC$$$.INC	include file

(3) Errors during compiling

Errors during compiling are stored in the assembler source file $n.ASM and also displayed on the screen. Confirm errors by referring to this file.

(4) Program operation confirmation

Before compiling a program, execute it using an interpreter, and confirm proper operation. Since programs cannot be edited after compiling, if an error occurs after compiling, that program must be modified by an interpreter and must be compiled again.

(5) Errors when the DBC compiler is used

If the DBC compiler is used, LINK: warning L4021:no stack segment is displayed during the link. However, this message does not create any difficulty in the operation of a compiled program. Therefore, ignore this error.

(6) Variable sizes

The interpreter consumes only a character-string area of the length of a character and the compiler always consumes 256 bytes per variable.

(7) Order in which an expression is executed

To increase execution speed, the compiler optimizes expressions. Therefore, the priority level of the expression and the combination rule do not change. However, the order in which items in an expression are executed may not be the same.
For example, in ASC(INKEY$)-ASC(INKEY$)*2, it cannot be judged which is executed first. If the result of an expression differs from that of the interpreter, execute the expression after dividing it and storing the middle value to the variable.

(8) Integer operation upgrading

If the middle result of an expression is not in the range of an integer in the addition, subtraction, multiplication, or division of an integer, it is upgraded automatically to a real number by the interpreter, and the calculation is executed. However, the operation is executed in the range of the integer by the compiler. In this case, use the CSNG or CDBL function to convert an integer value into a real number.

7.7 Execution Using a Communication Module

To execute a compiled program using a communication module, the compiled program must be registered in the execution area of a memory card/EEP-ROM.
The procedure for registration to the execution area of a memory card/EEP-ROM is shown below.

MSAVE <Task ID>[,[V],"< File name>" [,[Location]] or

MSAVE <Task ID>[,V]

Task ID	BASIC task ID that is stored in the execution area AD51H-S3: 1 to 8 A1SD51S : 1, 2
V	Specify verification. After writing is completed, the contents of the main memory and the memory card/ EEP-ROM are verified.
"File name"	Specify the compiled file name (~.EXE) that is read to the main memory/EEP-ROM.
Location	Specify the position where a task is allocated. 0, 16, 32,...368 (Increased by 16) If it is abbreviated, the allocation is automatically executed.

POINTS

- The 'Error: Location' error, which shows that a location cannot be allocated, will sometimes occur when MSAVE is executed. When this happens, either designate a vacant location and execute MSAVE or set the start conditions of all tasks to OFF and execute MSAVE.
- The 'System: code =824' error, which shows that there is no work area, will sometimes occur when MSAVE is executed. When this happens, set the start conditions of all tasks to OFF, and execute MSAVE of all tasks again.

For instance, when saving a compiled file (example: COMP.EXE) by using MSAVE to position 32 of task ID 1, the procedure is as follows:

```
1) S>MSAVE 1,,"3:COMP.EXE",32 [Enter]
                                .......Saving a file to the execution area of
                                       task 1.
   SAVE(Y/N)? Y                 .......Select Y.
   SAVE OK
   S>                           .......Completed.
2) S>SET 1,START [Enter]
                                .......Changing the start condition from
                                       'BOOT' to 'START'.
   SET OK
   S>
3) S>GO R [Enter]               ..... Transeferring to the execution mode.
```

7.8 Instructions and Functions

7.8.1 Compilability of instructions and functions

The compilability of instructions and functions is shown in Table 7.1 below.

o: Compilable

Δ: Compilable with restrictions

x: Non-compilable

Table 7.1 Compilability of Instructions and Functions

Instructions and Functions	Compilability	Remarks	Section
ABS	o		*
ASC	o		*
ATN	Δ	With restrictions	S7.8.2-1,
AUTO	x	Non-compilable	S7.8.2-2,
BEEP	o		*
BIN$	Δ	With restrictions	S7.8.2-3,
BSWAP	o		*
CDBI	o		*
CDBL	o		*
CHAIN	x	Non-compilable	S7.8.2-4
CHR$	o		*
CIDB	o		*
CINT	o		*
CISN	o		*
CLEAR	x	Non-compilable	S7.8.2-5
CLOSE	o		*
CLS	o		*
COM ON/OFF/STOP	o		*
COMMON	x	Non-compilable	S7.8.2-6
CONSOLE	o		*
CONT	x	Non-compilable	S7.8.2-7
COS	Δ	With restrictions	S7.8.2-8
CSNG	o		*
CSNI	Δ	With restrictions	S7.8.2-9
CVD	o		*
CVDMBF	o		*
CVI	o		*
CVS	o		*

*: See Section 11 of AD51H-BASIC (Command) Programming Manual.

Table 7.1 Compilability of Instructions and Functions (continued)

Instructions and Functions	Compilability	Remarks	Section
CVSMBF	o		*
DATA	Δ	With restrictions	S7.8.2-10
DATE$	o		*
DEFDBL	Δ	With restrictions	S7.8.2-11
DEF FN	Δ	With restrictions	S7.8.2-12
DEFINT	Δ	With restrictions	S7.8.2-13
DEFSNG	Δ	With restrictions	S7.8.2-14
DEFSTR	Δ	With restrictions	S7.8.2-15
DEF ZEVENT	o		*
DELETE	x	Non-compilable	S7.8.2-16
DIM	Δ	With restrictions	S7.8.2-17
END	o		*
EOF	o		*
ERASE	x	Non-compilable	S7.8.2-18
ERL	o		*
ERR	o		*
ERROR	o		*
EXP	Δ	With restrictions	S7.8.2-19
FIELD	o		*
FILES	x	Non-compilable	S7.8.2-20
FIX	o		*
FOR to NEXT	Δ	With restrictions	S7.8.2-21
FORMAT	o		*
FRE	Δ	With restrictions	S7.8.2-22
GET	o		*
GETMEM	o		*
GOSUB RETURN	Δ	With restrictions	S7.8.2-23
GOTO	o		*
HEX$	Δ	With restrictions	S7.8.2-24
IF GOTO ELSE	o		*
IF THEN ELSE	o		*
INKEY$	o		*
INPUT	Δ	With restrictions	S7.8.2-25
INPUT$	o		*
INPUT#	o		*
INSTR	o		*
INT	o		*

Table 7.1 Compilability of Instructions and Functions (continued)

Instructions and Functions	Compilability	Remarks	Section
KEY	o		*
KEYLIST	x	Non-compilable	S7.8.2-26
KILL	o		*
LEFT$	o		*
LEN	o		*
LET	o		*
LFILES	x	Non-compilable	S7.8.2-27
LINE INPUT	Δ	With restrictions	S7.8.2-28
LINE INPUT#	o		*
LIST	x	Non-compilable	S7.8.2-29
LLIST	x	Non-compilable	S7.8.2-30
LOAD	x	Non-compilable	S7.8.2-31
LOC	o		*
LOCATE	o		*
LOF	o		*
LOG	Δ	With restrictions	S7.8.2-32
LPRINT	o		*
LPRINT USING	o		*
LSET	o		*
MERGE	x	Non-compilable	S7.8.2-33
MID$ (No.1)	o		*
MID$ (No.2)	o		*
MKD$	o		*1-
MKDMBF$	Δ	With restrictions	S7.8.2-34
MKI$	o		*
MKS$	o		*
MKSMBF$	Δ	With restrictions	S7.8.2-35
NAME	o		*
NEW	x	Non-compilable	S7.8.2-36
OCT$	Δ	With restrictions	S7.8.2-37
ON COM GOSUB	Δ	With restrictions	S7.8.2-38
ON ERROR GOTO	o		*
ON GOSUB	o		*
ON GOTO	o		*
OPEN	o		*
PCRD	o		*
PCWT	o		*

Table 7.1 Compilability of Instructions and Functions (continued)

Instructions and Functions	Compilability	Remarks	Section
PRINT	o		*
PRINT USING	Δ	With restrictions	S7.8.2-39
PRINT#	o		*
PRINT# USING	Δ	With restrictions	S7.8.2-40
PUT	o		*
PUTMEM	o		*
RDSET	Δ	With restrictions	S7.8.2-41
READ	Δ	With restrictions	S7.8.2-42
REM	o		*
RENUM	x	Non-compilable	S7.8.2-43
RESTORE	Δ	With restrictions	S7.8.2-44
RESUME	Δ	With restrictions	S7.8.2-45
RIGHT$	o		*
RND	o		*
ROT	Δ	With restrictions	S7.8.2-46
RSET	o		*
RUN (No.1)	x	Non-compilable	S7.8.2-47
RUN (No.2)	o		*
SAVE	x	Non-compilable	S7.8.2-48
SEARCH	o		*
SGN	o		*
SHA	Δ	With restrictions	S7.8.2-49
SHT	Δ	With restrictions	S7.8.2-50
SIN	Δ	With restrictions	S7.8.2-51
SPACE$	o		*
SPC	Δ	With restrictions	S7.8.2-52
SQR	Δ	With restrictions	S7.8.2-53
STOP	Δ	With restrictions	S7.8.2-54
STR$	o		*
STRING$	o		*
SYSTEM	x	Non-compilable	S7.8.2-55
SWAP	o		*
TAB	Δ	With restrictions	S7.8.2-56
TAN	Δ	With restrictions	S7.8.2-57
TIME$	o		*
TROFF	x	Non-compilable	S7.8.2-58
TRON	x	Non-compilable	S7.8.2-59

Table 7.1 Compilability of Instructions and Functions (continued)

Instructions and Functions	Compilability	Remarks	Section
VAL	Δ	With restrictions	S7.8.2-60
WHILE WEND	Δ	With restrictions	S7.8.2-61
WIDTH	o		*
WTSET	Δ	With restrictions	S7.8.2-62
ZBAS	o		*
ZCLOSE	o		*
ZCNTL	o		*
ZEVENT	o		*
ZIDV	o		*
ZLDV	o		*
ZMESSAGE	o		*
ZMESSAGE CLOSE	o		*
ZMESSAGE GET	o		*
ZMESSAGE KILL	o		*
ZMESSAGE OPEN	o		*
ZMESSAGE PUT	o		*
ZMOVE	o		*
ZODV	o		*
ZOPEN	o		*
ZRECEIVE	o		*
ZRELEASE	o		*
ZRESERVE	o		*
ZSEND	o		*
ZSIGNAL	o		*
ZSTART	Δ	With restrictions	S7.8.2-63
ZURGENCY	o		*
ZWAIT DELAY	o		*
ZWAIT EVENT	o		*

7.8.2 Different instruction and function specifications when using a compiler

When executing an operation using a compiler, the specifications are different from the specifications when an interpreter is used.

This section explains these differences. Chapter 11 of AD51H-BASIC (Command) Programming Manual gives details about the following instructions and functions. Instructions other than instructions explained in this section have specifications that are the same as for an interpreter. Therefore, for those instructions, see the interpreter explanations.

Table 7.2 Different Instruction and Function Specifications when Using a Compiler

No.	Instructions and Functions	Different Specifications, Restrictions, and Precautions	Alternative Plans
1	ATN	• When a double precision real number is included in the <numeric expression>, a double precision value is returned. In other cases, a single precision value is returned.	• Enclose the <numeric expression> by using the CSNG function, and always make the numerical value single precision.
2	AUTO	• The AUTO instruction is unusable. • AUTO can be used as a variable name.	———
3	BIN$	• If a value outside the range (–32768 to 65535) is designated to the expression, the operation result is the same as when 32767 is designated.	• Before using the BIN$ function, use the IF instruction to check the range, and generate an error using the ERROR instruction.
4	CHAIN	• The CHAIN instruction is unusable. • When a compiler is used, a "not support" error occurs.	• Substitute a RUN (No.2) instruction. • Make sure the RUN (No.2) instruction does not have the following CHAIN instruction function. • Execution at the place of interruption by designating a line number: Use GETMEM and PUTMEM instructions to deliver numerical values between programs, and use the ON GOTO instruction to execute a jump to the desired line number according to its numerical value. • Variable transfer using the ALL option: Use the GETMEM and PUTMEM instructions to execute the transfer.
5	CLEAR	• The CLEAR instruction is ignored. • When a compiler is used, it is ignored.	• When clearing a variable, use an assignment instruction.
6	COMMON	• The COMMON instruction is unusable. • When a compiler is used, a "not support" error occurs.	• Use the GETMEM and PUTMEM instructions to execute the transfer.

Table 7.2 Different Instruction and Function Specifications when Using a Compiler (continued)

No.	Instructions and Functions	Different Specifications, Restrictions, and Precautions	Alternative Plans
7	CONT	• The CONT instruction is unusable. • When a compiler is used, a "not support" error occurs.	———
8	COS	• When a double precision real number is included in the <numeric expression>, a double precision value is returned. In other cases, a single precision value is returned.	• Enclose the <numeric expression> by using the CSNG function, and always make the numerical value single precision.
9	CSNI	• The CSNI instruction does not check overflow.	• Before using the CSNI instruction, use the IF instruction to check the range, and generate an error using the ERROR instruction.
10	DATA	• Double quotation marks (") in the DATA instruction can be used as the symbols to enclose character-string constants. • If there is no double quotation mark (") symbol at both end of a character-string constant, the range from the head to the (") symbol, or the range from the (") symbol to the end of the character-string constant is considered data.	• Use the double quotation marks (") correctly.
11	DEFDBL	• Definition is executed before execution statement. • A variable already declared in DEFINT, DEFSNG, DEFDBL, and DEFSTR instruction cannot be declared again to a different type by using another instruction.	• Execute declarations before execution statements. • Do not execute re-definitions.
12	DEFFN	• When defining or calling a function, do not put a blank between FN and the name. • The type designation must always be added to the variable in the name, dummy argument, and function definition of the DEFFN instruction. • When calling other user-defined functions in the function definition expression, they must be defined before they are called. • Once defined, a user function cannot be redefined.	• Add the type designation. • When using other user-defined functions, define them before calling them.

Table 7.2 Different Instruction and Function Specifications when Using a Compiler (continued)

No.	Instructions and Functions	Different Specifications, Restrictions, and Precautions	Alternative Plans
13	DEFINT	• Definition is executed before execution statement. • A variable already declared in DEFINT, DEFSNG, DEFDBL, and DEFSTR instruction cannot be declared again to a different type by using another instruction.	• Execute declarations before execution statements. • Do not execute re-definitions.
14	DEFSNG	• Definition is executed before execution statement. • A variable already declared in DEFINT, DEFSNG, DEFDBL, and DEFSTR instruction cannot be declared again to a different type by using another instruction.	• Execute declarations before execution statements. • Do not execute re-definitions.
15	DEFSTR	• Definition is executed before execution statement. • A variable already declared in DEFINT, DEFSNG, DEFDBL, and DEFSTR instruction cannot be declared again to a different type by using another instruction.	• Execute declarations before execution statements. • Do not execute re-definitions.
16	DELETE	• The DELETE instruction is unusable. • DELETE can be used as a variable name.	———
17	DIM	• A variable cannot be used for the numeric expression that designates the size of an array by using the DIM instruction. • During execution, array subscript ranges are not checked. (However, execution of a check can be designated by using option [-d] for debugging during compiling.)	• Designate the maximum size.
18	ERASE	• The ERASE instruction is ignored. • When executing a compilation, a warning occurs.	• When using this to define a new array, define the maximum size array and use the array again. • When using this to erase an array, delete the ERASE instruction.
19	EXP	• When a double precision real number is included in the <numeric expression>, a double precision value is returned. In other cases, a single precision value is returned.	• Enclose the <numeric expression> by using the CSNG function, and always make the numerical value single precision.
20	FILES	• The FILES instruction is ignored. • When executing a compilation, a warning occurs.	———

Table 7.2 Different Instruction and Function Specifications when Using a Compiler (continued)

No.	Instructions and Functions	Different Specifications, Restrictions, and Precautions	Alternative Plans
21	FOR ~ NEXT	• FOR and NEXT instructions must always have a one-to-one correspondence.	• Give the FOR and NEXT instructions one-to-one correspondence.
22	FRE	• The FRE function always returns to 0. • When executing a compilation, a warning occurs.	———
23	GOSUB ~ RETURN	• "RETURN without GOSUB" is not checked.	• Use a counter to count and check the GOSUB and RETURN instructions.
24	HEX$	• If a value outside the range (–32768 to 65535) is designated to the <numeric expression>, the operation result is the same as when 32767 is designated.	• Before using the HEX$ function, use the IF instruction to check the range, and generate an error using the ERROR instruction.
25	INPUT	• When executing an input operation using the INPUT instruction, this does not have a screen edit function. • When a numerical value is input, overflow is not checked. Therefore, when too large a value is input, an error will not occur, and it becomes a negative number. • When the number of items divided by ',' is not the same as the number of variables, 'Redo from start' is displayed, and the INPUT instruction is executed again.	• Separate it into another task and execute the input operation using an interpreter.
26	KEYLIST	• The KEYLIST instruction is ignored. • When executing a compilation, a warning occurs.	———
27	LFILES	• The LFILES instruction is ignored. • When executing a compilation, a warning occurs.	———
28	LINE INPUT	• When executing an input operation using the INPUT instruction, this does not have a screen edit function.	• Process only part of the LINE INPUT instruction by using another interpreter task.
29	LIST	• The LIST instruction is unusable. • LIST can be used as a variable name.	———
30	LLIST	• The LLIST instruction is unusable. • LLIST can be used as a variable name.	———
31	LOAD	• The LOAD instruction is unusable. • LOAD can be used as a variable name.	———

Table 7.2 Different Instruction and Function Specifications when Using a Compiler (continued)

No.	Instructions and Functions	Different Specifications, Restrictions, and Precautions	Alternative Plans
32	LOG	• When a double precision real number is included in the <numeric expression>, a double precision value is returned. In other cases, a single precision value is returned.	• Enclose the <numeric expression> by using the CSNG function, and always make the numerical value single precision.
33	MERGE	• The MERGE instruction is unusable. • When a compiler is used, a "not support" error occurs.	———
34	MKDMBF$	• Only double precision internal representation IEEE format data is converted. When data of other formats is input, it is considered as IEEE format data and is converted.	———
35	MKSMBF$	• Only double precision internal representation IEEE format data is converted. When data of other formats is input, it is considered as IEEE format data and is converted.	———
36	NEW	• The NEW instruction is unusable. • NEW can be used as a variable name.	———
37	OCT$	• If a value outside the range (–32768 to 65535) is designated to the <numeric expression>, the operation result is the same as when 32767 is designated.	• Before using the OCT$ function, use the IF instruction to check the range, and generate an error using the ERROR instruction.
38	ON COM GOSUB	• An interrupt is executed at the head of each instruction by the interpreter, but an interrupt is executed at the head of the line by the compiler.	• Do not describe a multiple statement on the first line of interruption processing.
39	PRINT USING	• One PRINT USING instruction can describe eight units of display data.	• Split display data into several PRINT USING instructions.
40	PRINT# USING	• One PRINT# USING instruction can describe eight units of display data.	• Split display data into several PRINT# USING instructions.
41	RDSET	• Array subscripts and bit ranges are not checked.	• Before using the RDSET function, use the IF instruction to check the range, and generate an error using the ERROR instruction.

Table 7.2 Different Instruction and Function Specifications when Using a Compiler (continued)

No.	Instructions and Functions	Different Specifications, Restrictions, and Precautions	Alternative Plans
42	READ	• Even if the value exceeds the real number maximum value when reading a decimal constant, an overflow does not occur, and the real number maximum value is returned. • Even if the value exceeds the integer maximum value when reading octal and hexadecimal constants, an overflow does not occur, and the integer maximum value is returned. • When a values defined in the variable types of the READ and DATA instructions do not agree, a 'Syntax error' can occur on the READ instruction side. • However, this error does not always occur. Example: 10 DATA &H000012 20 READ A! 'An error does not occur. 30 DATA &H12X 40 READ A! 'An error occurs. • When an error occurs, data next to the data that caused the error is read.	• Try not to execute an abnormal read.
43	RENUM	• The RENUM instruction is unusable. • RENUM can be used as a variable name.	———
44	RESTORE	• Line number 0 is unusable.	———
45	RESUME	• Execution is restarted in a line unit. • When a multiple statement is used, pay attention to the following: RESUME: Execution is restarted from the first number of the line where it occurred. RESUME NEXT: Execution is restarted from the head of the following line. RESUME line number: Execution is restarted from the head of the designated line.	• Split multiple statements into several lines.
46	ROT	• The ROT function does not check an overflow.	• Before using the ROT function, use the IF instruction to check the range, and generate an error using the ERROR instruction.
47	RUN (No.1)	• The RUN (No.1) instruction is unusable.	———
48	SAVE	• The SAVE instruction is unusable. • SAVE can be used as a variable name.	———

Table 7.2 Different Instruction and Function Specifications when Using a Compiler (continued)

No.	Instructions and Functions	Different Specifications, Restrictions, and Precautions	Alternative Plans
49	SHA	• Argument overflows are not checked.	• Before using the SHA function, use the IF instruction to check the range, and generate an error using the ERROR instruction.
50	SHT	• Argument overflows are not checked.	• Before using the SHT function, use the IF instruction to check the range, and generate an error using the ERROR instruction.
51	SIN	• When a double precision real number is included in the <numeric expression>, a double precision value is returned. In other cases, a single precision value is returned.	• Enclose the <numeric expression> by using the CSNG function, and always make the numerical value single precision.
52	SPC	• When the SPC function is put at the end of the PRINT instruction, a line feed is executed.	• Add ';' (semicolon) after the SPC function.
53	SQR	• When a double precision real number is included in the <numeric expression>, a double precision value is returned. In other cases, a single precision value is returned.	• Enclose the <numeric expression> by using the CSNG function, and always make the numerical value single precision.
54	STOP	• A STOP instruction terminates a program. (Operates the same as an END instruction) • When executing a compilation, a warning occurs.	———
55	SYSTEM	• The SYSTEM instruction is unusable. • SYSTEM can be used as a variable name.	———
56	TAB	• When the TAB function is put at the end of the PRINT instruction, a line feed is executed.	• Add ';' (semicolon) after the TAB function.
57	TAN	• When a double precision real number is included in the <numeric expression>, a double precision value is returned. In other cases, a single precision value is returned.	• Enclose the <numeric expression> by using the CSNG function, and always make the numerical value single precision.
58	TROFF	• The TROFF instruction cannot be used. Even when a compilation is executed, errors are not displayed. Therefore, reliable operations when a compilation is executed cannot be guaranteed.	———
59	TRON	• The TRON instruction cannot be used. Even when a compilation is executed, errors are not displayed. Therefore, reliable operations when a compilation is executed cannot be guaranteed.	———

Table 7.2 Different Instruction and Function Specifications when Using a Compiler (continued)

No.	Instructions and Functions	Different Specifications, Restrictions, and Precautions	Alternative Plans
60	VAL	• Double precision values are always returned.	• After using the VAL function for conversion, substitute and use it for the variable of a necessary type.
61	WHILE ~ WEND	• WHILE and WEND instructions must always have a one-to-one correspondence.	• Give the WHILE and WEND instructions one-to-one correspondence.
62	WTSET	• Array subscripts and bit ranges are not checked.	• Before using the WTSET function, use the IF instruction to check the range, and generate an error using the ERROR instruction.
63	ZSTART	• If the multitask setting of the task designated to the <number> is IP, the interpreter program is started. If the multitask setting is CP, a compiler program is started. • The file designated to the <file name> must be a file (~.EXE) that is created by DBC (the BASIC compiler), if the task designated to the <number> of the ZSTART instruction is compiler BASIC. If a file other than a file created by DBC is designated, an error occurs or the communication module system malfunctions. • A task that has been executed one time can be restarted by designating a file name using the ZSTART instruction.	• When restarting a task, designate the loading file name (~.EXE) of the compiler program.

Appendix 1 Error Messages when Using the Line Number Tool

Error Message	Action to Take
Abnormal condition after line number change.	Due to the -S/-e option was designated, the changed line numbers are no longer in ascending order. Review the designated option.
Cannot close file. (File name)	Check the relevant drive for the following: no free capacity, write protection ON, no disk inserted, faulty drive status, etc.
Cannot create output file (output file name).	Review the file name. Check the relevant drive for the following: no free capacity, write protection ON, no disk inserted, faulty drive status, etc.
Cannot open file. (File name)	Review the file name. Check the relevant drive for the following: no free capacity, write protection ON, no disk inserted, faulty drive status, etc.
Cannot specify output file name with extension ".old".	Since ".old" cannot be designated, change it to another extension file name.
Could not make backup (file name) of source file (file name).	Review the file name. Check the relevant drive for the following: no free capacity, write protection ON, no disk inserted, faulty drive status, etc.
Drive not ready.	Check the relevant drive for the following: no free capacity, write protection ON, no disk inserted, faulty drive status, etc.
Error in option designation.	Review the designated option.
Failed to delete temporary file. (File name)	Check the relevant drive for the following: no free capacity, write protection ON, no disk inserted, faulty drive status, etc.
File name/path name is too long.	Review the file name.
File not found. (File name)	Review the file name.
Insufficient memory.	Increase the free area in the memory and try again.
Number of charactgers in one line has exceeded 254. (Number of lines)	Reduce the number of characters in the relevant line to within 254.
Number of characters in one line has exceeded 254. (Number of lines)	educe the number of characters in the relevant line to within 254, taking the increase in the number of characters when the line numbers change into consideration.
Processing terminated.	This message is always preceded by another error message: follow the corrective action for this other message.
Read error. (File name)	Check the relevant drive for the following: no free capacity, write protection ON, no disk inserted, faulty drive status, etc.
Warning: Reference line number cannot be changed. (Number of lines)	Occurs when, for example, the line number referred to in the relevant line does not exist in the source file. (Line number change processing continues.) Review the line number used to reference the relevant line.
Write error. (File name)	Check the relevant drive for the following: no free capacity, write protection ON, no disk inserted, faulty drive status, etc.

APPENDIX 2 ERROR MESSAGES WHEN COMPILING

This section explains the error messages that can occur during compiling. Error messages are classified into the following three types:

(1) Fatal error

This error immediately stops compilation.
A fatal error includes errors in connection with files, memory shortage errors, and errors inside the compiler.

[Sample error display]

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

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

(2) Error

This error indicates syntax errors in BASIC programs, violations of restrictions, and locations that cannot be correctly compiled.

[Sample error display]

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

After detecting as many errors as possible, the compiler stops compiling. Reexecute compilation after removing the error causes and confirming the operation by using an interpreter. When a compiler stops compiling, the "compiler aborted" message is displayed.

(3) Warning

This error message indicates a problem position, which is not actually an error. A typical warning is the detection of a statement to be ignored by the compiler.

[Sample error display]

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

When a warning is displayed, the compiler does not stop compiling. All processing is done until an EXE file is generated (~.EXE). When a warning is displayed, check the cause. Then, ignore the warning or modify the program as appropriate, and reexecute compilation. Warning displays can be cleared by using the [-w-] compiler option.

[Attention]

When an error is detected, the compiler skips data from the error occurrence position to the end of the statement. Therefore, a normal position immediately after an error is sometimes judged to be the error occurrence position. Or the position immediately after an error occurrence position sometimes cannot be detected.Therefore, the second error position and any subsequent ones may not always be correct.
The program line number and the ^ symbol which are displayed simultaneously with an error message indicate the location in the program the compiler read when the error was detected. The displayed location generally indicates the location near the error occurrence position. However, when an error is detected considerably after the compiler read the program, the displayed location indicates the location in the program considerably after the position of error occurrence. In order to solve this problem, if a program contains program lines which contain long complicated formulas and/or multiple statements which make an error difficult to detect, divide such formulas and lines into short statements.

(1) Fatal error

Error Message	Meanings	Corrective Actions
can't create output file	An intermediate file $x.ASM (x is a numerical character) could not be created in the current directory.	The condition (directory full) in which a directory cannot be created is thought to be the cause. Delete any unnecessary files, and reexecute compilation.
can't create work file #1	An intermediate file BASIC$$$.INC could not be created in the current directory.	The condition (directory full) in which a directory cannot be created is thought to be the cause. Delete any unnecessary files, and reexecute compilation.
can't create work file #2	An intermediate file BC.TMP could not be created in the current directory.	The condition (directory full) in which a directory cannot be created is thought to be the cause. Delete any unnecessary files, and reexecute compilation.
compiler stack overflow	The compiler stack area overflowed.	Simplify any complex expressions. Reduce the number of nesting times: i.e., FOR ~ NEXT etc. (If the number of nesting times is 20 to 35, an error occurs.)
date area overflow (65000 bytes)	The data area necessary for variables and constants was too large to be secured	Since, in most cases, an error occurs because a huge array is declared, reduce the size of an array.
evaluation stack overflow	An expression in one statement was too complicated.	Reduce the complexity of the expression. For example, substitute the intermediate value of an expression for a variable.
file I/O error	An error occurred when accessing a source file or an intermediate file.	The disk is faulty or is not empty enough (disk is full). when the disk is full, delete any unnecessary files and reexecute compilation.
line too long	A source program line was too long. (If the line consists of 299 characters or more, an error occurs.)	Either it is not a source file stored by interpreter BASIC of the AD51H-S3 or a source file has been damaged. Store it again correctly.
source file 'XXXX' not found	The source file designated in the command line was not found.	Designate a correct source file name.
symbol table overflow	The number of variables, labels, or FN functions was excessive. (If there are 500 variables that consist of 9-character names, an error occurs.)	Change any variables, labels, or FN functions with long names into short ones. In addition, delete any unnecessary variables, labels, or FN functions.
too many target line numbers	There are a great many target line numbers referred to in statements such as GOTO and GOSUB.	Replace such parts with labels or divide the program.
unexpected end of file in 'XXXX"	A source file was interrupted in the middle of a statement.	Compile it after completing the program.
'dbb.obj' not found 'dbc.lib' not found	A start up module file or library file were not found.	The two dbb.obj and dbc.lib files must exist in a compiler start path or current directory. Confirm that these two files exist. Confirm that these two files exist.

(continued)

Error Message	Meanings	Corrective Actions
'XXXX' failed: error level X	An execution error was reported by the assembler or linker.	Use the assembler or linker manual to investigate the cause of the error. [Attention] When executing a compiler by using Microsoft Macro Assembler Ver 4.0, if the [-4] option is not designated, this error occurs.
'XXXX' failed: Exec format error	The contents of the loading file of the assembler or linker were damaged.	Reregister the assembler or linker to the hard disk.
'XXXX' failed: No such file or directory	The assembler or linker were not found.	Place an assembler or linker in the current directory or the directory set in the environment variable PATH.
'XXXX' failed: Not enough memory	The assembler and a linker cannot be started because there is not enough memory.	Cancel the resident PRINT commands, etc. of the resident program. Remove any unnecessary device drivers. Or, make the designation of BUFFERS of CONFIG.SYS small. Taking these steps will increase the usable memory .

(2) Error messages

Error Message	Meanings	Corrective Actions
bad line number XXXXX	Incorrect syntax: There is a line number outside the range from 1 to 65529.	Designate the line number in the range from 1 to 65529.
DEF --- syntax error	Incorrect syntax: There is a syntax error in the DEFINT, DEFSNG and DEFDBL and DEFSTR commands.	Modify the program so that it has correct syntax.
DEF --- what ?	Incorrect syntax: There is a syntax error in the DEF command.	Modify the program so that it has correct syntax.
DIM --- syntax error	Incorrect syntax: There is a syntax error in the DIM command.	Modify the program so that it has correct syntax.
divide by 0	Improper parameter: There is a division by zero (/, \, MOD) in the numeric expression.	Modify the program so that it has no division by zero. [Attention] The compiler detects only the cases where a constant is divided by constant zero. When executing the program, division by zero is not detected as an error, and the maximum numerical value is returned. Single precision : +1.70141E+38 Double precision : +1.70141183460469D+38
expression too complex	Compiler restrictions: The numeric expression of a real number is too much complicated.	Reduce the complexity of the expression by taking some measures such as substituting an intermediate value of an expression for a variable.
FOR --- syntax error	Incorrect syntax: • The control variable and a substitution statement of an initial value are not found in the FOR~NEXT command. • A character-string variable is used as a control variable by mistake.	Designate a control variable or an initial value correctly.
FOR without NEXT	Incorrect syntax: • The NEXT command which corresponds to the FOR command is not found. • The FOR command and the NEXT command are not corresponding correctly.	Modify the program so that the FOR command and the NEXT command correspond correctly.
GOSUB not found GOSUB/GOTO not found GOTO not found	Incorrect syntax: The GOSUB command or the GOTO command is not found in the ON XX GOSUB command or the ON XX GOTO command.	Modify the program so that it has correct syntax.

(continued)

Error Message	Meanings	Corrective Actions
illegal constant	Incorrect syntax: A character that is not recognized as a number is used in the octal constant and hexadecimal constant.	Modify the program so that it has correct syntax.
illegal parameter	Improper parameter: • There is an improper parameter. • There is a thing other than a variable or an arrangement name in the place where a variable or an arrangement name is needed.	Modify a program by using a proper parameter.
index must be 0...32766	Compiler restrictions: The value that designates the size of an arrangement in the DIM command is too large. Or, a variable or an expression is used to designate the size.	Reduce the size. Change the value that designates the size into a constant.
INPUT --- , or ; not found	Incorrect syntax: A thing other than ", " or ";" is found after the INPUT " ay string " command.	Modify the program so that it has correct syntax.
LINE INPUT --- ';' not found	Incorrect syntax: A thing other than " , " is found after the LINE INPUT "ay string" command.	Modify the program so that it has correct syntax.
LINE INPUT --- must be string variable	Incorrect syntax: A thing other than a character-string variable is designated to the string variable in the LINE INPUT command.	Designate a character-string variable to the string variable.
line number not found	Incorrect syntax: There is not a line number in the source program.	Describe a line number.
line number or label not found	Incorrect syntax: • There is not a line number or a label after the GOTO command or the GOSUB command. • An improper label name is used in the THEN, ELSE, RETURN, RESUME and RESTORE commands.	• Designate a line number or a label. • Designate a correct label name.
line number XXXXX not sequential	Incorrect syntax: The line number of a source program is not designated in ascending order.	Designate the line number in ascending order.
missing operand	Incorrect syntax: There is not an argument or an expression in the place where there must be an argument or an expression,	Designate an argument or an expression.
NEXT without FOR	Incorrect syntax, compiler restrictions: The NEXT command without a corresponding FOR command is found.	Describe a corresponding FOR command.

(continued)

Error Message	Meanings	Corrective Actions
ON --- line number or label not found	Incorrect syntax: There is a syntax error in the row of line numbers or labels in the ON XX GOSUB command or the ON XX GOTO command.	Modify the program so that it has correct syntax.
ON --- string expression not allowed	Incorrect syntax: A character-string expression is designated in the ON <expression> GOSUB command and the ON <expression> GOTO command.	Designate a numeric expression to <expression>
OPEN --- file name not found	Incorrect syntax: There is not a character string of a file name in the OPEN command.	Designate a correct file name.
OPEN --- INPUT/OUTPUT/APPEND not found	Incorrect syntax: There is a thing other than the INPUT, OUTPUT and APPEND commands after the OPEN "XXX" FOR command.	Designate either INPUT, OUTPUT or APPEND command.
parameter must be numerical expression	Improper parameter: There is a thing other than a numeric expression in the place where a numeric expression is necessary.	Designate a numeric expression.
parameter must be string expression	Improper parameter: There is a thing other than a character string in the place where a character string is necessary.	Designate a character string.
parameter must be variable	Improper parameter: There is a thing other than a variable in the place where a variable is necessary.	Designate a variable.
port number must be constant	Improper parameter: The port number (%X) must be an integer constant.	Designate an integer constant.
PRINT USING --- ';' not found	Incorrect syntax: There is a character that is not ";" next to the format character string in the LPRINT and PRINT USING commands.	Change the character to ";".
PRINT USING --- format string not found	Incorrect syntax: There is a thing other than a format character string after the USING command in the LPRINT and the PRINT USING commands.	Designate a format character string.
PRINT USING --- illegal parameter	Incorrect syntax: There is a syntax error in the row of data to display in the LPRINT and the PRINT USING commands. (The number of data to display is restricted to 8 in the compiler.)	Modify the program so that it has correct syntax.
PRINT USING --- too many parameters	Compiler restrictions: The number of data to display is too large in the LPRINT and the PRINT USING commands. (The number of data to display is restricted to 8 in the compiler.)	Split one PRINT USING command to several PRINT USING commands.

(continued)

Error Message	Meanings	Corrective Actions
RESTORE --- line number or label not found	Incorrect syntax: There is a thing other than a line number or a label after the RESTORE command.	Designate a line number or a label.
RESUME --- syntax error	Incorrect syntax: There is a syntax error in the RESUME command.	Modify the program so that it has correct syntax.
statement expected	Incorrect syntax: There is other thing such as a constant or a symbol, etc. than a command or a function at the beginning of a sentence.	Modify the program so that a command or a function is put at the beginning of a sentence.
STEP --- string expression not allowed	Incorrect syntax: The increment value specified by the STEP command is a character-string expression in the FOR-NEXT command.	Modify the program so that it has correct syntax.
string expression not allowed	Improper parameter: A character-string variable or a character-string expression is used.	This error occurs when a character-string variable or a character-string expression is designated by mistake to the place where a numerical value variable or a numerical expression must be designated. Modify the program so that it has correct syntax.
string expression too complex	Compiler restrictions: A character-string expression is too much complicated.	Reduce the complexity of the expression by taking some measures such as substituting an intermediate value of an expression for a variable,
subscript out of range	Improper parameter: • The value of the subscript of an array is outside the range specified in the DIM command. • Dimensionality specified in the DIM command and the one actually written in the program are not the same.	• Modify the program so that the value of the subscript of an array is within the range specified in the DIM command. • Modify the program so that the dimensionality specified in the DIM command and the one actually written in the program are the same.
swap type mismatch	Improper parameter: Types of two variables to be exchanged are not the same type.	Modify the program so that types of two variables to be exchanged are the same.
syntax error	Incorrect syntax: There is a syntax error in a command or a function.	Modify the program so that it has correct syntax.
syntax error --- binary operator	Incorrect syntax: There is only one term for a binary operator.	Modify the expression so that it is a correct expression.
syntax error at end of statement	Incorrect syntax: There is a reserved word, a symbol, or an expression, etc. in the place that should be the statement end.	Modify the program so that it has correct syntax.
syntax error in expression	Incorrect syntax: There is a syntax error in the expression.	Modify the program so that it has correct syntax.

(continued)

Error Message	Meanings	Corrective Actions
syntax error in function parameter list	Incorrect syntax: There is a syntax error in the formal argument of the FN function in the DEF FN function.	Modify the program so that it has correct syntax.
syntax error in parameter	Incorrect syntax: There is a syntax error in the row of the arguments of a statement (a command or a function).	Modify the program so that it has correct syntax.
syntax error in string expression	Incorrect syntax: • There is a syntax error in the character string expression. • Operators that cannot be used for character strings are used.	Modify the program by using operators that can be used for character strings.
THEN/GOTO not found	Incorrect syntax: The THEN and GOTO commands are not found after the conditional expression in the IF command.	Modify the program by using the THEN and GOTO commands.
TO --- string expression not allowed	Incorrect syntax: The end value that is indicated after the TO command becomes a character-string expression in the FOR command.	Designate a numeric expression or a numerical value variable as the end value.
TO not found	Incorrect syntax: The TO command is not found in the FOR command.	Modify the program by using the TO command.
type mismatch	Improper parameter: • The parameter type is not proper. • A character string is given to a place where a numerical value is needed as the argument of a command or a function. Or a numerical value is given to a place where a character string is needed as the argument of a command or a function.	Modify the program so that the parameter type is proper.
WEND without WHILE	Incorrect syntax, compiler restrictions: The WEND command without a corresponding WHILE command is detected.	Modify the program so that the WHILE and the WEND commands correspond correctly.
WHILE without WEND	Incorrect syntax: the WHILE command without a corresponding WEND command is detected.	Modify the program so that the WHILE and the WEND commands correspond correctly.
XXXXX --- file number not found	Incorrect syntax: There is not a file number in the statement XXXX. Or there is a thing other than a numerical value in the position where there must be a file number.	Designate a file number.
XXXXX --- name too long	Incorrect syntax: Name XXXXX of a variable or a FN function is too long. (A maximum of 15 characters)	Shorten the name to 15 characters or less.

(continued)

Error Message	Meanings	Corrective Actions
XXXXX --- ON/OFF/STOP not found	Incorrect syntax: The ON, OFF and STOP commands are not found in the statement XXXXX where they are required.	Designate ON, OFF, and STOP commands.
XXXX --- redimensioned array	Compiler restrictions: Array variable XXXXX has more than one definition.	Modify the program so that an array variable is not redefined.
XXXX --- string variable expected	Improper parameter: There is a thing other than a character-string variable in the place where there must be a character-string variable in the statement XXXXX.	Designate a character-string variable.
XXXXX --- undefined function	Incorrect syntax: An undefined FN function is called.	Define the function, or designate a defined function.
XXXXX --- undefined label	Incorrect syntax: Label XXXXX that does not exist is referred to in the GOTO and the GOSUB commands, etc..	Designate an existing label.
XXXXX --- undefined line number	Incorrect syntax: Line number XXXXX that does not exist is referred to in the GOTO and the GOSUB commands, etc..	Designate an existing line number.
XXXXX --- undefined variable	Compiler restrictions: An undefined variable is referred to.	———
XXXXX expected 'X' expected	Incorrect syntax: There are other things than XXXXX or "X" in the position where there must be XXXXX or 'X'.	Designate XXXXX or 'X'.
XXXXX not supported	Compiler restrictions: XXXXX contains a statement (a command or a function) that cannot be used by the compiler.	Modify the program so that it does not contain a command or a function that cannot be used by the compiler.

(3) WARNING

Error Message	Meanings	Corrective Actions
STOP assumed to be END	The STOP command is compiled being assumed as the END command.	———
XXXXX ignored	The command or the function XXXXX is ignored.	———

IMPORTANT

Design the configuration of a system to provide an external protective or safety inter locking circuit for the PCs.

Under no circumstances will Mitsubishi Electric be liable or responsible for any consequential damage that may arise as a result of the installation or use of this equipment.

All examples and diagrams shown in this manual are intended only as an aid to understanding the text, not to guarantee operation. Mitsubishi Electric will accept no responsibility for actual use of the product based on these illustrative examples.

Owing to the very great variety in possible applications of this equipment, you must satisfy yourself as to its suitability for your specific application.

type AD51H-BASIC (Program edit, Compile)

Programming Manual

MODEL	AD51H-P(PR,CMP)-E
MODEL CODE	13JF44
IB(NA)66568-A(9506)MEE	

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100-0005 TELEX : J24532 CABLE MELCO TOKYO
NAGOYA WORKS : 1-14 , YADA-MINAMI 5 , HIGASHI-KU, NAGOYA , JAPAN