USER'S MANUAL

NEC

IE-75000-R/IE-75001-R

IN-CIRCUIT EMULATOR

Document No. EEU-1455
(O. D. No. EEU-846B)
Date Published January 1997 N
Printed in Japan

USER'S MANUAL

NEC

IE-75000-R/IE-75001-R

IN-CIRCUIT EMULATOR

MS-DOS™ is a trademark of Microsoft Corp. USA.

PC DOS™ is a trademark of IBM Corp. USA.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

J96. 8

Major Revisions in This Version

Section	Description
Whole manual	Devices to be debuged, the uPD75036, 116H, 117H, 312B, 316B, P336, are developed.
5-1	CHAPTER 5 OPERATION OVERVIEW Change of "Note" in "5.2.1 (5) When setting set-up file"
7-1	CHAPTER 7 VERSION OF FIRM WARE ROM Firmware ROM (Ver.1.4) and control program (Ver.1.1) are developed.
8-46 8-47 8-272	CHAPTER 8 COMMAND EXPLANATION 8.4.6 (2) Example o Addition of "Note" in "(c) When operand and those that follow are omitted" o Addition of "(d) When deleting an event condition that was previously specified o Change of "8.4.45 Example (a) Specifying condition to activate and deactivate the tracer
B-6	APPENDIX B LIST OF ERROR MESSAGES Change of the error message, No.49 Addition of the error messages, NO.50, 51, 52
C-1 to C-4	APPENDIX C NOTES ON CORRECT USE Deletion of No.4, 13, 15 in the preceding edition.

PREFACE

Users: This manual is for engineers who intend to use a 4-bit single-chip microcomputer (75X Series device) for system debugging with the IE-75001-R.

Purpose: The purpose of this manual is to help users understand the debugging capabilities of the IE-75001-R.

Organization:

This manual contains the following major chapters:

- System configuration
- Function overview
- Command explanation

Guidance: Before using this manual, the user should be familiar with the functions and how to use a 75X Series device to be debugged and have a knowledge of a debugger.

When using the IE-75000-R, please refer to the IE-75001-R as the IE-75000-R throughout this manual (Refer to CHAPTER 1 GENERAL).

To understand the general functions and operation methods of the IE-75001-R:

Read the entire manual in the order of the table of contents.

To understand the basic specification and operating environments:

Read Chapters 1 through 4.

To understand the basic operating procedures and functions:

Read Chapters 5 and 6.

To understand the types, functions, and general formats of commands:

Read Chapter 8.

To connect a peripheral device to the IE-75001-R using a serial or parallel interface:

Read Chapter 9.

Terminology:

The following table shows the meaning of words frequently used in this manual.

Terminology	Meaning
Emulation device	Device that emulates a target device in the emulator, including an emulation CPU
Emulation CPU	CPU that executes the program coded by the user in the emulator
Target device	Device to be emulated such as a uPD78350 chip
Target program	Program to be debugged, that is, user-coded program
Target system	System to be debugged, that is, user-produced system, including a target program and user-produced hardware. In a narrow sense, refers only to hardware.

Legend:	Data weight:	Higher digits on the left side Lower digits on the right side
	Within the box:	Contents of the displayed monitor screen or the entered command
	Key descriptions:	xxxxx: Key entry <cr> : Return key [ESC] : Escape key
	Note:	Explanation of the indicated part of the text
	Caution:	Information requesting the user's special attention
	Remark:	Supplementary information
	Numeric value:	Binary: xxxxY Octal: xxxxQ Decimal: xxxxT Hexadecimal: xxxxH

List of relevant documents

Document name		Document No.
IE-75000-R-EM USER'S MANUAL		EEU-673
IE-75617-R-EM USER'S MANUAL (PRELIMINARY)		EEU-840
PG-1500 USER'S MANUAL		EEU-651
RA 75X ASSEMBLER PACKAGE USER'S MANUAL	Operation PC-9800 series (MS-DOS) base IBM PC series (PC-DOS) base	EEU-731
	Language	EEU-730
PG-1500 CONTROLLER USER'S MANUAL		EEU-754

Note The contents of the relevant documents may be changed without notice. Be sure to conform you use the latest manual before designing.

SUMMARY OF CONTENTS

CONTENTS

LIST OF PHOTOGRAPHS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 GENERAL

The IE-75001-R is an in-circuit emulator for efficiently debugging hardware or software which uses a 75X Series device, which is a 4-bit single-chip microcomputer. It cannot operate without an optional emulation board, the IE-75000-R-EM or IE-75617-R-EM(Note). The IE-75000-R-EM is provided in the IE-75000-R.

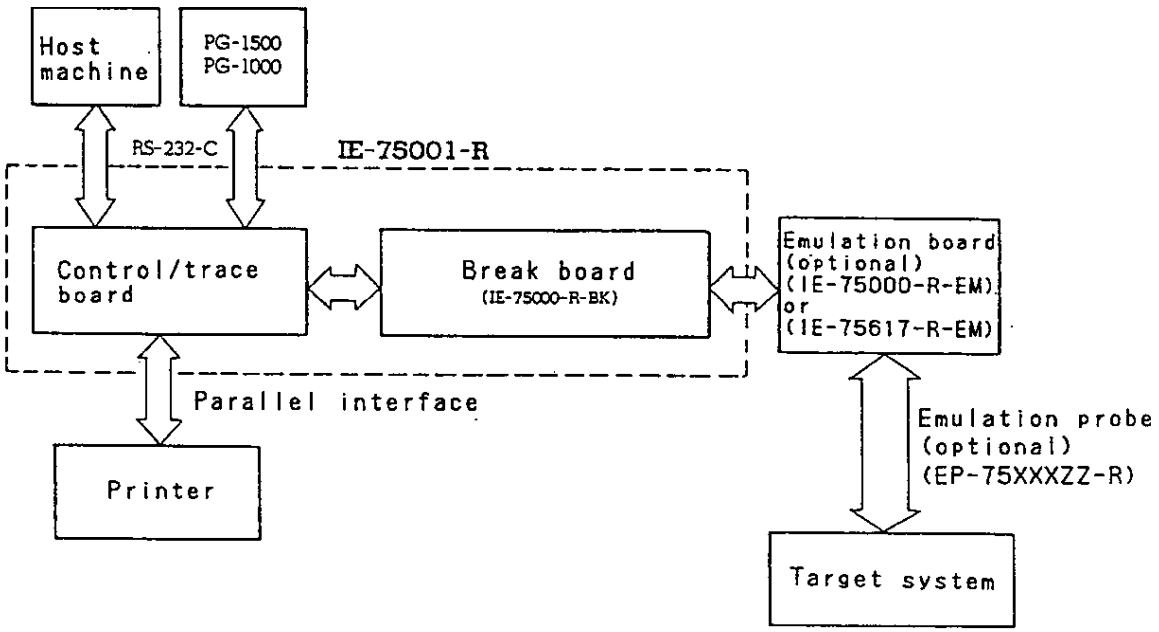
This chapter explains the system configuration and the basic specification of the IE-75001-R.

Note: Under development

1.1 System Configuration

Figure 1-1 shows the system configuration of the IE-75001-R.

Fig. 1-1 IE-75001-R System Configuration



1.2 Basic Specification

Table 1-1 Basic Specification

Item		Content
Device to be emulated		uPD75xxx
Operating frequency		32 kHz to 6 MHz (CPU clock)
Memory capacity		Program memory: 64K bytes Data memory: 4K nibbles
Emulation function		Realtime execution by non-break (RUN N) Realtime execution with breakpoint (RUN B) CPU trace execution by a single step (RUN T)
Event detection conditions	Access-type detection (BRA1-BRA4)	Detects the access to the specified full event. (address, data, status, external signal) x 4 points (address maskable). Two points can be address-partitioned.
	Fetch-type detection (BRS1, BRS2)	Can set 7-point parallel fetch, 4-point sequential fetch (BRS1), 1-point fetch (BRS2), and AND with an external signal.
Path count		Counts the number of times a break event condition is detected (1 to 255). Simultaneous detection of two or more event conditions is counted as one.
Event delay function		Sets an event detecting point for a trace frame. FIRST/MIDDLE/LAST
Forcible break		Manual break or guard break (data memory, SPR, register, stack)

Table 1-1 Basic Specification (Cont'd)

<table>
<tr><th colspan="2">Item</th><th colspan="2">Content</th></tr>
<tr><td rowspan="5">Trace function</td><td rowspan="4">Trace mode</td><td>Trace condition</td><td>Clock cycle</td></tr>
<tr><td>Total trace:
Performs traces unconditionally.</td><td>Machine cycle:
Performs trace per system clock.
Valid only when total trace is selected (port trace is also possible).</td></tr>
<tr><td>Section trace:
Starts trace when the enable condition is established and terminates it when the disable condition is established.</td><td rowspan="2">Event cycle:
Trace only a necessary bus when a fetch, read, write, or interrupt occurs.
Valid when total mode is selected (port trace is impossible).</td></tr>
<tr><td>Qualify trace:
Performs trace only when the qualify condition is established.</td></tr>
<tr><td>Trace capacity</td><td colspan="2">49 bits x 2K words</td></tr>
<tr><td></td><td>Data display</td><td colspan="2">Instruction display, retrieval display, checkpoint display, and binary display (port, external data)</td></tr>
<tr><td>Active mode</td><td>Functions realizable during emulated device operation</td><td colspan="2">Event detection condition reference/modification
Event delay reference/modification
Trace data display
Trace mode reference/modification
Trace restart</td></tr>
<tr><td colspan="2">Event output</td><td colspan="2">The trigger signal is output when a break condition is established (active low level).</td></tr>
<tr><td colspan="2">External interface</td><td colspan="2">RS-232-C (300-19200 bps): 2 channels
CH1: For console or host machine connection
CH2: For PROM programmer or VAX
Centronics interface: 2 channels
Centronics input: For object download
Centronics output: Output of centronics input by through</td></tr>
</table>

Table 1-1 Basic Specification (Cont'd)

Item	Content
Coverage function	Can set the CO coverage function and a coverage area.
Check function	Writes the contents of up to five SPR register memories into the tracer when an event is established. Execution is automatically restarted immediately after it has been terminated temporarily.
Access-type data output	Can output R/W data in two addresses (each nibble) of SPR data memory to the external sense clip in realtime mode.
Environment condition loading/saving	Can load and save debugging environment, and data memory.
Family expansion	Replacing the adapter board enables the IE-75001-R to cope with all the products in a family. Refer to the IE-75000-R-EM User's Manual and IE-75617-R-EM User's Manual (Preliminary) for correspondence between target devices and emulation probes.
Outer dimension	Depth: 370 mm, Width: 160 mm, height: 283 mm
Weight	Approx. 8.5 kg
Input voltage	AC100/120V (50/60 Hz) Input automatic AC200/240V (50/60 Hz) switching
Input current (TYP.)	AC100V 2A, AC200V 1A (50/60Hz)
Working temperature range	10° C to 40° C
Storage temperature range	-20° C to +45° C (no condensation)
Ambient humidity range	0% to 90% (RH)
Installation location	Install the IE-75001-R in a place free from refuse and dirt. Do not place any obstacles near the air inlet.

1.3 Configuration of the Control/Trace Board and the Break Board

This section explains the control/trace board block and the break board (IE-75000-R-BK) block, which are a nucleus of the IE-75001-R.

Refer to the IE-75000-R-EM User's Manual and IE-75617-R-EM User's Manual (Preliminary) for the emulation board block.

1.3.1 Control/trace board

Figure 1-2 shows the following components of the control/trace board.

(1) Driver control

This is an interface to a driver module.

(2) Trace RAM

The trace RAM holds newest 2047-step data up to the event detection point during each bus cycle.

(3) Memory bank selector

The memory bank selector selects the ROM, DRAM unit, or trace RAM by memory bank switching.

(4) Serial interface

The serial interface, which conforms to the RS-232-C standard, has two channels.

(5) Parallel interface

The parallel interface has two channels: One for high-speed download and the other for through output.

(6) I/O selector

The I/O selector selects the serial interface, parallel interface, or driver control by memory bank switching.

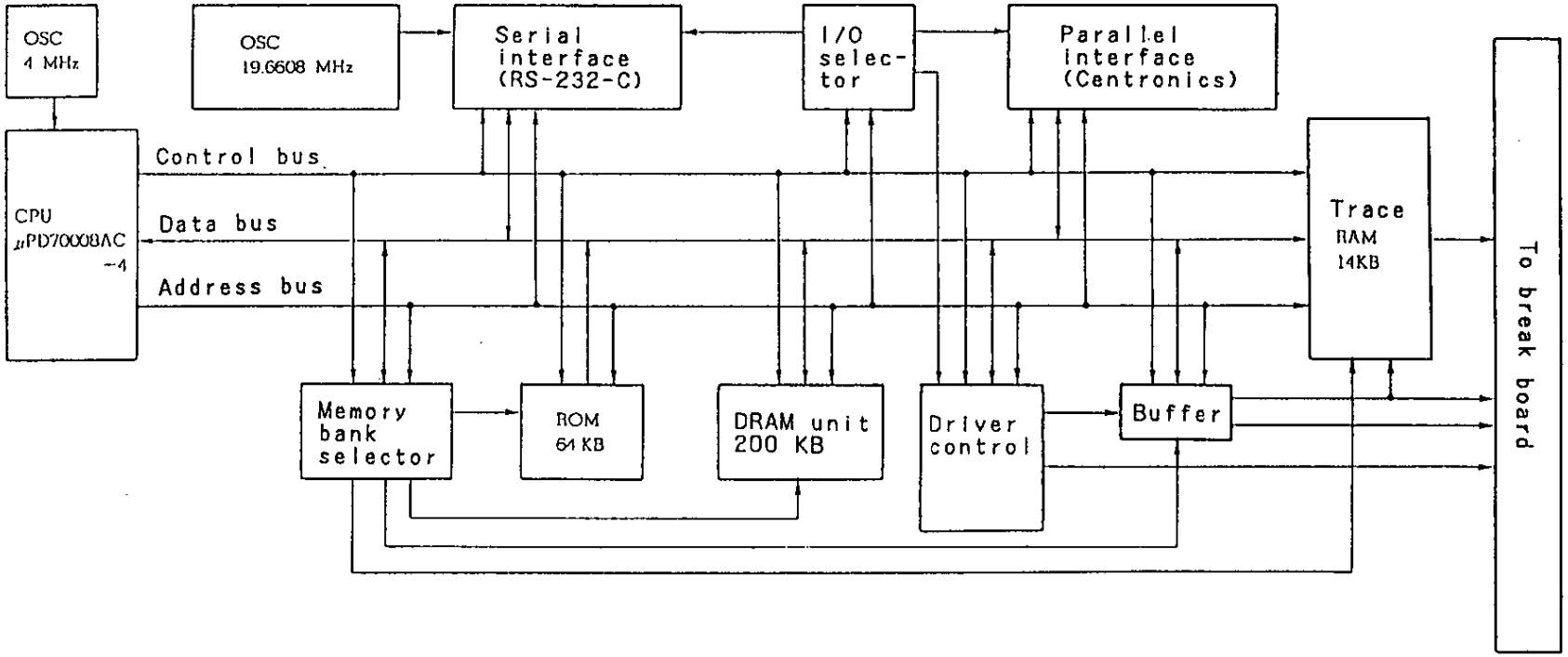
(7) DRAM unit

The DRAM unit has a 200K-byte memory, 192K-byte work area for symbols, and 8K-byte work area for programs.

(8) ROM

This is a 64K-byte ROM containing a program which activates the IE-75001-R.

Fig. 1-2 Block Diagram of Control/Trace Board



1.3.2 Break board

Figure 1-3 shows the following components of the break board.

(1) Break control

This portion controls the break conditions. The IE-75001-R has various break functions. Various break conditions can be set by a combination of some of event conditions.

(2) Trace control

This portions controls the trace conditions. The IE-75001-R has the trace function to store the execution state of the CPU. Various trace conditions can be set by a combination of some of event conditions.

(3) Alternate memory

The alternate memory is used for communication alternately by the supervisor CPU and the emulation CPU.

(4) Program memory and data memory

The IE-75001-R has a 64K-byte program memory and a 2K-byte data memory. When the target system is not developed, software can be debugged using these memories.

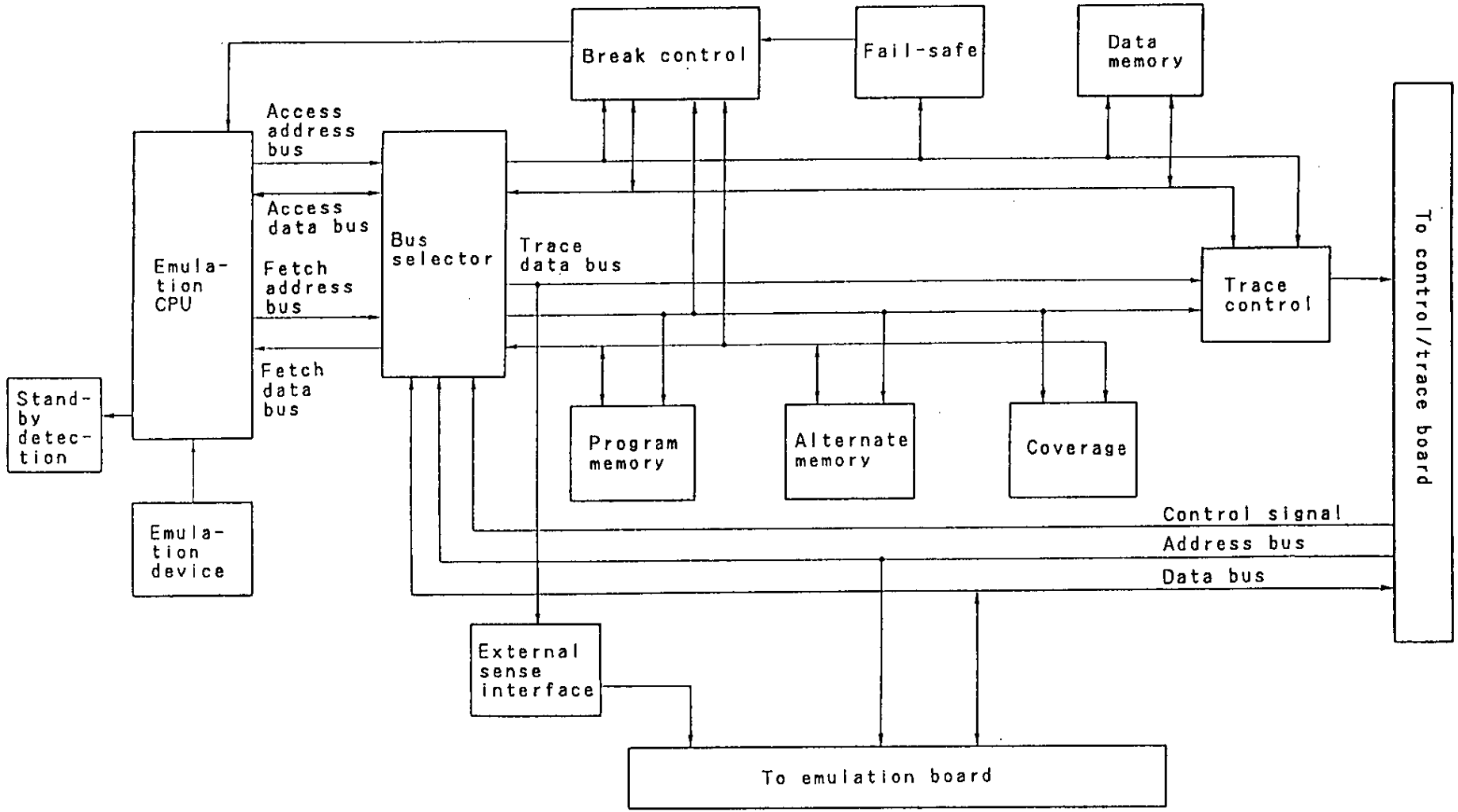
(5) Fail-safe

This is a circuit which protects the internal RAMs, SFRs, registers, and stacks that a device does not have from damage.

(6) Coverage

This portion measures the target programs that are executed during emulation to display the executed parts and the non-executed parts discriminately.

Fig. 1-3 Block Diagram of Break Board



CHAPTER 2 NOMENCLATURE

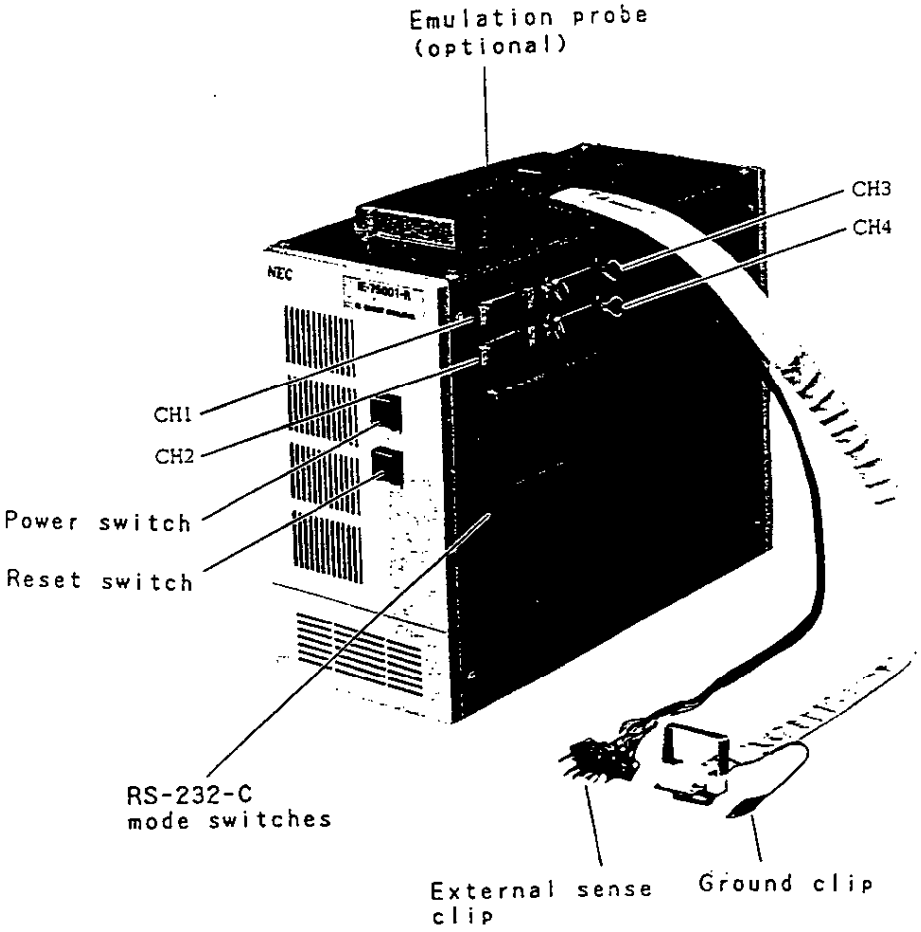
This chapter shows the IE-75001-R and its accessories.

The packing box contains the IE-75001-R main unit and accessory box. If any part is missing or damaged, please contact NEC's special agent.

2.1 Nomenclature of the IE-75001-R

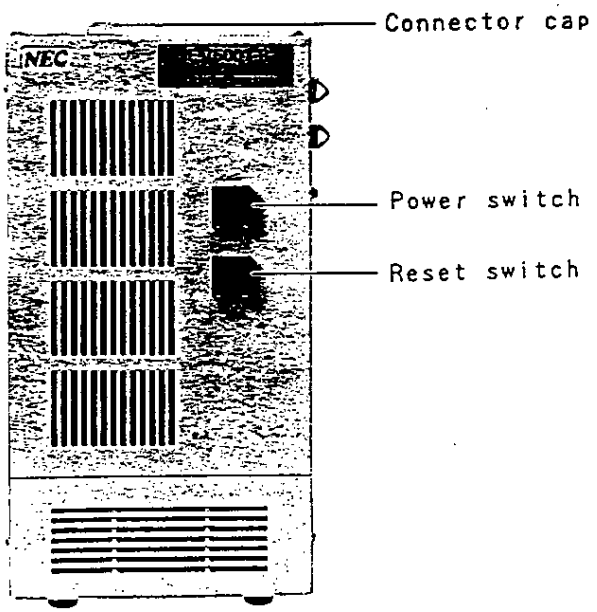
(1) External view

Photo 2-1 IE-75001-R



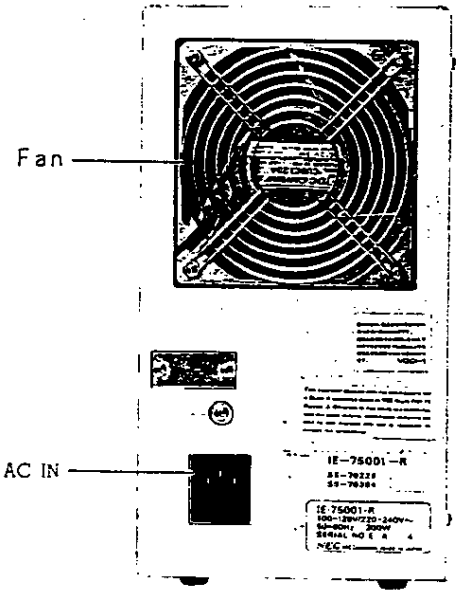
(2) Front view

Photo 2-2 Front View of IE-75001-R



(3) Rear view

Photo 2-3 Rear View of IE-75001-R



(4) Side views

Photo 2-4 Side View of IE-75001-R

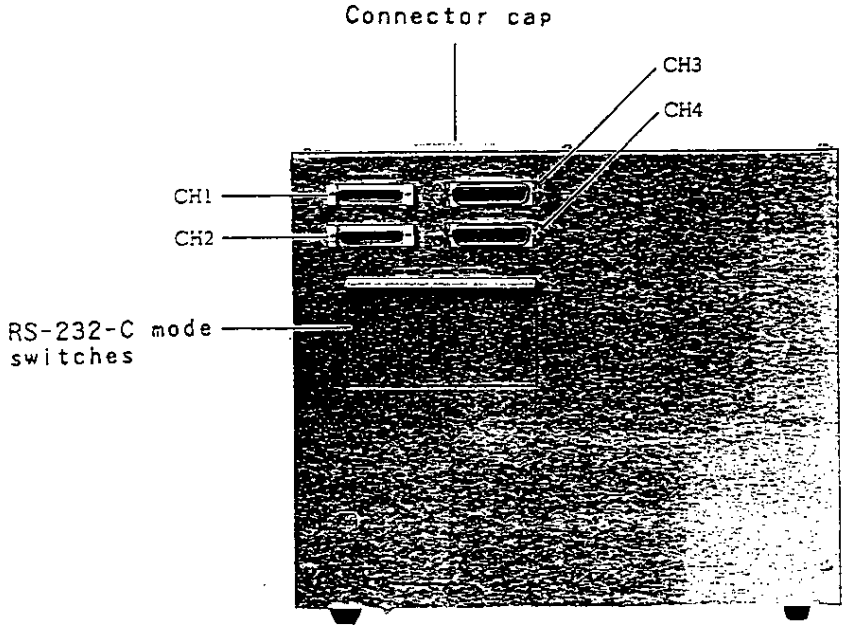
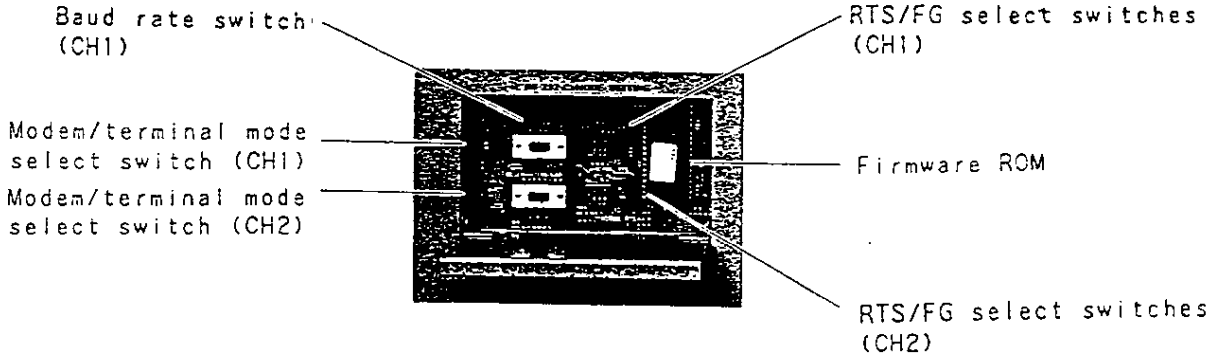


Photo 2-5 Enlarged View of the RS-232-C Mode Switches



(5) Boards

The IE-75001-R contains the following two boards.

(a) Break board (IE-75000-R-BK): 1
(b) Control/trace board (attached to IE-75001-R): 1

Remove the six screws on the top of the IE-75001-R to open the cover.

Photo 2-6 Location of the Boards

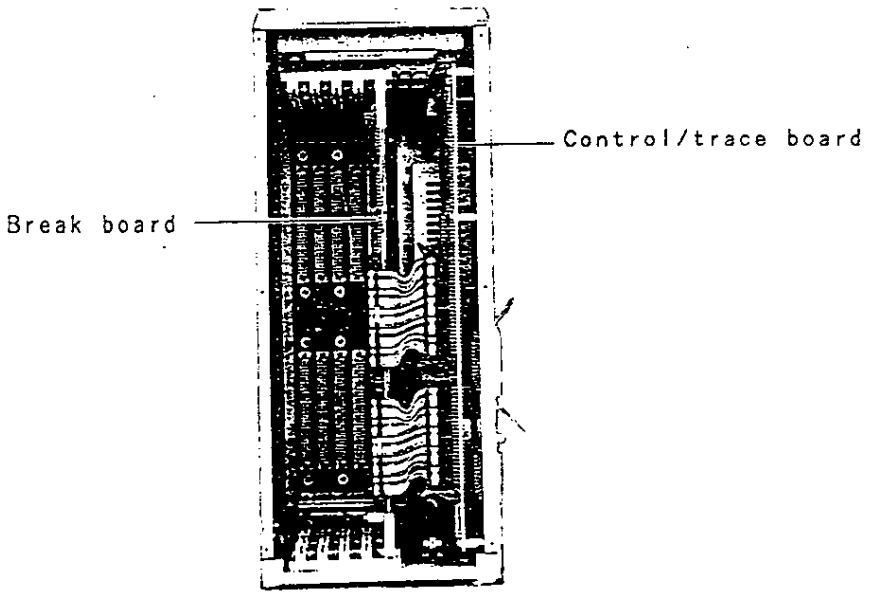


Photo 2-7 Break Board (IE-75000-R-BK)

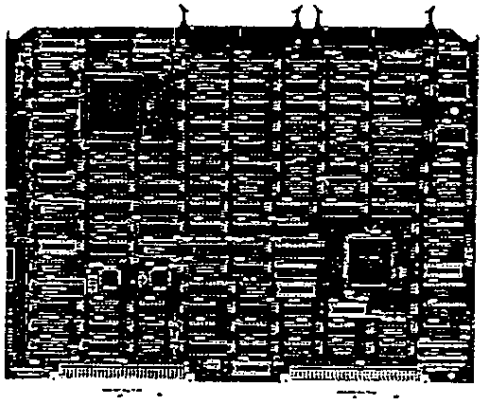
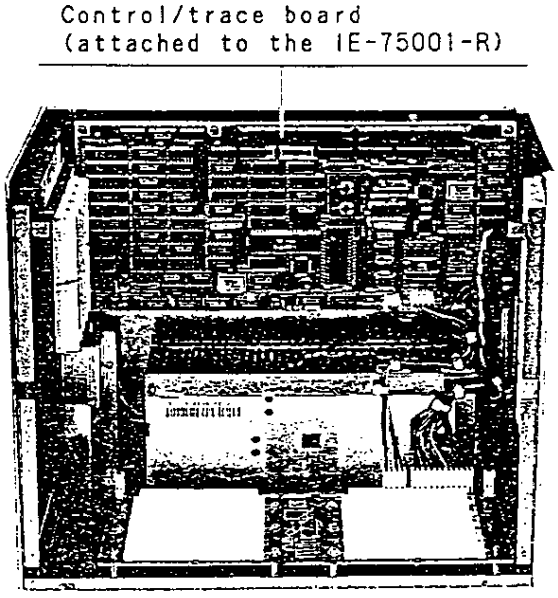


Photo 2-8 Control/Trace Board



2.2 Accessories

Confirm that the accessory box contains the following accessories.

(a)	IE-75000-R/IE-75001-R user's manual (this manual):	1
(b)	100-VAC power cord:	1
(c)	200-VAC power cord:	1
(d)	RS-232-C interface cable:	1
(e)	Ground lead cable:	1
(f)	Spare fuse:	1
(g)	AC adapter:	1
(h)	Accessory list:	1
(i)	Warranty:	1
(j)	List of packed contents:	1

CHAPTER 3 INSTALLATION

This chapter explains how to connect the emulation board to the IE-75001-R and set the RS-232-C mode switches.

3.1 Installation

Connect the emulation board, emulation probe (EP-75XXXZZ-R, optional), and accessory cables to the IE-75001-R. Use the emulation probe that matches the 75X series package used in the target system.

(1) Connecting the emulation board and emulation probe

① Remove the screws from the top of the IE-75001-R to open the cover, then pull the IE-75000-R-BK out from the slot.

② Screw down an optional emulation board, the IE-75000-R-EM or IE-75617-R-EM on the IE-75000-R-BK.

③ Connect the adapter board of the emulation probe to the two boards attached in step ②.

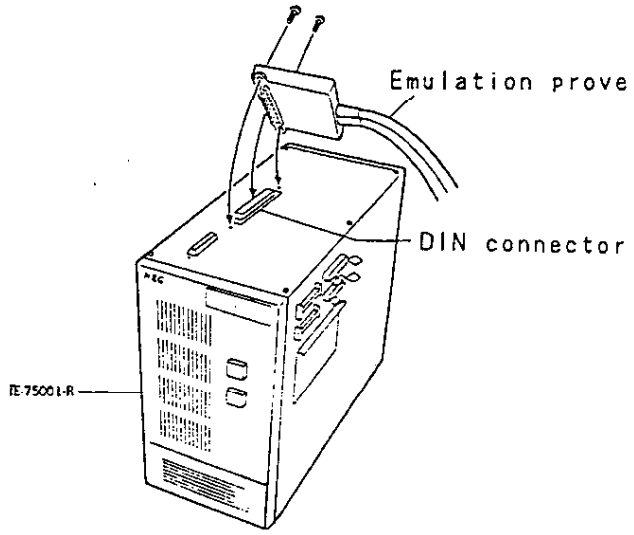
④ Install the three boards attached in step ③ in the IE-75001-R housing.

⑤ Close and screw down the housing cover. Remove the connector cap if necessary, depending on the emulation probe to be used.

⑥ Connect the emulation probe to the connector and screw it down securely.

Caution: If the emulation board and emulation probe are connected incorrectly, the IE-75001-R may be damaged. For details of how to connect the board and probe, refer to the descriptions of the emulation board (IE-75000-R-EM or IE-75617-R-EM) and other emulation probes in the user's manual.

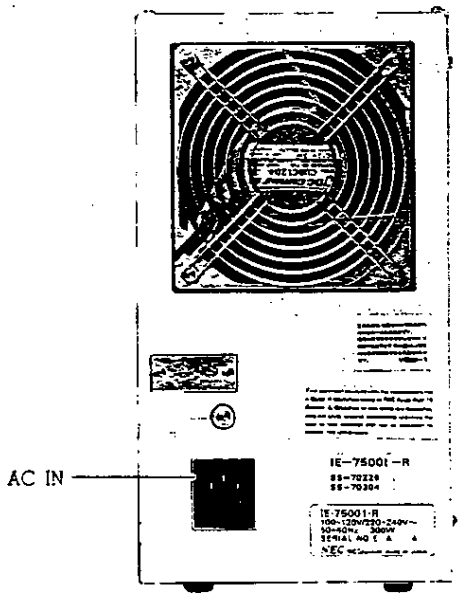
Fig. 3-1 Emulation Probe Connection



(2) Connecting the power cord

Connect the 100- or 200-VAC power cord to AC IN on the rear of the IE-75001-R.

Photo 3-1 Rear View of IE-75001-R (AC IN)



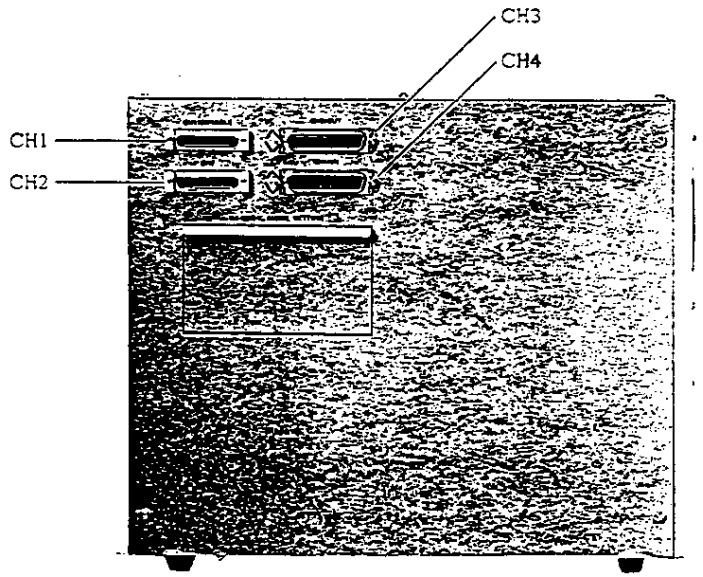
(3) Connecting the RS-232-C interface cable

When a host machine is connected, connect the RS-232-C interface cable to the CH1 connector on the side of the IE-75001-R. When PG-1500 or -1000 is connected, connect the interface cable to the CH2 connector on the side.

(4) Connecting the parallel interface cable

When a printer is connected, connect the parallel interface cable to the CH3 connector on the side of the IE-75001-R. When a host machine is connected, connect the interface cable to the CH4 connector on the side.

Photo 3-2 Side View of IE-75001-R
(Connecting the Interface Cable)



3.2 Setting the RS-232-C Modes

To set the RS-232-C mode switches, open the cover of these switches on the side of the IE-75001-R.

Fig. 3-2 RS-232-C Mode Switches

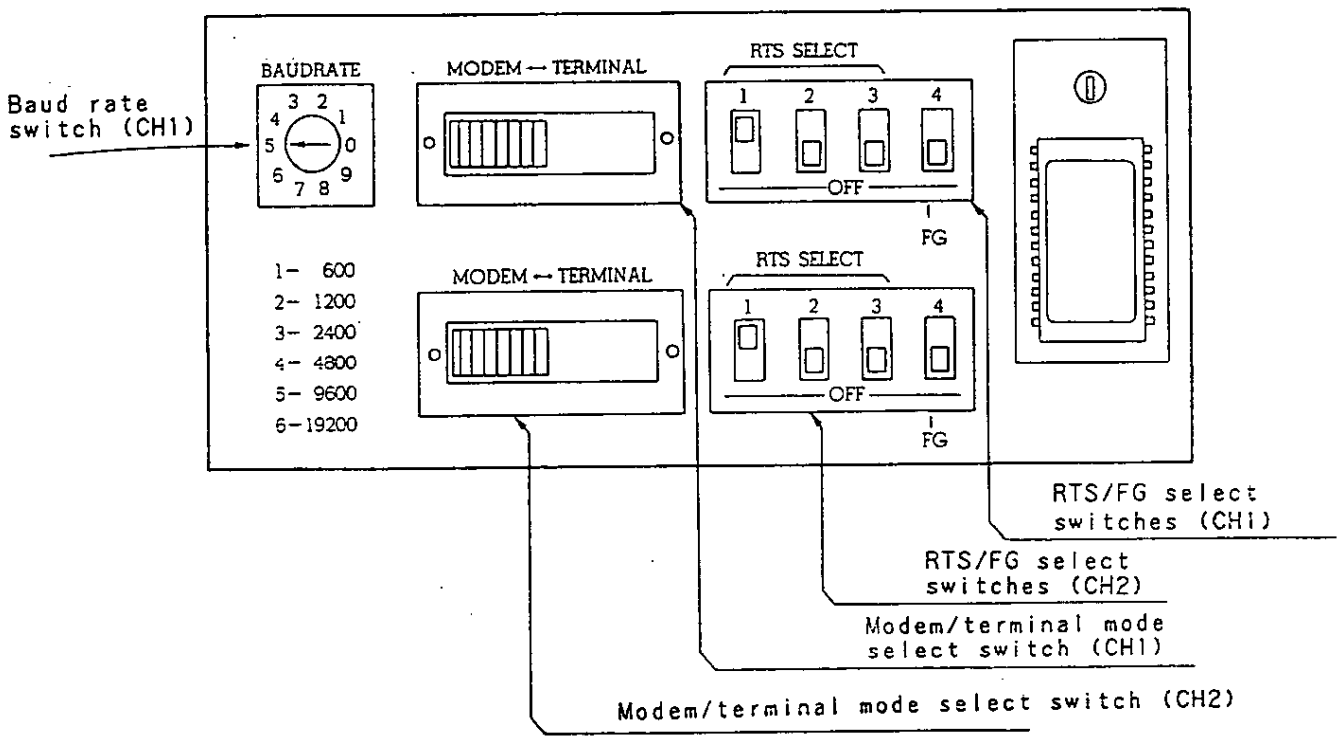
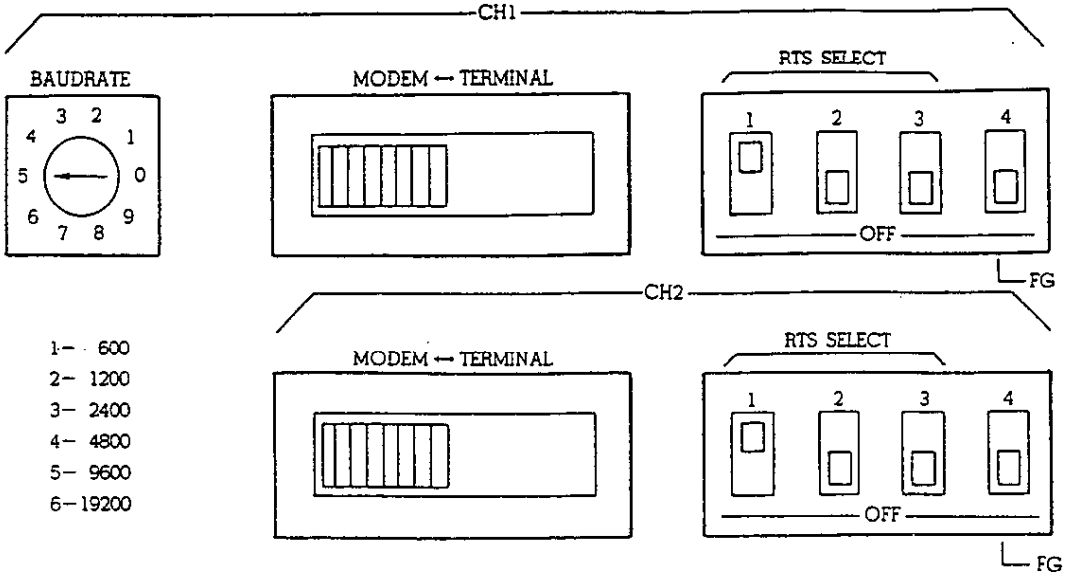


Fig. 3-3 Factory-Settings of the RS-232-C Mode Switches



Setting the modem/terminal mode select switch:

This is a slide switch. To set the modem mode, slide the switch from right to left when viewed from the side panel. To set the terminal mode, slide the switch from left to right.

This switch is factory-set to the modem mode.

Setting the RTS select switches:

These are DIP switches. To set RTS, set the switches to the ON position (upward). To release RTS, set the switches to the OFF position (downward).
RTS is set by switches 1 to 3.
Switch 1 is factory-set to ON and switches 2 and 3 are factory-set to OFF.

Setting the FG (frame ground) select switch:

This is a DIP switch. To set FG, set the switch to the ON position (upward). To release FG, set the switch to the OFF position (downward).
FG is set by switch 4.
Switch 4 is factory-set to OFF. (The frame ground (FG) and signal ground (SG) are in the open state.)

Setting the baud rate switch:

This is a micro DIP switch.
This switch has 10 (0 to 9) positions, but positions 7 through 9 are not used.
This switch is used to set the baud rate for CH1. The factory-setting of this switch is position 5 (9600 bps).
To set the baud rate for CH2, use an MOD command.
For details, see Chapter 8.

Setting the character specification:

The setting for CH1 cannot be changed.
The specification for CH2 is set with an MOD command.
See Table 3-1 below.

Table 3-1 Character Specification

Character specification	CH1 (fixed)	CH2 (set with a command)
Character length	8 bits	7 or 8 bits
Parity bit	None	Even, odd, or none
Stop bits	2 bits	1 or 2 bits

3.3 Setting the Control/Trace Board

Table 3-2 lists the factory-setting of the jumper on the control/trace board. Do not change this jumper setting.

In normal use, this setting need not be changed.

Table 3-2 Factory-Setting of the Jumper on the Control/Trace Board

Jumper No.	Setting
JP1	1-2 short

Caution: Follow the procedure below to connect the control/trace board with the IE-75001-R.

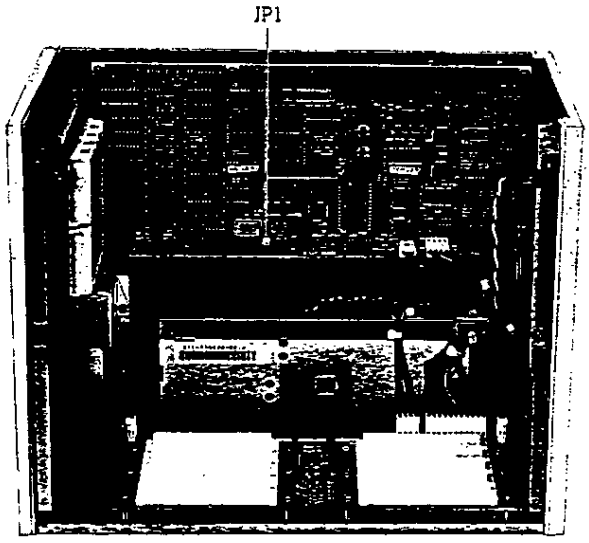
Remark: Follow the procedure below to connect the control/trace board with the IE-75001-R.

① Remove the six screws on the top of the IE-75001-R to open the cover.

② Disconnect the J1-J2 cable which connects the control/trace board and the IE-75000-R-BK.

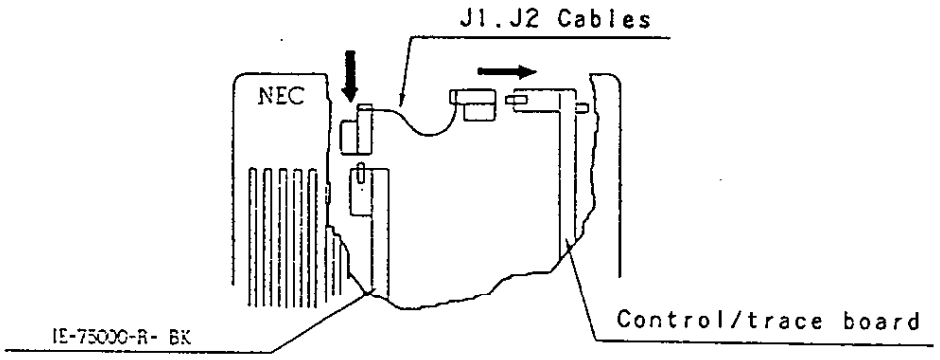
③ Remove all boards in the slot. (In this case, pull the card pullers on either end of each board.)

Photo 3-3 Inside of the IE-75001-R



④ See Figure 3-3 when connecting the J1-J2 cable again.

Fig. 3-4 Connection Diagram of the J1-J2 Cable



CHAPTER 4 SYSTEM CONFIGURATION

This chapter explains the methods for connecting the IE-75001-R and the following peripheral devices. Read this chapter before starting connection. In this chapter, it is assumed that the IE-75000-R-EM is connected to the IE-75001-R.

Host machine: PC-9800 Series
IBM PC Series

PROM programmer: PG-1500
PG-1000 (We have stopped manufacturing this model. It is available only as maintenance parts.)

PC-9800 Series

The PC-9800 Series provides a consistent development environment for software development up to the overall evaluation of a system including hardware by running optional control program on MS-DOS™.

IBM PC Series

The IBM PC Series provides a consistent development environment for software development up to the overall evaluation of a system including hardware by running optional control program on PC DOS™.

PG-1500

The PG-1500 is a PROM programmer for programming typical 256K-bit to 1M-bit PROMs. Use of an optional socket adapter enables the programming of PROMs built in NEC single-chip microcomputers.

The PG-1500 has key panel switches and a serial interface, so it can operate as a stand-alone PROM programmer. It can also operate as a PROM programmer for remote operations through a console connected to the serial interface.

When connecting the PG-1500 to the IE-75001-R, use a general RS-232-C interface cable.

PG-1000

The PG-1000 is a PROM programmer for programming bipolar PROMs and single-chip microcomputers built in PROMs by replacing the optional personal module. The PG-1000 has key panel switches and a serial interface, so it can operate as a stand-alone PROM programmer. It can also operate as a PROM programmer for remote operations through a console connected to the serial interface.

When connecting the PG-1000 to the IE-75001-R, use an RS-232-C interface cable attached to the PG-1000.

4.1 Connecting a Host Machine

4.1.1 Connecting a PC-9800 Series

When connecting a PC-9800 Series to the IE-75001-R, follow the procedure below:

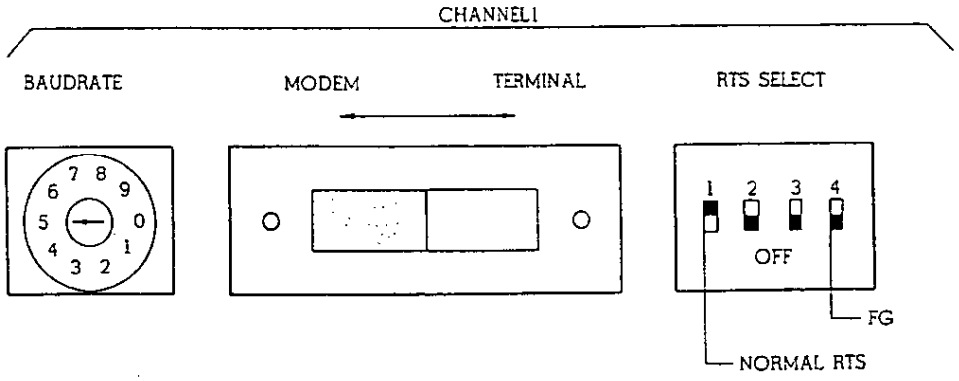
① Turn off the power for the PC-9800 Series and IE-75001-R.

② Set Channel 1 of the IE-75001-R as shown in Table 4-1.

Table 4-1 Channel 1 Setting (Connecting a PC-9800 Series)

Item	Setting
Baud rate	9600 bps
Mode	Modem mode
RTS	Standard RTS

Fig. 4-1 Channel 1 Setting (Connecting a PC-9800 Series)



③ Set the PC-9800 Series as shown in Table 4-2.

Table 4-2 PC-9800 Series Setting

Item	Setting
Baud rate	9600 bps
Character length	8 bits
Parity check	None
Stop bits	2 bits
X-on/X-off	None

④ Connect the standard RS-232-C channel on the rear of PC-9800 Series and channel 1 of the IE-75001-R using the RS-232-C cable attached to the IE-75001-R.

When using a parallel interface, connect the printer connector on the rear of the PC-9800 Series and channel 4 of the IE-75001-R using the printer cable for the PC-9800 Series.

Table 4-3 Connecting a PC-9800 Series

IE-75001-R	Connection	PC-9800 Series
Channel 1	RS-232-C cable ←→	Standard RS-232-C channel
Channel 4	Printer cable ←	Printer connector

⑤ Turn on the PC-9800 Series, then turn on the IE-75001-R. However, the IE-75001-R must be turned off first in the turn-off sequence.

4.1.2 Connecting an IBM PC Series

When connecting an IBM PC Series to the IE-75001-R, follow the procedure below:

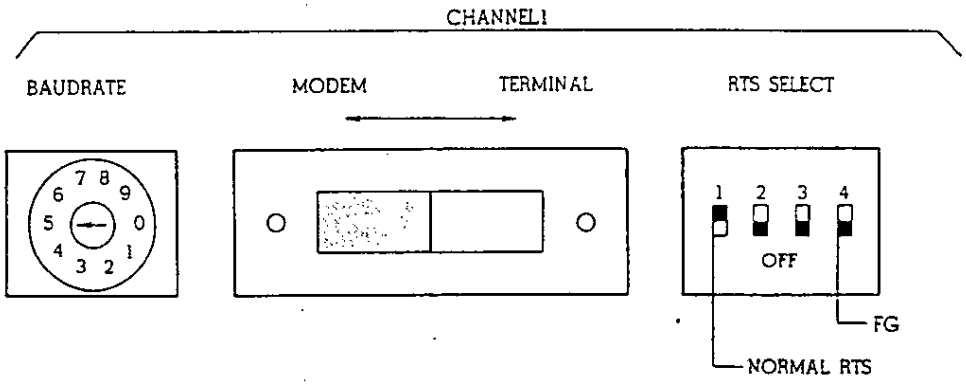
① Turn off the power for the IBM PC Series and IE-75001-R.

② Set Channel 1 of the IE-75001-R as shown in Table 4-4.

Table 4-4 Channel 1 Setting (Connecting an IBM PC Series)

Item	Setting
Baud rate	9600 bps
Mode	Modem mode
RTS	Standard RTS

Fig. 4-2 Channel 1 Setting (Connecting an IBM PC Series)



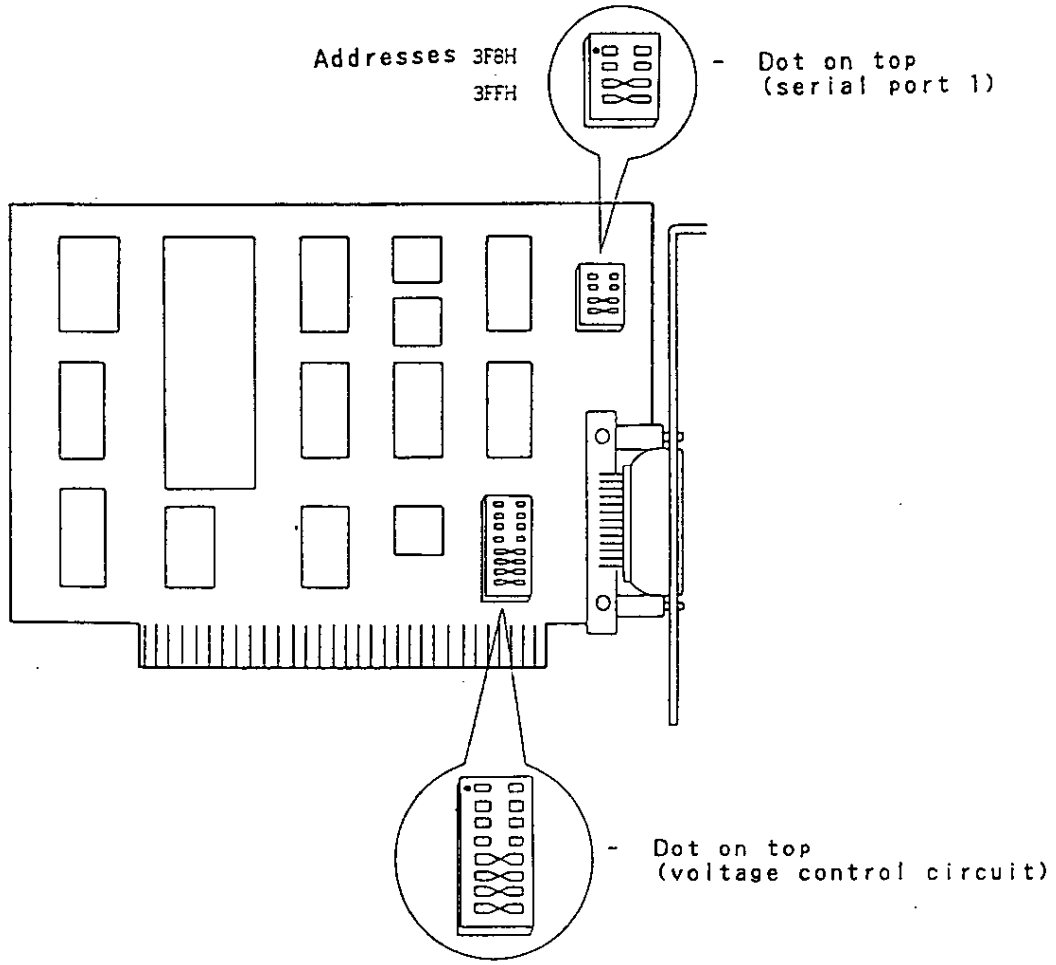
③ Set the IBM PC Series as shown in Table 4-5.

Table 4-5 IBM PC Series Setting

Item	Setting
Baud rate	9600 bps
Parity check	None
Data bits	8 bits
Stop bits	2 bits

④ Set the asynchronous communication adapter inserted in the IBM PC Series as shown in Figure 4-3. This control program supports only serial port 1 (No. 0).

Fig. 4-3 Setting the Asynchronous Communication Adapter



⑤ Connect the RS-232-C channel of the asynchronous communication adapter of the IBM PC option and channel 1 of the IE-75001-R using the RS-232-C cable for the IBM PC Series (see Figure 4-4).

When using a parallel interface, connect the printer connector on the rear of the IBM PC Series and channel 4 of the IE-75001-R using the printer cable for the IBM PC Series.

Fig. 4-4 Connecting the IBM PC and RS-232-C

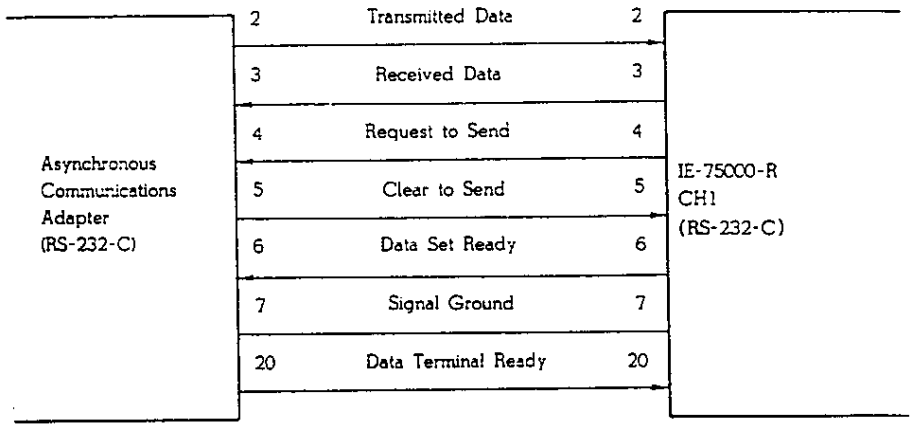


Table 4-6 Connecting an IBM PC Series

IE-75001-R	Connection	IBM PC Series
Channel 1	RS-232-C cable ←→	RS-232-C channel of the asynchronous communication adapter
Channel 4	Printer cable	Printer connector

⑥ Turn on the IBM PC Series, then turn on the IE-75001-R. However, the IE-75001-R must be turned off first in the turn-off sequence.

4.2 Connecting the PG Series

4.2.1 Connecting the PG-1500

When connecting the PG-1500 to the IE-75001-R using an RS-232-C interface, follow the procedure below:

① Turn off the power for the IE-75001-R and PG-1500.

② Connect the serial interface connector of the PG-1500 and channel 2 of the IE-75001-R using an RS-232-C cable.

Table 4-7 Connecting the PG-1500

IE-75001-R	Connection	PG-1500
Channel 2	RS-232-C cable ⟷	Serial interface connector

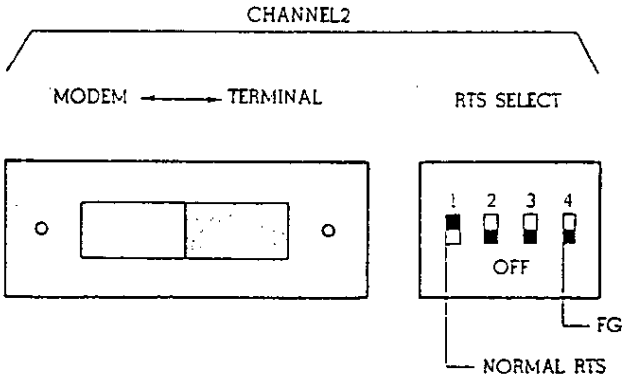
③ Turn on the PG-1500, then turn on the IE-75001-R.

④ Set channel 2 of the IE-75001-R as shown in Table 4-7. See Section 8.4.23 for setting the handshaking, baud rate, and character specifications.

Table 4-8 Channel 2 Setting (Connecting the PG-1500)

Item		Setting
Mode selection		Terminal mode
Frame ground		Set pin 4 to OFF.
RTS select		Set pin 1 to ON, and pins 2 and 3 to OFF.
Handshaking		CHAR (set with a command)
Baud rate		9600
Character specification	Data length	8
	Parity bit	None
	Stop bits	2

Fig. 4-5 Channel 2 Setting (Connecting the PG-1500)



⑤ Set the PG-1500 by the entry from the keyboard. Refer to the PG-1500 user's manual for details.

Table 4-9 PG-1500 Setting

Item	Setting	Marking	Parameter
Baud rate	1200 2400 4800 [9600] 19200 (bps)	BR	9600
Parity	[NON] (none) EVN (even) ODD (odd)	P	NON
X-on/X-off control	[ON] OFF	XN	OFF
Bit configuration	7 [8]	B	8
Stop bits	1 [2]	SB	2
Precheck	ON [OFF]	PC	OFF

Remark: The settings in a box are factory-settings.

4.2.2 Connecting the PG-1000

When connecting the PG-1000 to the IE-75001-R, follow the procedure below:

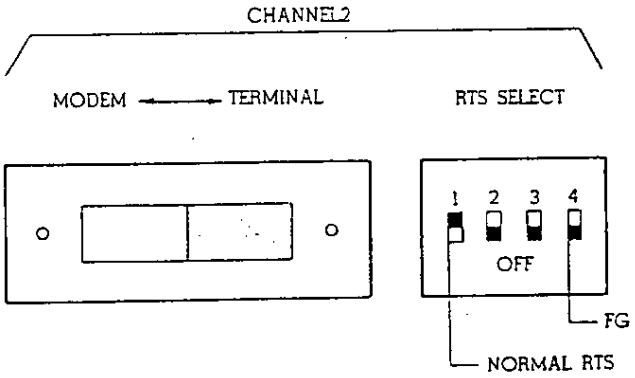
① Turn off the power for the IE-75001-R and PG-1000.

② Set channel 2 of the IE-75001-R as shown in Table 4-10. See Section 8.4.23 for setting the handshaking, baud rate, and character specifications.

Table 4-10 Channel 2 Setting (Connecting the PG-1000)

Item		Setting
Mode selection		Terminal mode
Frame ground		Set pin 4 to OFF.
RTS select		Set pin 1 to ON, and pins 2 and 3 to OFF.
Handshaking		CHAR (set with a command)
Baud rate		9600
Character specification	Data length	8
	Parity bit	None
	Stop bits	2

Fig. 4-6 Channel 2 Setting (Connecting the PG-1000)



③ Set the PG-1000 using the 4-bit and 8-bit DIP switches on the bottom of the body. Table 4-1 lists the settings of these DIP switches.

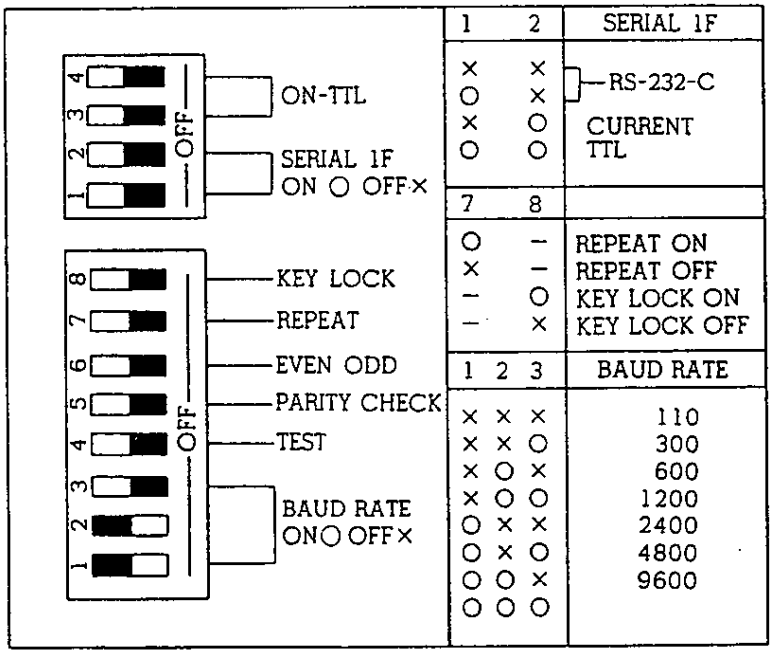
Table 4-11 PG-1000 Setting

Item	Switch No.	Setting
Serial interface mode	4-bit switch 1 to 4	1 to 4: ON
Baud rate	8-bit switch 1 to 3	1 and 2: ON 3: OFF
Parity check	5 and 6	5: OFF 6: ON/OFF (*)

* Either ON or OFF may be set.

Caution: Do not set bits 4, 7, and 8 to ON.

Fig. 4-7 Setting the PG-1000 (Bottom DIP Switches)



④ Connect the serial interface connector of the PG-1000 and channel 2 of the IE-75001-R using a cable attached to the PG-1000.

Caution: Do not use a cable of other types.

Table 4-12 Connecting the PG-1000

IE-75001-R	Connection	PG-1000
Channel 2	RS-232-C cable ⟷	Serial interface connector

⑤ Turn on the PG-1000, then turn on the IE-75001-R.

CHAPTER 5 OPERATION OVERVIEW

This chapter explains the IE-75001-R system operating environment and the operation sequence from power-on to system termination and power-off. Before reading this chapter, the user must have set each peripheral device explained in Chapters 1 through 4.

5.1 System Operating Environment

The IE-75001-R which is connected to a host machine (PC-9800 Series or IBM PC Series) is controlled by the control program (IE75000.COM) running on the host machine (see Figure 5-1).

Fig. 5-1 System Operating Environment

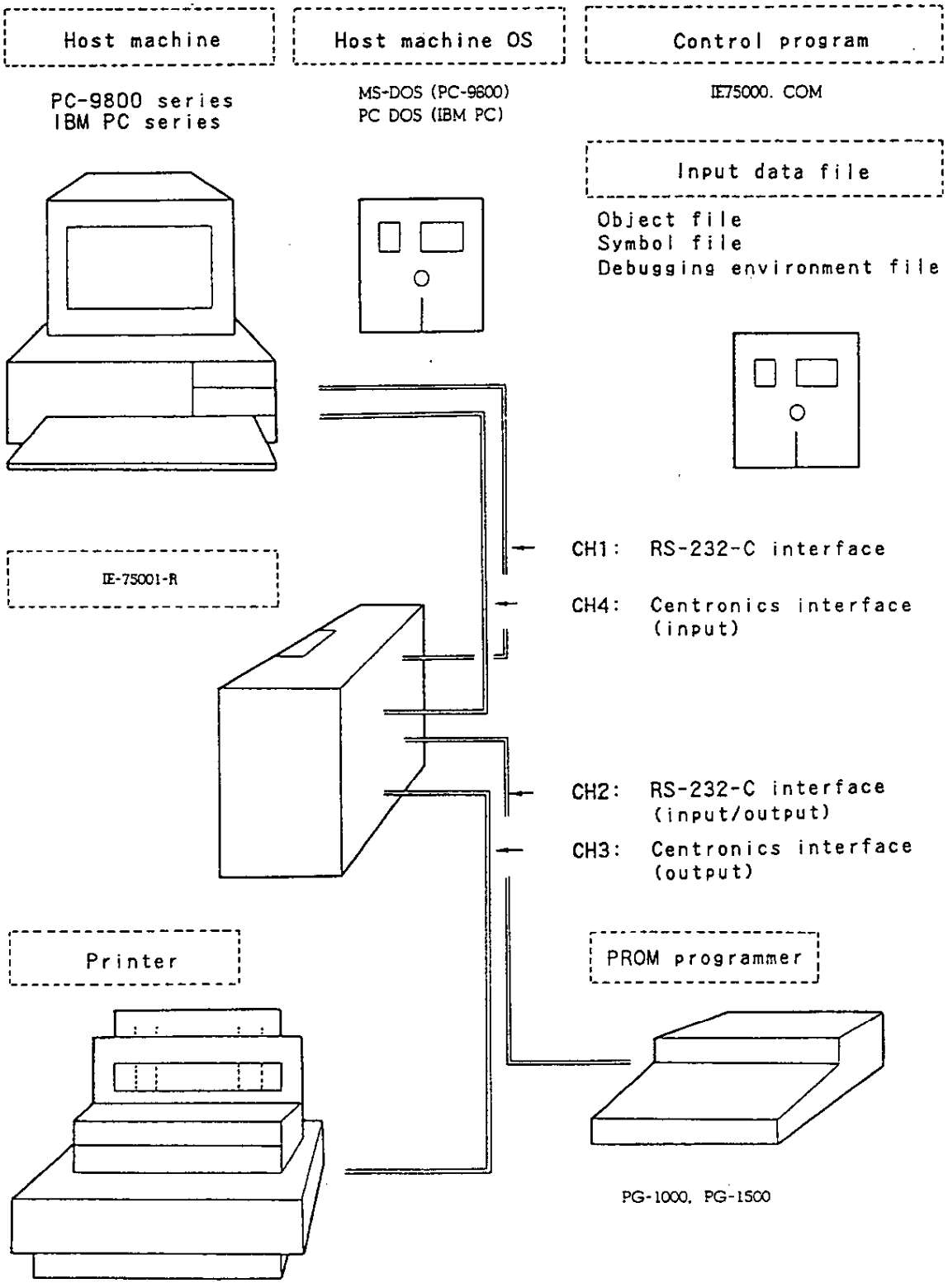


Table 5-1 Hardware Environment

Device used	Channel, peripheral	Use
IE-75001-R	CH1 (input/output) (RS-232-C interface)	Connected to a host machine. Used to transfer control data between the IE-75001-R and host machine. Also used to download and upload object files.
	CH2 (input/output) (RS-232-C interface)	Connected to a PROM programmer (PG Series). Used to transfer control data between the IE-75001-R and PROM programmer.
	CH3 (output) (Centronics interface)	Connected to a printer. Used to output data to the printer by the host machine. Also used for through-output of input data of CH4 (Centronics interface).
	CH4 (input) (Centronics interface)	Connected to a host machine. Used to download an object file at high speed from a host machine.
Host machine PC-9800 Series (except LT and XT) or IBM PC Series	Keyboard	Used for all operations such as command entry.
	FDD	Used for the following purposes: ① OS loading ② Control program loading ③ Data file loading/saving . Object file . SYmbol file . Debugging environment file

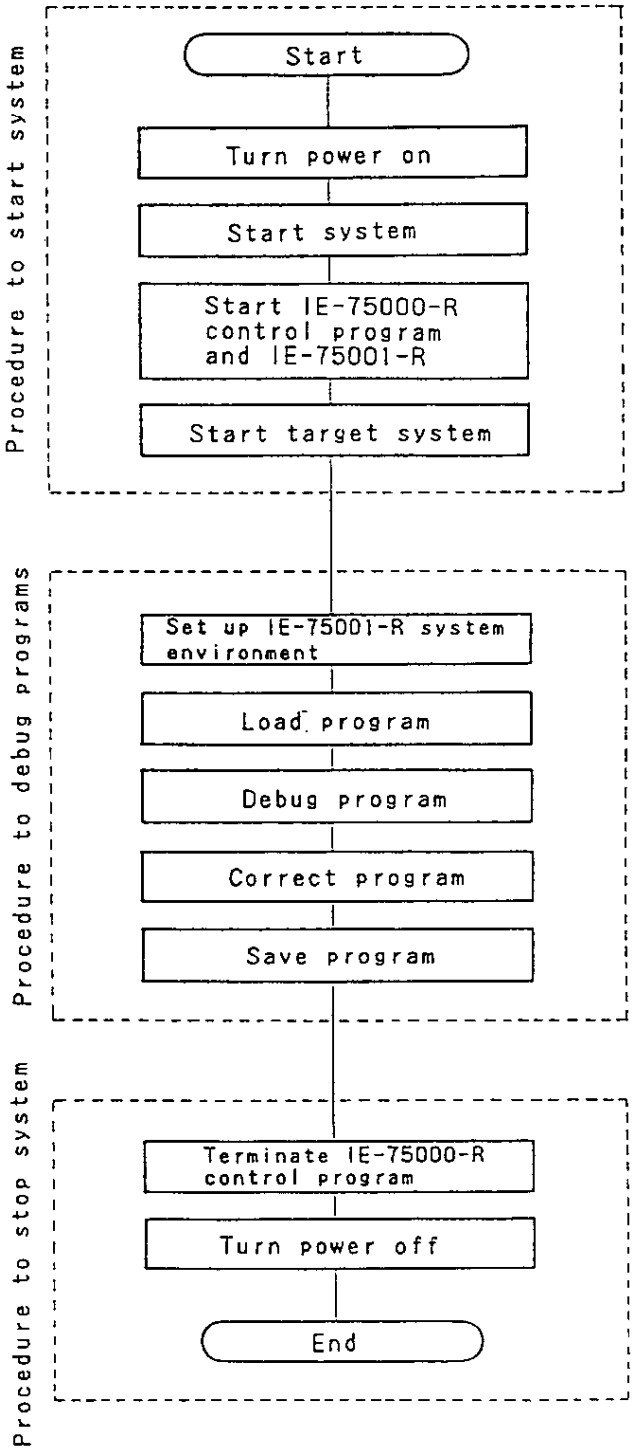
Table 5-2 Software Environment

Software used	Use
MS-DOS (Ver 3.30 and later) PC DOS (Ver 3.10 and later)	Used when a PC-9800 Series except LT and XT is used as a host machine. Used when an IBM PC Series is used as a host machine.
IE-75001-R control program (File name: IE75000.COM)	Control the IE-75001-R. This program runs on a host computer.
Object file (File name: xxxxxxxx.HEX)	Stores object codes of a target program.
Symbol file (File name: xxxxxxxx.SYM)	Stores user-defined symbols for a target program. Caution: This file need not be created. However, if this file is not created, no symbols can be processed.
Debugging environment file (File name: xxxxxxxx.DBG)	Stores information of a debugging environment. An environment can be set with the following commands when this file is created. BNK, BRA1, BRA2, BRA3, BRA4, BRK, BRM, BRS1, BRS2, CHK, CLK, DLY, MOD, OUT, PAS, RGM, REG, SET, STS, TRF, TRM, TRX, TRY Caution: This file need not be created. However, if this file is not created, no environment can be set.

5.2 Operation Sequence

Figure 5-2 shows the overview of the operation sequence of the IE-75001-R system.

Fig. 5-2 Flowchart of the Operation Sequence (Overview)



5.2.1 Procedure to start the system

(1) Power-on sequence

Turn on the power for all devices according to the following procedure. Note that the order in the power-on sequence is opposite to that in the power-off sequence.

① Turn on the power for a host machine.
② Turn on the power for the IE-75001-R.
③ When a PG Series is connected to the IE-75001-R, turn on the power for the PG Series.
④ When a printer is connected to the IE-75001-R, turn on the power for the printer.

(2) Starting the system

How to start the system depends upon whether MS-DOS (OS for PC-9800 Series) or PC DOS (OS for IBM PC Series) is used(Note).

① Starting an OS

Start an OS for the host machine. Any drive can be selected.

② Setting the serial channel of the host machine

Set the serial channel of the host machine as shown in Table 5-1. The command used for setting the serial channel varies with an OS used (MS-DOS or PC DOS).

Note: When using PC DOS, be sure to install ANSI.SYS in CONFIG.SYS before starting the system.

Table 5-3 Setting Serial Channel of Host Machine

Item	Setting
Baud rate	9600 bps
Character length	8 bits
Parity	None
Stop bits	2
X-on/X-off	On or off

Remark: The descriptions in a box (monitor screen) mean as follows:

xxxxx: Key entry
<cr>: Return key
[ESC]: Escape key
^: Control key ([CTRL] key)
^X: [X] key with the control key held down

Setting with MS-DOS

Use a SPEED command to set the serial channel.

```
A>SPEED <cr>                                                    ①

  SPEED Version X.X
  RS232C-0 XXXX BITS-X PARITY-XXX STOP-X XXXX
- RS232C-0 9600 BITS-8 PARITY-NONE STOP-2 <cr>                  ②

A>^C
```

① Enter a SPEED command.
② The serial channel is set.

Caution: When using MS-DOS, be sure to install RSDRV.SYS in CONFIG.SYS before starting the system.

Setting with PC DOS

Use a MODE command to set the serial interface.

```
A>MODE COM1:mode <cr>                    ①
  COM1:XXXX,X,X,X,-                      ②
```

① Enter a MODE command.
② The serial channel is set.

(3) Starting the IE-75000-R control program and IE-75001-R

Start the IE-75000-R control program (IE75000.COM) and IE-75001-R by the following the procedure, which will connect the host machine and the IE-75001-R logically.

① Set the IE-75000-R control program (IE75000.COM, IE75000.OV1, IE75000.OV2, IE75000.OV3, or IE75000.OV4) in a floppy disk drive of the host machine.

② Change the current drive specification to the drive in which the control program is set.

③ Enter the name of the control program (IE75000).

Specification screen for MS-DOS

```
A>B: <cr>                                                    ①

B>IE75000 <cr>                                               ②

     IE-75000 CONTROLLER (PC-9800 SERIES) Vx.x [DD Mmm YY]   ③
            Copyright(C) YYYY by NEC Corporation

     IE-75000/1-R Monitor V1.3 [1 Nov 91]                    ④
            Copyright(C) 1989, 1991 by NEC Corporation
```

① Change the drive.
② Enter the name of the control program (IE75000).
③ Version No. of the IE-75000
④ Version No. of the IE-75001-R monitor

Specification screen for PC DOS

```
A>B: <cr>                                                    ①

B>IE75000 <cr>                                               ②

     IE-75000 CONTROLLER (IBM PC SERIES) Vx.x [DD Mmm YY]    ③
            Copyright(C) 1989 by NEC Corporation

     IE-75000/1-R Monitor V1.3 [1 Nov 91]                    ④
            Copyright(C) 1989, 1991 by NEC Corporation
```

① Change the drive.
② Enter the control program name (IE75000).
③ Version No. of the IE-75000
④ Version No. of the IE-75001-R monitor

(4) Starting the target system

To debug the target system which is connected to the IE-75001-R, start the target system according to the following procedure. Thus, the IE-75001-R and the target system are logically connected.

First turn on the power for the target system, then enter an appropriate key.

```
Self check ok

Target CPU  uPD75104/104A/106                                    ①
Program Memory        0-FFFFH
Data Memory           00H-13FH, F80-FFFH
Memory Bank           0-1, 15
Register Bank         0-3
Power on target system (Y/N)Y <cr>                               ②
 Create new set up mode (Y or N) :Y <cr>                         ③
 Do you use high speed down load mode? (Y/N)=N <cr>              ④
 Lod object file name=TEST C S <cr>
brk :0>
```

① Start the target system after selecting the target device set by the DIP switch (SW1) of the IE-75000-R-EM. The uPD75104, uPD75104A, uPD75106 are set here.

② Enter Y.

③ Enter Y.

④ Enter N.

(5) When setting set-up file

Set and create a set-up file.

Note: Be sure to set a set-up file by the control program Ver.1.1 and the Firmware ROM Ver.1.4. This is because the parameter that can be automatically set is only the high-speed down loading if the control program Ver.1 is used with the Firmware ROM Ver.1.0 through 1.3.

(a) When creating a set-up file

```
Create new set up mode (Y or N) :Y <cr>                    ①
Do you use high speed down load mode? (Y/N)=N <cr>          ②
Lod object file name=TEST C S <cr>                         ③
```

① Enter Y to create a set-up file.

② Select high-speed download. In this example, N is entered, therefore, high-speed download is not selected.

③ Enter the object file name and option (select load of the object file (.HEX) and the symbol file (.SYM)) to be loaded. This option should not be omitted. If omitted, automatic setting like (b) is impossible.

Note: When there is no set-up file (SETUP.STR) in the current directory, ① is omitted and the set-up file is generated automatically.

(b) When executing the system automatically from the set-up file.

```
Create new set up mode (Y or N) :N <cr>                    ①
Do you use high speed down load mode? (Y/N)=N
Lod object file name=TEST C S
```

① Enter N.

② The system is set automatically as the contents of set-up file.

③ Same as ②.

(c) When the setting is not set automatically from the set-up file.

When a set-up file is created without entering the option of object file name, the set-up file is not set automatically when loading the object file.

```
Create new set up mode (Y or N) :N <cr>                    ①
Do you use high speed down load mode? (Y/N)=N              ②
Lod object file name=TEST                                  ③
Debug condition load (Y/N)?=N <cr>                         ④
```

① Enter N.

② Set-up file is set automatically as the contents of set-up file.

③ Set-up file is set automatically as the contents of set-up file.

④ The system is not set automatically. Specify the input of the debug environment.

(6) Specifying high-speed downloading

When downloading a program from an object file of the host machine at high speed, specify high-speed downloading.

```
    Do you use high speed down load mode? (Y/N) Y <cr>     ①
brk:0>                                                     ②
```

① Enter Y. (Note)

② The brk:0> prompt is displayed and the break mode is enabled.

Note: The IE-75001-R must be connected to the host machine via the Centronics interface.

(7) Loading object file

Enter the object file name to be debugged.

```
Lod object file name=TEST C S <cr>
```
(Note) ①

① Enter the object file name and option to be loaded. Do not add the extension (.HEX).

Note: Enter the object file name [C / D / S]

C: Loads only object code

D: Loads only debug environment

S: Loads only symbol

Default: Loads all three files above (Load of the debug environment is selective.)

When unloading object file

```
 Lod object file name= <cr>                                    ①
brk: 0>                                                        ②
```

① Enter the return key

② brk:0> is displayed as a prompt and the program enters break mode.

(8) End of starting the system

A series of operations described above make the prompt to brk:n> (break mode) and starting the system ends. From this point, it is possible to enter various commands of the IE-75000-R control program in order to debug the target program.

5.2.2 Procedure to debug programs

When the start of the target system ends, it is possible to debug programs using various commands in the IE-75000-R control program. The following outlines the debugging procedure.

See Chapter 6 for the basic functions of commands.

(1) Setting an IE system environment

Set a system environment in the following procedure so that the IE-75001-R can debug the target system.

. Reset the target device.

. Select the clock.

. Load a debugging environment file. (This file can be loaded together with a program and symbol file.)

(2) Loading a program

Load the program to be debugged into the IE-75001-R in either of the following two methods.

. Loading a program from an object file of the host machine.

. Loading a program from the PG Series (connected to CH2).

(3) Debugging the program

Debug the program with various commands and correct it.

(4) Saving the program

Save the debugged program into the target medium in either of the following two methods.

. Save the program in an object file of the host machine.

. Save the program in the PG Series (connected to CH2).

5.2.3 Procedure to stop the system

(1) Terminating the IE-75000-R control program

After all the debugging is completed, terminate the IE-75000-R control program by the EXT command and return the control to OS.

(2) Power-off sequence

Turn off the power for all devices according to the following procedure. Note that the order in the power-off sequence is opposite to that in the power-on sequence.

① Remove all media such as floppy disks.

② Turn off the power for the target system.

③ When a printer is connected to the IE-75001-R, turn off the power for the printer.

④ When a PG Series is connected to the IE-75001-R, turn off the power for the PG Series.

⑤ Turn off the power for the IE-75001-R.

⑥ Turn off the power for the host machine.

CHAPTER 6 USE OF BASIC FUNCTIONS

This chapter describes major debugging functions provided by the IE-75001-R and also explains the basic use of them.

Users who will use the IE-75001-R for the first time should read Sections 6.1 and 6.2 to understand basic functions, then start operations following the basic debugging procedure explained in Section 6.3. Section 6.4 gives some examples indicating how to use the basic functions of the IE-75001-R in conjunction with the IE-75000-R-EM.

6.1 System Operation Modes and Command Input

A system operation mode indicates the operation state of the system at a given point of time: whether the target program execution (emulation) function and the trace function are operating.

There are three system operation modes as listed below. Note that command input is restricted according to the system operation modes.

Break mode (brk:n>)

Neither the target program execution (emulation) function nor trace function is operating.

Emulation mode (emu:n>)

The target program execution (emulation) function is operating, but the trace function is stopped. The emulation mode is used when the execution of the target program must not be interrupted.

Trace mode (trc:>)

The target program execution (emulation) function and the trace function are both operating.

(1) Prompt indication and system operation modes

The user can recognize the current system operation mode by the prompt output on the screen by the monitor. When no prompt appears on the screen, no command can be accepted.

System operation mode	Prompt	CPU status	Tracer status
Break mode	brk :n>	Stopped	Stopped
Emulation mode	emu :n>	Operating	Stopped
Trace mode	trc :n>	Operating	Operating

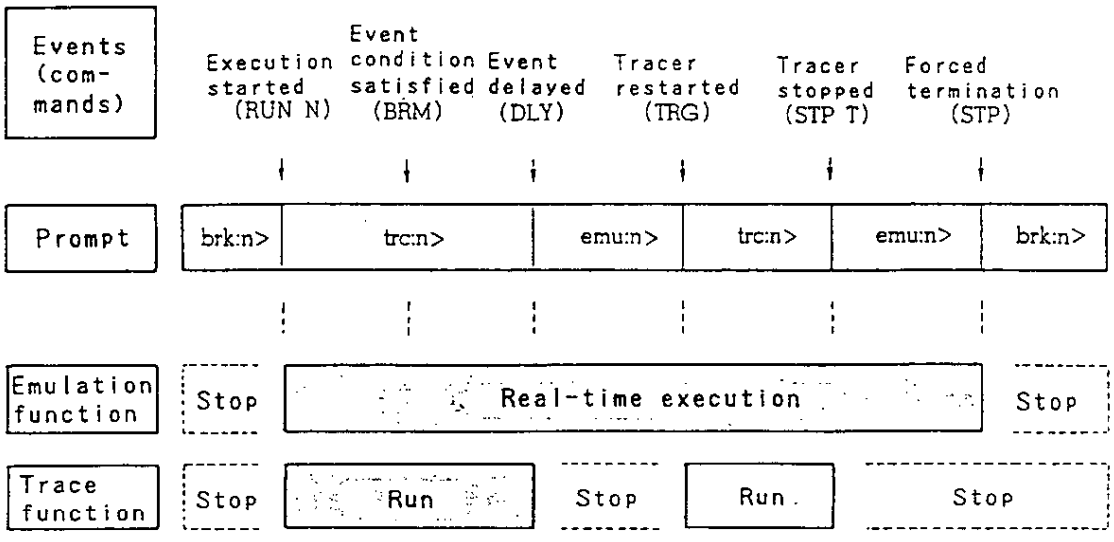
: Indicates a system operation mode. For commands available in each system operation mode, see Section 8.1.2.

n: Serial port No. in the host machine

(2) Interrelation between prompt indication and system operation mode

Figure 6-1 shows an interrelation between prompt indication, and the emulation device and trace function.

Fig. 6-1 Example of Prompt Indication and System Operation States (1)



6.2 Use of Basic Functions

This section explains basic functions of the various debugging functions provided by the IE-75001-R.

The explanation is made along the basic debugging procedure described in Section 6.3. For easier understanding, it is recommended to read this section to know the outline of the basic functions, then go to practical operations with referring to the description of the basic debugging procedure.

6.2.1 Clock selection function

The clock selection function specifies a clock source to be supplied to the target device. One of the following two clock sources can be selected:

Clock in the IE-75001-R (CLK I command)
Clock in the target system (CLK U command)

- Fixed clock (4.19 MHz) is selected at starting.
- When selecting the clock in the target system, setting the same clock is also required for the emulation board.
- When executing "CLK U" command without setting a clock for the emulation board, "E-CPU ERROR" message is displayed.

Note: When a clock is selected, the RES command is issued internally.

```
PORT xxH, xxH DATA xx
xx
E-CPU  ERROR!!
```

6.2.2 Reset function

The reset function resets the whole IE-75001-R system or an emulation device.

Resetting the IE-75001-R system (RES H command)

The whole system of the IE-75001-R is reset. When reset, the IE-75001-R is restored to the operating environment present at the start of the IE-75001-R.

Resetting the emulation device (RES command)

Only the emulation device is reset. Trace memory and coverage memory are cleared.

The RES command must be executed when performing the emulation from the top of a program.

6.2.3 Coverage function

When a part of the target program has been executed with a RUN command (real-time emulation), the coverage function assigns an identifier to that part.

With these identifiers, the user can easily find which part of the target program has already been executed and which part is not yet executed.

The following functions are provided to control the coverage function:

Setting a coverage measurement range

- Setting and adding a coverage measurement range (CVM A command)

- Canceling a coverage measurement range (CVM K command)

- Displaying a coverage measurement range (CVM D command)

Manipulating coverage measurement results

. Deleting coverage measurement results
  (CVD K command)

. Displaying coverage measurement results
  (CVD D command)

6.2.4 Load function

The load function loads a target program stored in a device connected to a channel of the IE-75001-R into the IE-75001-R memory area. From the host machine, debugging environment information and the symbol table can also be loaded.

Device to be connected	Channel in IE-75001-R	Type of loading
Host machine (PC-9800 series, IBM PC series)	CH1 (for I/O)	Standard downloading, standard uploading
PROM programmer (PG-1000/1500)	CH2 (for I/O)	Standard downloading, standard uploading
Host machine	CH4 (for input only)	High-speed downloading (*)

* To perform high-speed downloading, specify so at the start of the system. For details, see the description of the LOD command in Section 8.4.19.

(1) Loading from the host machine

Object, debugging environment, and symbol files on a disk device connected to the host machine can be loaded into the IE-75001-R memory area with the LOD command.

Usually, data in the host machine are downloaded via channel 1 (CH1) at a normal speed. When high-speed downloading is specified at the start of the system, data are downloaded via channel 4 (CH4) at high- speed.

Table 6-1 briefly explains the files that can be downloaded, and Table 6-2 lists the commands available for loading and their functions.

Table 6-1 Files to Be Downloaded from the Host Machine

File	Description
Object file (file name: xxxxxxxx.HEX)	Contains the object code of a target program (Intel hexadecimal format)
Symbol file (file name: xxxxxxxx.SYM)	Contains user-defined symbols for the target program
Debugging environment file (file name: xxxxxxxx.DBG)	Contains debugging environment information. This file can hold the following commands for environment setup: BRA1, BRA2, BRA3, BRA4, BRK, BRM, BRS1, BRS2, CHK, CLK, DLY, MOD, OUT, PAS, PGM, REG, STS, TRF, TRM, TRX, TRY

Notes 1. The symbol file may be omitted. When it is omitted, however, no symbols can be processed.

Append symbols are loaded with the SYM L command.

2. The debugging environment file may be omitted. When it is omitted, however, an environment can not be set up.

Table 6-2 Load Commands and Their Functions

Command	Function
LOD file	Loads all the object, debugging environment, and symbol files at a time.
LOD file C	Loads the object file only. Remark: When the object file has been loaded, the address range of the target program is set unconditionally as the coverage measurement range.
LOD file D	Loads the debugging environment file only.
LOD file S	Loads the symbol file only.

(2) Loading from the PROM programmer

Object code in the Intel hexadecimal format on a PROM programmer (PG series) connected to channel 2 (CH2) can be loaded in the IE-75001-R memory area.

To load data from the PROM programmer, an environment must be set up as follows:

① Setting the channel 2 mode

The communication active state is set for serial channel 2 (CH2) with the MOD command.

Remark: See Chapter 9 for hardware setting.

② Setting the PG mode (terminal mode)

The PG mode (terminal mode) is set by establishing the interface between serial channel 2 and the PROM programmer with the PGM command. The PG mode enables uploading and downloading of object code from and to the PROM

programmer.

③ To end the terminal mode, enter ^Z (CTLR + Z).

6.2.5 Run emulation function

The run emulation function triggers emulation and tracing of the target program for the target device.

This function is classified as follows, according to how emulation runs:

Real-time execution

- Nonbreak real-time execution function (RUN N command)
- Real-time execution under break condition (RUN B command)

Nonreal-time execution function

- Step-by-step execution function (RUN T command)

(1) Real-time execution function

The real-time execution function includes a nonbreak real-time execution and real-time execution under break conditions. The former function does not specify any break in executing the target program, and the latter specifies breaks in running the target program.

(a) Nonbreak real-time execution function (RUN N command)

Nonbreak real-time execution starts execution and tracing of the target program at a specified address, and stops only tracing when a specified event is detected.

At the point when tracing is stopped, the system enters the emulation mode (emu:n>).

(i) Trace stop condition

Tracing is stopped usually when an event detection condition set in one of the following registers is satisfied:

. Event mode register (BRM command)

One or more event condition registers must be specified in the BRM command.

. Event condition registers (BRA1 to BRA4, BRS1 and BRS2)

Detail conditions of event detection must be set in the event condition registers. For how to set the conditions of event detection, see Section 6.2.9.

Tracing can also be stopped forcibly.
The forced termination will be explained in (iii) and (iv) in this section.

(ii) Restart of tracing (TRG command)

Tracing is restarted in the emulation mode.

(iii) Manual stop of tracing (STP T command)

In the trace mode (trc:n>), only tracing can be stopped with the STP T command.

(iv) Forced termination

Forced termination functions to stop emulation and tracing of the target program are provided (manual break and fail-safe break).

. Manual break (STP command)

. Fail-safe break

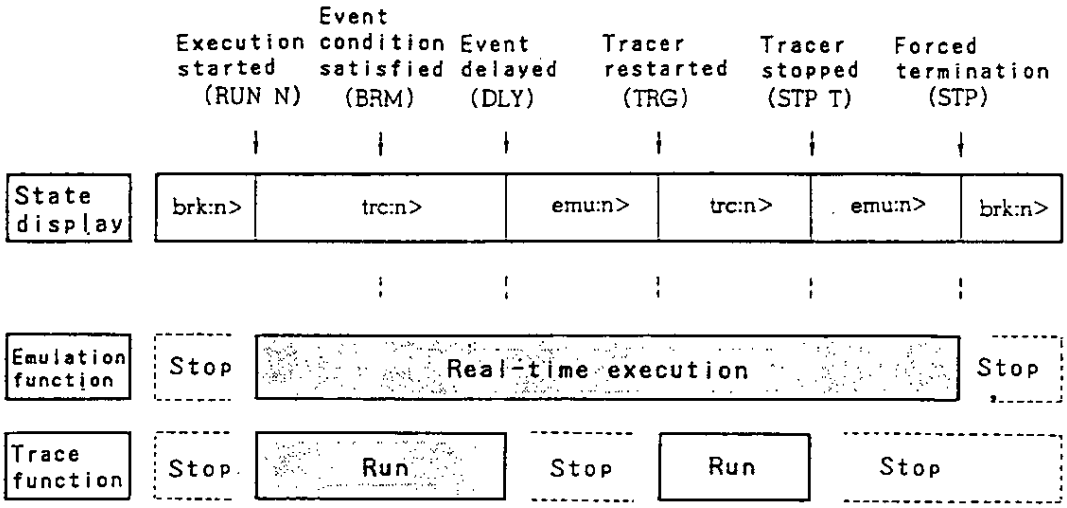
A fail-safe break occurs when the target program tries to access an area not present in the target device, such as DMEM (data memory), a SPR (special register), a register area, and a stack area.

For details on forced termination (forced break), see Section 6.2.6.

(v) Interrelation between the system operation states

Figure 6-2 shows an interrelation between the status indication (prompt indication), CPU operation, and tracer operation in the nonbreak real-time execution.

Fig. 6-2 Example of Prompt Indication and System Operation States (2)



(b) Real-time execution under break conditions (RUN B command)

The function for real-time execution under break conditions starts executing the target program at a specified address, and stops the operation of the emulation device and tracer when detecting a specified event condition.

As the emulation device and tracer stop when the event is detected, one-step execution mode is entered automatically.

(i) Break due to event detection

A break takes places place under the conditions set in the event mode register (BRM) and event condition registers (BRA1 to BRA4, BRS1 and BRS2).

For how to set conditions of event detection, see Section 6.2.9.

(ii) Break due to forced termination

In addition to a break due to event detection, execution (emulation) and tracing of the target program can be stopped by forced termination (manual break and fail-safe break).

- Manual break (STP)

- Fail-safe break

  A fail-safe break occurs when the target program tries to access an area not present in the target device, such as DMEM (data memory), an SPR (special register), a register area, and a stack area.

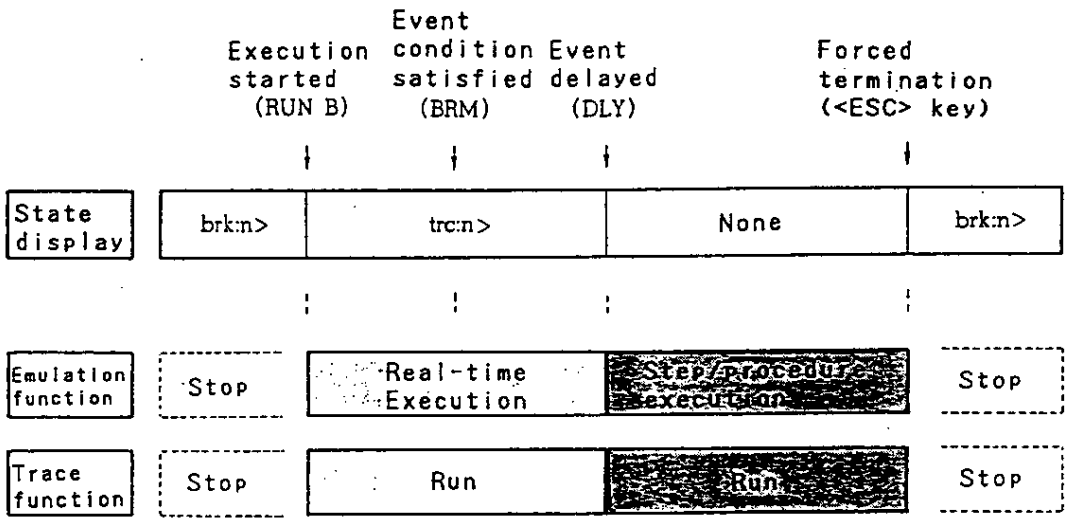
(iii) One-step execution and procedure execution

For information on these functions, see the description of the nonreal-time execution function.

(iv) Interrelation between system operation states

Figure 6-3 shows an interrelation between the status indication, CPU operation, and tracer operation in real-time execution under break conditions.

Fig. 6-3 Example of Prompt Indication and System Operation States (3)



(2) Nonreal-time execution function

Nonreal-time execution functions are divided into step-by-step execution and procedure execution. The step-by-step function stops the user program after each instruction is executed, indicates or detects the internal register contents, and traces the execution of the program. The procedure execution function starts a routine, but does not indicate nor detect the internal registers nor perform tracing for routines at deeper nesting levels than the routine started.

(a) Step-by-step execution function (RUN T command)

The step-by-step execution function executes one instruction in the target program starting at a specified address, then indicates or detects the internal

register contents, and traces the execution.

Step-by-step execution continues until a specified condition expression becomes true, or until as many instructions as specified have been executed.

After step-by-step execution terminates, the one-step execution mode is entered.

(i) One-step execution mode

In the on-step execution mode, every time the <cr> is pressed, the target program is executed by one step.

During one-step execution, trace data, register contents, and disassembled results are indicated unconditionally, regardless of whether display of trace data (TRD) and display of registers (REG) are specified.

In the one-step execution mode, the procedure execution is enabled.

Note: One-step execution erases the trace data previously written in trace memory.

(ii) Forced termination

Functions for forced termination (manual break and fail-safe break) are provided to stop execution of the target program and tracing.

. Manual break

When the <ESC> key is entered

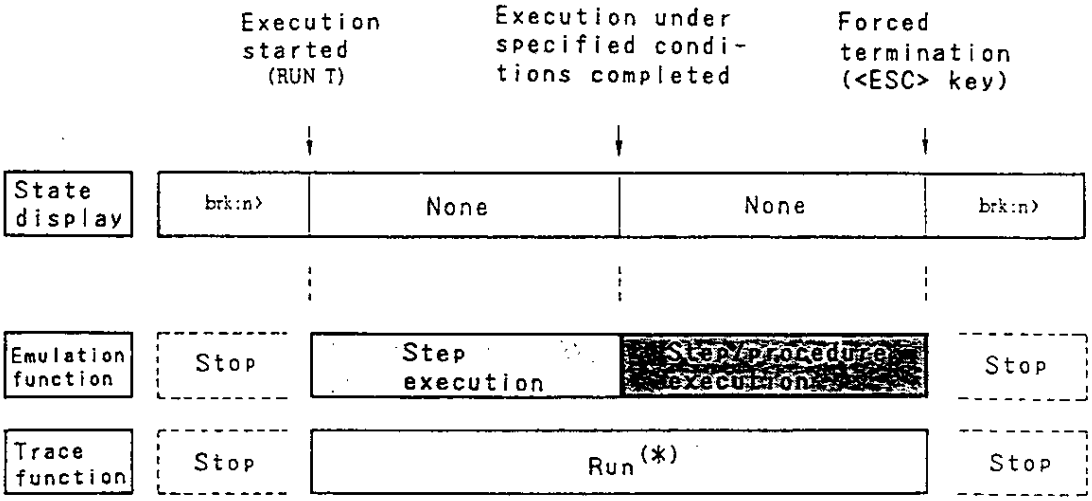
. Fail-safe break

When the target program tries to access an area not present in the target device, such as DMEM (data memory), an SPR (special register), a register area, and a stack area.

(iii) Interrelation between system operation states

Figure 6-4 shows an interrelation between the status indication, CPU operation, and tracer operation in the step-by-step execution.

Fig. 6-4 Example of Prompt Indication and System Operation States (4)



* Data in trace memory are erased every time the user program is executed by one step.

(b) Procedure execution function (RUN T command)

The procedure execution function is available only in the one-step execution mode.

The procedure execution function executes in real time a routine at a deeper nesting level than the level of the routine with which program execution was started. The function thus does not detect any event nor perform tracing.

The procedure execution function is operated in the following procedure:

① Execute each step of the program by pressing the <cr> key.

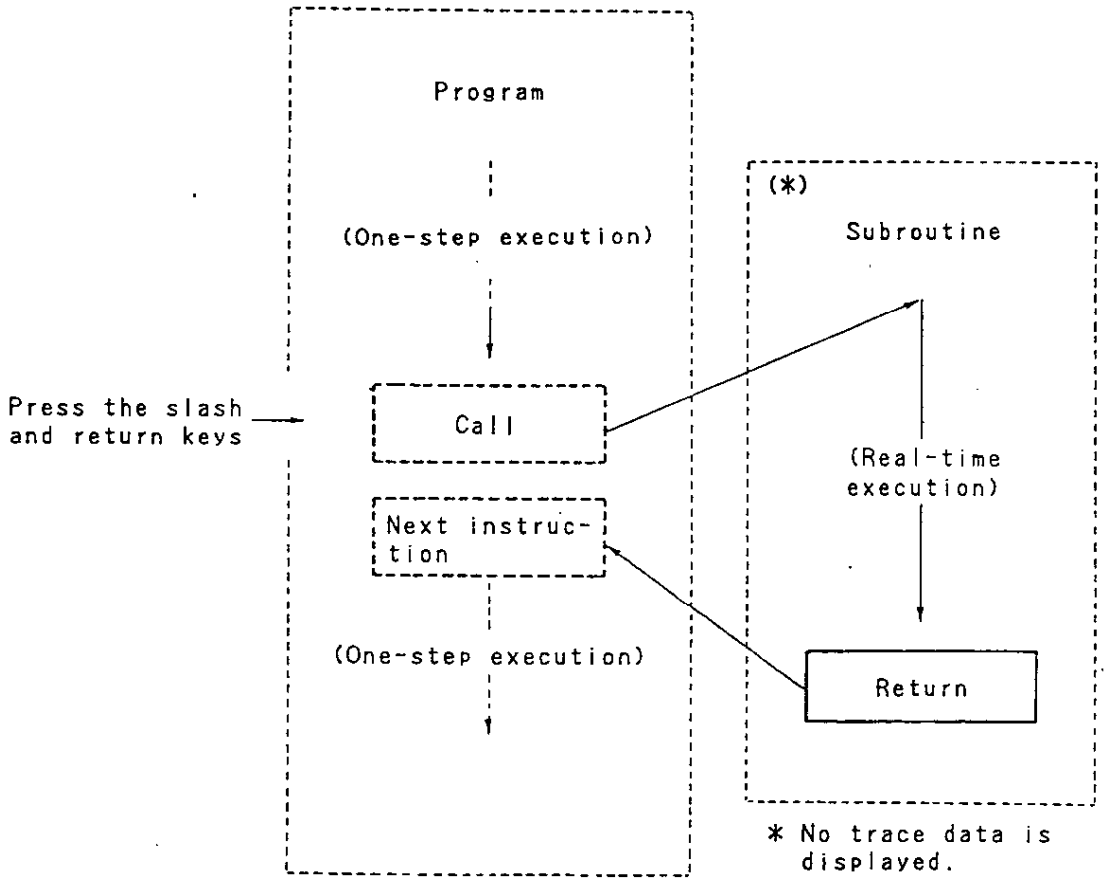
② Call instruction CALL, CALLA, CALLF, or CALLT appears. Pressing the / key and <cr> key enters the procedure mode.

③ The subroutine of the branch destination of the call instruction is executed in the procedure execution mode (real-time execution).

④ A return instruction coded in the subroutine returns to the calling execution program. Execution returns to the instruction next to the call instruction.

⑤ The instruction next to the call instruction and the following instructions are executed in the one-step execution mode.

Fig. 6-5 Concept of Procedure Execution



6.2.6 Break function

The break function stops emulation and tracing of the target program for the emulation device.

Break functions are divided into three major breaks as follows:

Event detection break
Manual break
Fail-safe break

The following shows the relationship between these break functions and emulation function.

	Event detection break	Manual break	Fail-safe break
Nonbreak real-time execution (RUN N command)	No	Yes	Yes
Real-time execution under break conditions (RUN B command)	Yes	Yes	Yes
One-step real-time execution (nonreal time) (RUN T command)	No	Yes	Yes

(1) Event detection break

The event detection break function stops the target program execution when a specified event condition is detected.

This type of break is valid only for real-time execution under break conditions (RUN B command).

The conditions of event detection must be set in the event mode register and event condition registers beforehand.

(i) Event mode register (BRM)

The BRM must be loaded with some of the following event condition register names:

BRA1, BRA2, BRA3, BRA4, BRS1, BRS2

The contents of the event condition registers selected in the BRM are used as the conditions of break. BRS is specified by default.

(ii) Event condition registers (BRA1 to BRA4, BRS1 and BRS2)

The event condition registers must be loaded with detailed conditions of event detection.

BRA1 to BRA4: Information related to data memory access is loaded as event detection conditions.

BRS1 and BRS2: Information related to program memory access is set as event detection conditions.

For how to set conditions of event detection, see Section 6.2.9.

(2) Manual break

The manual break function is provided for the user to force real-time emulation and trace operation to stop (forced break). The forced break function is valid to every execution function.

There are two manual breaks.

Forced break (STP)

The STP command stops the emulation device and tracer. A forced break by an STP command is valid only to the real-time execution function.

Forced break (RES)

The RES command stops the emulation device and tracer.

Note: The emulation device is reset at the same time when the target program is stopped.

(3) Fail-safe break

The fail-safe break function is provided by the IE-75001-R to force the emulation device to terminate when the device operates abnormally. The forced break function is valid to every execution function.

The following fail-safe break is provided:
Invalid access break

When the target program tries to access an area not present in the target device, including DMEM (data memory), an SPR (special register), a register area, and a stack area, an invalid access break occurs to force the program to terminate.

6.2.7 Trace function

In tracing the target program, the trace function writes information about the status of program memory, data memory, I/O ports, and external sense clip in trace memory in the IE-75001-R.

Data written in trace memory can be displayed on the screen with the TRD command.

Major functions for tracing and displaying trace data are as follows:

Trace operation

. Tracing in nonbreak real-time execution (RUN N command)

. Tracing in real-time execution under break conditions (RUN B command)

. Tracing in step-by-step execution (RUN T command)

Setting of trace conditions

. Specification of qualified trace conditions (TRX command)

. Specification of sectional trace conditions (TRY command)

. Specification of trace mode (TRM command)

Display of trace data and pointer manipulation

. Setting of trace data retrieval conditions (TRF command)

. Display of trace data (TRD command)

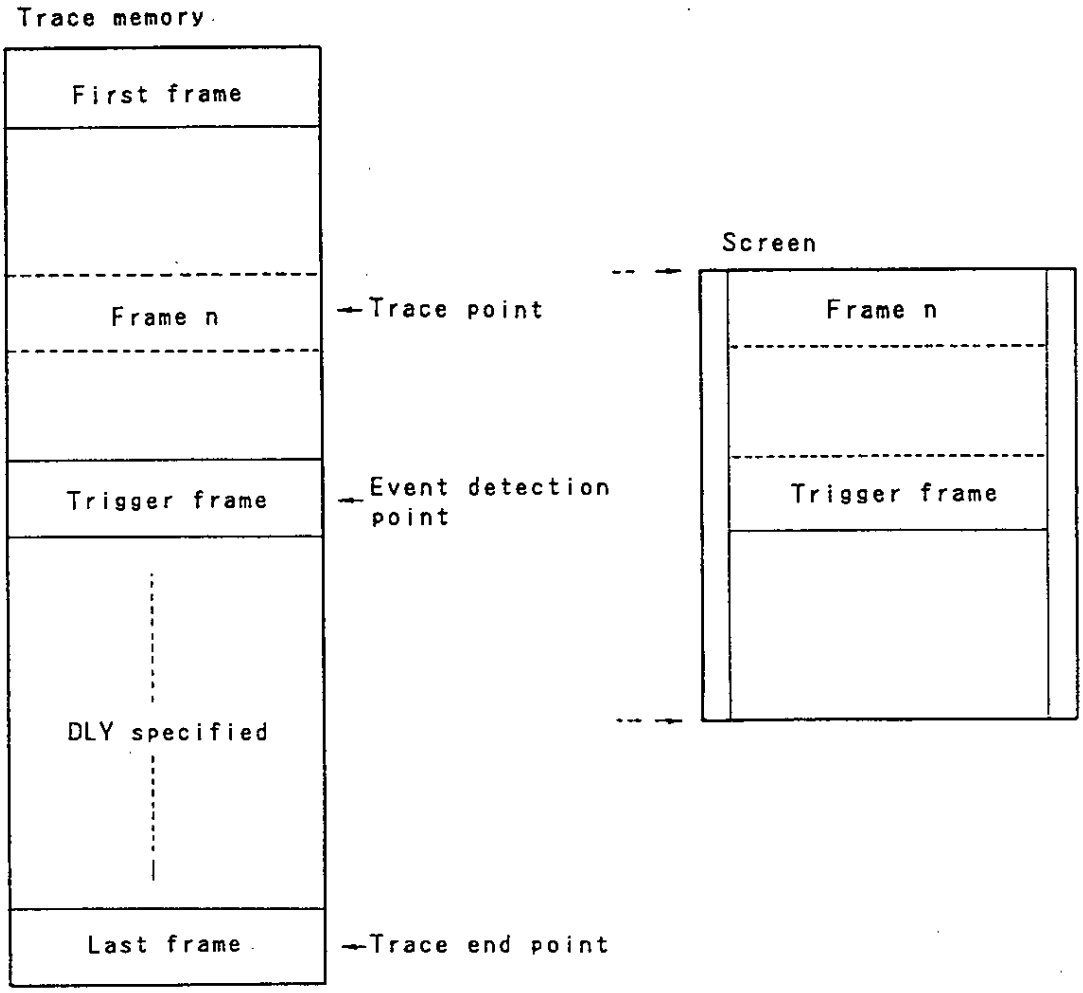
. Trace pointer manipulation (TRP command)

(1) Tracing and trace memory

The trace function records trace information starting from the first frame to the last frame. As the function has traced the last frame, it again returns to the first frame position for continuous recording.

Figure 6-6 shows a concept of trace in which a RUN B command and a DLY M command are specified.

Fig. 6-6 Concept of Trace



Remark: The DLY M command sets the trigger frame in the middle of trace memory.

(2) Trace operation

The operation of the tracer depends on the execution form.

(a) Operation in non-break real-time execution

When a RUN N command is entered, the tracer starts tracing. If an event condition specified with the BRM command is satisfied, the tracer terminates after it reaches the delay point specified with the DLY command.

(b) Operation in real-time execution under break conditions

When a RUN B command is entered, the tracer starts tracing. If an event condition specified with the BRM command is satisfied, the tracer stops after it reaches the delay point specified with the DLY command.

As the tracer stops, execution of the target program (the emulation device) is also stopped, and one-step execution mode is entered.

In the one-step execution mode, the following operations are allowed for tracing:

<cr> input: Executes the next instruction, then displays trace data on that instruction on the screen.

Note: One-step execution erases the trace data already present in trace memory.

/ <cr> input: Causes procedure execution.

ESC input: Terminates one-step execution mode.

(c) Operation in step-by-step execution

The RUN T command operates the tracer after one-step execution. In this case, trace information for only one step is recorded. The trace data in trace memory are erased every one-step execution.

(3) Trace condition setting function

Conditions of trace can be specified with some commands as explained below. If no condition is specified, unconditional tracing is performed, and trace information is recorded after every instruction in the target program is executed.

(a) Specifying qualified-trace condition (TRX command)

When an access to a specified program address or specified data memory location is made, tracing is started. Conditions to be specified must be loaded in the following event condition registers beforehand:

BRA1 to BRA4: Data memory access conditions
BRS1 and BRS2: Program addresses

For how to load the event condition registers, see Sections 8.4.3 and 8.4.6.

(b) Specifying sectional trace (TRY command)

When a specified enable condition is satisfied, tracing starts, and when a specified disable condition is satisfied, tracing terminates. This specification thus sets the range to be traced.

TRY E [BR?] [BR?] [BR?] [BR?] [BR?] [BR?] :
Sets an enable condition.

TRY D [BR?] [BR?] [BR?] [BR?] [BR?] [BR?] :
Sets a disable condition.

The enable and disable conditions must be loaded in the event condition registers beforehand.

BR?:BRA1 to BRA4: Data memory access conditions
BRS1 and BRS2: Program addresses

(c) Specifying trace mode (TRM command)

A trace mode, qualified trace, sectional trace, or unconditional trace, is specified. The trace mode specified with the TRM command is used as the final trace mode.
TRM TRX: Qualified trace
TRM TRY: Sectional trace
TRM ALL: Unconditional trace

(4) Functions for displaying trace data and pointer manipulation

Display of trace data and retrieval conditions used for displaying data can be specified with particular commands

(a) Setting trace data retrieval conditions (TRF command)

The conditions of trace data retrieval can be specified. This command is valid when one of the following display trace data commands is specified:

TRD[ALL] $F command
TRD[ALL] $Q command

The all or some items listed in Table 6-3 may be specified as retrieval conditions:

Table 6-3 Trace Items and Valid Range

Item	Description	Range
PA	Program memory address	0-FFFFH
PD	Program memory data Contents of a location indicated by PA (instruction code)	0-FFH
MA	Data memory address	0-FFFH
MD	Data memory data Contents of a location indicated by MA (data)	0-FFH
MRW	Data memory access status MRD: Read MWR: Write MRW: Read-modify-write NC: All read and write	MRD: Read MWR: Write MRW: Read-modify-write NC: All read and write
Pn	I/O port data	0-FH
EXT	External sense clip data	0-FFH

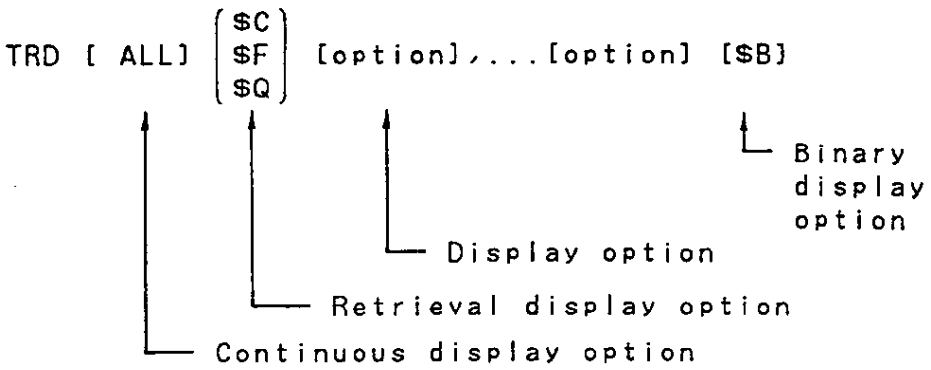
(b) Displaying trace data (TRD command)

Trace data stored in trace memory are displayed on the screen. Table 6-4 lists the items that can be displayed.

Table 6-4 Trace Data to Be Displayed

Item	Description	Value, range
Frame	Frame No. No. of trace memory frame	0-7FFH
PA	Program memory address	0-FFFFH
PD	Program memory data Contents of a location indicated by PA (instruction code)	Up to 3 bytes
MA	Data memory address	0-FFFH
MD	Data memory data Contents of a location indicated by MA (data)	0-FH (4 bits) 0-FFH (8 bits)
MRW	Data memory access status MRD: Read MWR: Write MRW: Read-modify-write	——
Pn	I/O port data	0-FH
EXT	External sense clip data	0-FFH
Label	Label Label resulting from symbolic translation of PA	8 characters
Mnemonic	Mnemonic code Code resulting from disassembly of PD (instruction code)	24 characters

Particular trace data items can be selected and displayed in a desired form by specifying appropriate options.



(i) Displaying all data (TRD ALL command)

The TRD ALL command displays all trace data, then terminates.

(ii) Displaying one-page trace data (TRD command)

The TRD command displays one screen page (12 lines) of trace data starting from the current frame, then enters the menu mode. In the menu mode, the following commands become available for display operation:

Table 6-5 Valid Trace Data Range

Command	Function
L	Displays trace data with the last trace frame used as the base.
F	Displays trace data with the first trace frame used as the base.
+, <cr>	Displays the next page to the current page.
-	Displays the preceding page to the current page.
T	Displays trace data with the trigger frame (event condition detection frame) used as the base.
.	Terminates the TRD command.

Note: For the commands other than the T and commands, functions are qualified by the specification of the retrieval display option ($C, $Q, or $F). See Chapter 8 for details.

(iii) Specifying retrieval display condition

When the retrieval display option $Q or $F is specified in the TRD command, the retrieval conditions specified with the TRF command become valid.

(c) Manipulating the trace pointer (TRP command)

The trace pointer manipulation function enables the trace pointer to point to a specified location. There are four pointing methods.

Specifying displacement (TRP word command)

The value of the trace pointer is changed by a specified number. A valid specification ranges from ±1 to ±7FF in hexadecimal.

Specifying the first frame (TRP F command)

The trace pointer is set to point to the first frame (oldest frame).

Specifying the last frame (TRP L command)

The trace pointer is set to point to the last frame (newest frame).

Specifying trigger frame (TRP T command)

The trace pointer is set to point to the trigger frame. If no trigger frame is present, this command is regarded insignificant. After emulation, the trace pointer points to the trigger frame. If there is no trigger frame, the pointer points to the first frame (oldest frame).

Specifying absolute address of trace frame

The trace pointer is moved to the specified absolute address. Specifiable range is only the currently traced range.

6.2.8 Check function

The check function interrupts execution when it encounters a specified checkpoint during real-time execution, and outputs specified information as check data to trace memory.

Upon completion of information output to trace memory, real-time execution resumes.

Outputting the contents of general registers as check data (CHK BR?...BR? REG command)

Outputting the contents of special registers as check data (CHK BR?...BR? SPR command)

Outputting the contents of data memory locations as check data (CHK BR?...BR? addr command)

(1) Outputting the contents of general registers as check data (CHK BR?...BR? REG command)

As the content of a specified event condition register (checkpoint) is detected, the contents of the general registers are written as check data in trace memory.

(a) Selection of checkpoint

The following event condition registers may be specified as checkpoints:

BRA1, BRA2, BRA3, BRA4:
Data memory access point
BRS1, BRS2: Program memory access point
OFF: Stops the check function.

Note: Checkpoints are set with the BRA1 to BRA4, BRS1 and BRS2 commands.

(b) Specification of check data

A register representative name (REG) is used for specification. The contents of the following registers are output as check data:

XA, HL, DE, BC, XA', HL', DE', BC':
Register pair

SP (including SBS), PC, RBS, MBS:
Control register

CY, RBE, MBE, IST0, IST1: Flag

(2) Outputting the contents of special registers as check data (CHK BR?...BR? SPR...SPR command)
As the contents of a specified event condition register (checkpoint) are detected, the contents of specified special registers are written as check data in trace memory.

(a) Selection of checkpoint

Same as Item (1) above.

(b) Specification of check data

Up to five special registers are specified. The special registers vary according to the devices.

(3) Outputting the contents of data memory locations as check data (CHK BR?...BR? addr...addr)
As the contents of a specified event condition register (checkpoint) are detected, the contents of specified data memory locations are written as check data in trace memory.

(a) Selection of checkpoint

Same as Item (1).

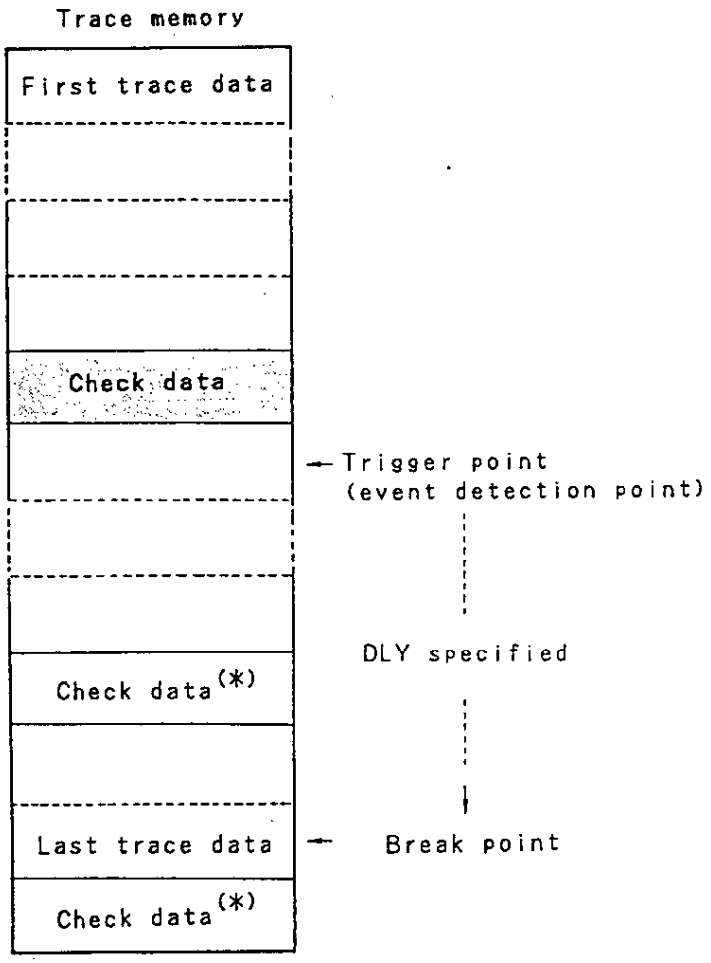
(b) Specification of check data

Up to five data memory addresses are specified.

(4) Check data output timing

During DLY (from detection of an event to occurrence of a break), no check data are output. (See Figure 6-7.)

Fig. 6-7 Trace Memory Holding Check Data



* This part of check data is not output.

6.2.9 Event setting and detection function

Event setting and detection functions are provided to set conditions to stop target program execution by the emulated device or trace operation by the tracer, and to detect an event under the set conditions.

There are three event condition setting and detection functions.

Event detection condition setting function

Setting data memory event conditions (BRA1 to BRA4 commands)
Setting program memory event conditions (BRS1 to BRS2 commands)

Event detection condition integration function

Integrating event conditions (BRM command)

Event detection delay function

Setting event detection points (DLY command)

(1) Event detection condition setting function

The event detection condition setting function loads conditions used for stopping target program execution and trace operation in event condition registers. The event detection conditions set with this setting function become valid only when they are set in the event mode register by the event detection condition integration function. For details, see Item (2) in this section.

There are two functions to set event detection conditions.

(a) Data memory event condition setting function (BRA1 to BRA4 commands)

An access to a specified data memory location made by the target program and data input to the external sense clip can be loaded as event detection conditions in the data memory event condition registers.

(i) Data memory event condition registers

There are four data memory condition registers. Up to four conditions can be set in these registers with the BRA1 to BRA4 commands, respectively.

(ii) Event condition

The following statuses can be set as event detection conditions. (See Section 8.4.3.)

MA: Data memory address
V: Data value
C: Access status
E: External sense clip data

(b) Program memory event condition setting function (BRS1 and BRS2 commands)

This function enables conditions of event detection to be set in the program memory event condition registers. Possible events are: execution of an instruction at a specified program memory address in the target program (that is, the instruction is accessed by the emulated device), and input of data to the external sense clip at the execution of the instruction.

(i) Program memory event condition registers

There are two program memory event condition registers, BRS1 and BRS2. Up to two conditions can be set in these registers with the BRS1 and BRS2 commands, respectively.

(ii) Event conditions

The following items can be set as event detection conditions:

PA: Program memory address
E: External sense clip data

(2) Event detection condition integration function (BRM command)

The event detection condition integration function integrates the event detection conditions set by the event detection condition setting function into the event mode register to make them valid.

BRS1 is specified by default.

(i) Event mode register

Only one event mode register, BRM1, is provided.

(ii) Event condition registers

There are six event condition registers that can be registered in the event mode register.

Data memory event condition registers:
BRA1, BRA2, BRA3, BRA4

Program memory event condition registers:
BRS1, BRS2

(3) Event detection point setting function

The event detection point setting function specifies the delay of execution (adjustment by delay) for the event detection point determined by the set event mode (BRM) command.

One of the following three is specified to place the event detection point (trigger point) in trace memory.

L is specified by default.

(i) Specifying F (First)

The event detection point is placed in the first frame of trace memory.

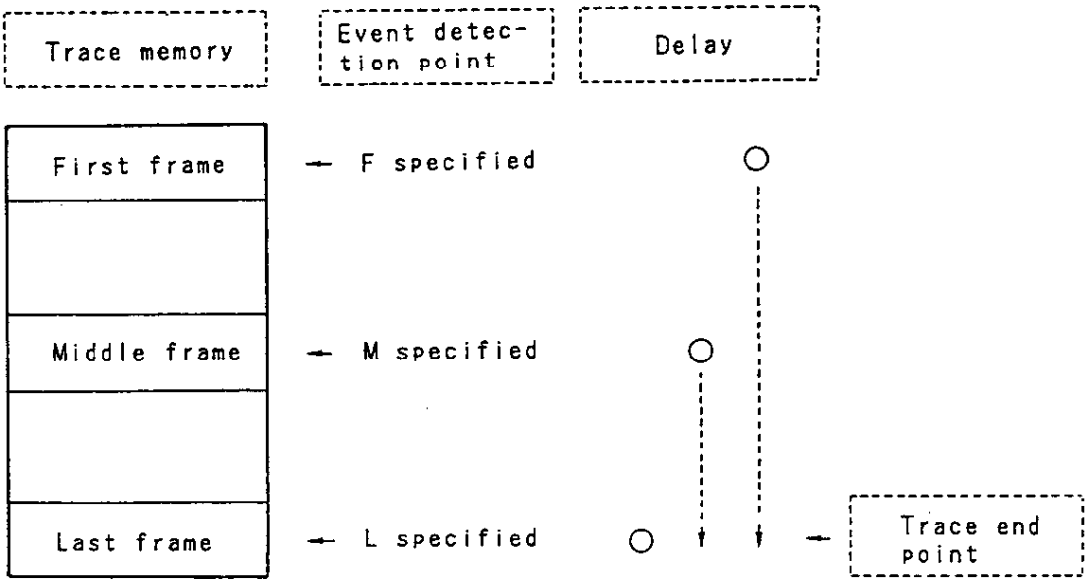
(ii) Specifying M (Middle)

The event detection point is placed in the middle of trace memory.

(iii) Specifying L (Last)

The event detection point is placed in the last frame of trace memory.

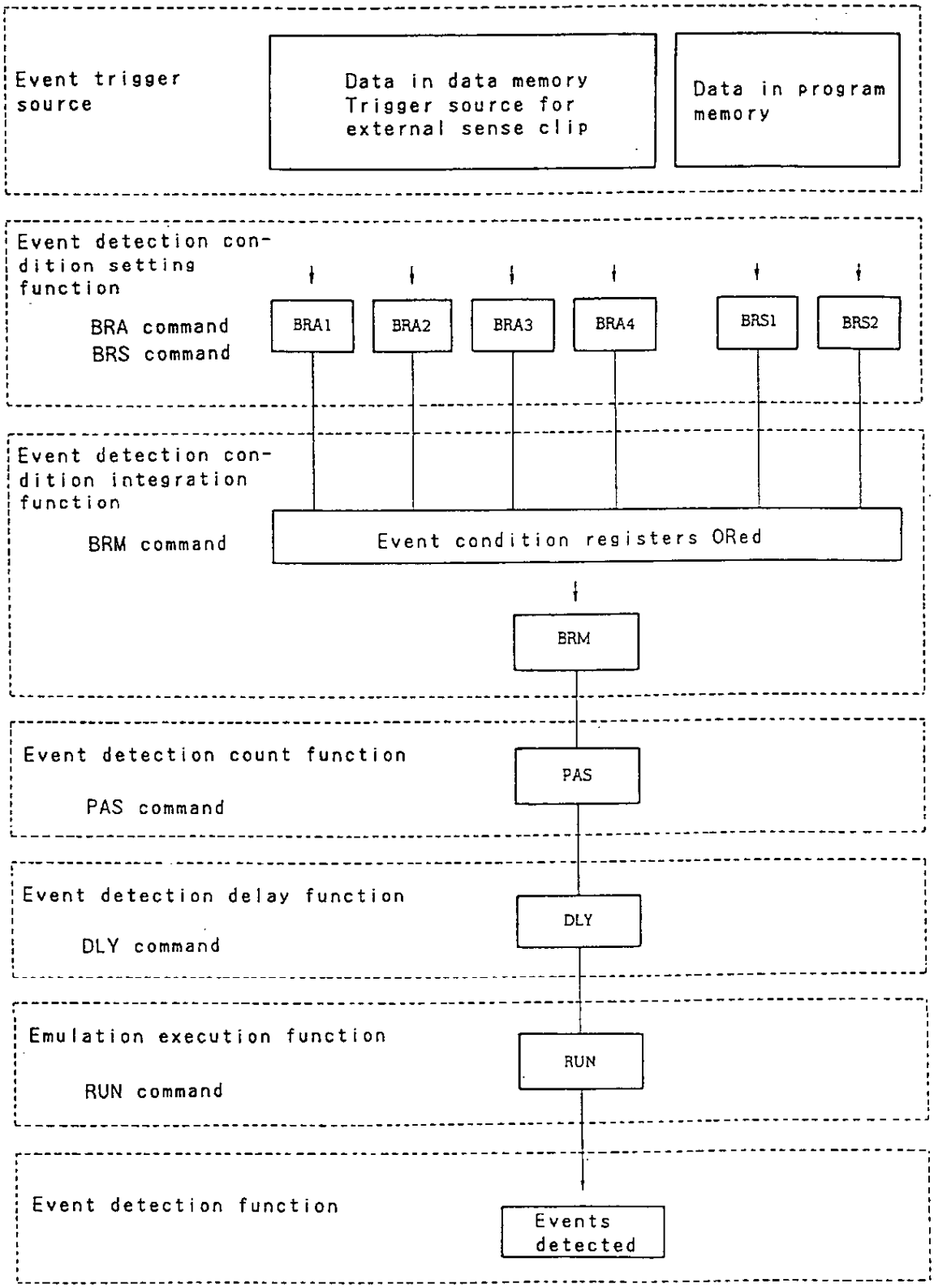
Fig. 6-8 Setting Event Detection Point



(4) Concept of event detection

Figure 6-9 shows the concept of the process from setting of event conditions to event detection.

Fig. 6-9 Concept of Event Detection



6.2.10 Register manipulation function

The register manipulation function displays and changes the contents of general registers and special registers.

The major functions are:

Manipulating general registers

- Display register contents (REG D command)
- Change register contents (REG C command)

Manipulating special registers

- Display register contents (SPR D command)
- Change register contents (SPR C command)

(1) Manipulate general register function

The manipulate general register function displays and changes the contents of control registers and general registers.

Control registers: PC, SP, PSW, RBS, MBS

General registers: XA, HL, DE, BC, XA', HL', DE', BC', X, A, H, L, D, E, B, C

The contents of the PSW are displayed and changed with the following PSW flag names. The registers vary depending on the target devices.

PSW flags: CY, RBE, MBE, IST1, IST0

(a) Displaying register contents (REG D command)

The contents of all registers in the current register bank or all register banks of the target device are displayed.

Current register bank: Register bank specified by RBS (register bank selector)

(b) Changing register contents (REG C command)

The contents of registers in the current register are changed.

(2) Manipulate special register function

The manipulate special register function displays and changes the contents of the I/O area of the target device (called special registers).

Registers to be manipulated are specified with a group name or register name.

Group name: The special registers in the 75X series are assigned to addresses 0F80H to 0FFFH in data memory.

In the SPR command, a group name can be specified with high-order eight bits of addresses to which special registers are assigned.

A group name varies depending on the target devices.

Register name: Special register names vary depending on the target devices.

(a) Displaying register contents (SPR D command)

The contents of specified registers, all registers in a group, or all special registers are displayed.

(b) Changing register contents (SPR C command)

The contents of specified registers, all registers in a group, or all special registers are changed.

6.2.11 Memory manipulation function

The memory manipulation function displays or changes the contents of data memory or program memory using mnemonic code or hexadecimal code.

Memory manipulation function using mnemonic code

- Displaying program memory contents in mnemonic code (DAS command)
- Changing program memory contents in mnemonic code (ASM command)

Memory manipulation function using hexadecimal code

- Manipulating program memory (MEM command)
- Manipulating data memory (RAM command)

(1) Memory manipulation function using mnemonic code

This function displays or changes a target program (object code) in program memory using mnemonic code.

(a) Displaying program memory contents in mnemonic code (DAS command)

Hexadecimal object code stored in program memory is displayed in mnemonic code.

(b) Changing program memory contents in mnemonic code (ASM command)

When mnemonic code is entered, modification or addition is enabled to the target program in program memory.

(2) Memory manipulation function using hexadecimal code

This function displays or changes the contents of program memory or data memory using hexadecimal code.

(a) Manipulating program memory

The following functions are provided for manipulating program memory:

Change memory contents	(MEM C command)
Display memory contents	(MEM D command)
Test memory contents	(MEM E command)
Initialize memory contents	(MEM F command)
Search memory contents	(MEM G command)
Copy memory contents	(MEM M command)
Exchange memory contents	(MEM X command)
Compare memory contents	(MEM V command)

(b) Manipulating data memory

The following functions are provided for manipulating data memory:

Change memory contents	(RAM C command)
Display memory contents	(RAM D command)
Test memory contents	(RAM E command)
Initialize memory contents	(RAM F command)
Search memory contents	(RAM G command)
Copy memory contents	(RAM M command)
Exchange memory contents	(RAM X command)
Compare memory contents	(RAM V command)
Load file to data memory	(RAM L command)
Save data memory contents to file	(RAM S command)

6.2.12 Save function

The save function saves the target program on the IE-75001-R in a device connected to a channel of the IE-75001-R.

For the host machine, debugging environment information can also be saved.

The devices that can be connected to the IE-75001-R are listed below.

Device connected	Channel in IE-75001-R	Application
Host machine (PC-9800 series, IBM PC series)	CH1 (for I/O)	Standard downloading Standard uploading
PROM programmer (PG-1000, 1500)	CH2 (for I/O)	Standard downloading Standard uploading

(1) Save in host machine

Object code and debugging environment information in the IE-75001-R are saved in files on a disk device connected to the host machine.

Table 6-6 lists the files with their brief descriptions, and Table 6-7 lists the commands available for the save function.

Table 6-6 Files Saved in the Host Machine

File	Description
Object file (file name: xxxxxxxx.HEX)	Contains the object code of the target program (in Intel hexadecimal format).
Symbol file (file name: xxxxxxxx.SYM)	Contains symbols defined by the target for the user program.
Debugging environment file (file name: xxxxxxxx.DBG)	Contains information on the debugging environment. This file can hold the following commands for environment setup. BRA1, BRA2, BRA3, BRA4, BRK, BRM, BRS1, BRS2, CHK, CLK, DLY, MOD, OUT, PAS, PGM, REG, STS, TRF, TRM, TRX, TRY

Notes 1. The symbol file may be omitted. When it is omitted, however, no symbols can be processed. Append symbols are loaded with the SYM L command.

2. The debugging environment file may be omitted. When it is omitted, however, an environment can not be set up.

Table 6-7 Types and Functions of Save Commands

File	Function
SAV file	Saves object code and debugging environment information in the object file and debugging environment file, respectively.
SAV file C	Saves only object code in the object file.
SAV file D	Saves only debugging environment information in the debugging environment file.

(2) Save to the PROM programmer

Object code in IE-75001-R memory is saved in a PROM programmer (PG series) connected to channel 2 (CH2).

6.2.13 System termination function

The system termination function terminates the IE-75001-R system, and returns control to the OS. When terminating the IE-75001-R system, be sure to use the EXT command.

6.2.14 Other functions

In addition to the functions explained in the previous sections, other functions are provided. This section briefly explains these functions on an application basis. For details, see Chapter 8.

(1) Symbol manipulation functions

Symbols used in the 75X series can be added, deleted, and displayed. The following functions are provided for manipulating symbols:

Define append symbol	(SYM A command)
Change append symbol value	(SYM C command)
Display all symbols	(SYM D command)
(Append, public, and local symbols)	
Delete append symbol	(SYM E command)
Delete all symbols	(SYM K command)
Load append symbol	(SYM L command)
Save append symbol	(SYM S command)
Specify current module name	(SYM M command)

(2) Command file manipulation functions

Various manipulations including open and close operation, and I/O, are performed for the command file.

(a) Redirect input (STR d:file command)

The redirect input function accepts commands and data from a file on a disk device connected to the host machine.

Input file

- File created with the create command file command (COM)
- Commands and data created by the editor

Start and end of input operation

- Stop input temporarily: ^L (CTRL + L)
- Restart input: ^L (Available only when the system is ready for command input)
- Terminate input: ^K (CTRL + K)

(b) Redirect output (LST command)

The redirect output function opens or closes a device specified for output of command execution results. There are four types of redirect output.

Open a file output device and a file
(LST d:file command)

Open a list output device
(LST LST:command)

Close a file or list output device
(LST CON:command)

Start and end of output

- Start output
- Stop output

Pressing ^P (CTRL + P) alternates start and stop operations.

(c) Create command file function (COM command)

The create command file function opens a file or list output device to accept commands and data output after the input of a COM command, and closes the open device.

There four types of create command file functions.

Open a command file (COM d:file command)
Open a list device (COM LST:command)
Close a command file (COM CON:command)

Start and end of output

- Start output — Pressing ^O (CTRL + O) alternates
- Stop output — start and stop operations.

(d) Display of command history (HIS command)

This function displays 20 lines of the current command history on the screen, and re-executes commands in the current command history.

Re-execute commands starting on a specified line: !n <cr> (n: line number)

Re-execute the command on the line immediately before the current line: !! <cr>

(3) Other functions

(a) Run DOS function (DOS command)

The run DOS function causes the IE-75000-R control program to pass control to the OS. After the DOS command is executed, the commands of the OS become available.

To return to the IE-75000-R control program from the OS, the EXIT command of the OS is used.

(b) Help function (HLP command)

The help command shows the user information about a command, including the input format of the command, explanation of input parameters, and the use of the command.

(c) Display directory function (DIR command)

The display directory function displays the directory of the files on a specified disk device.

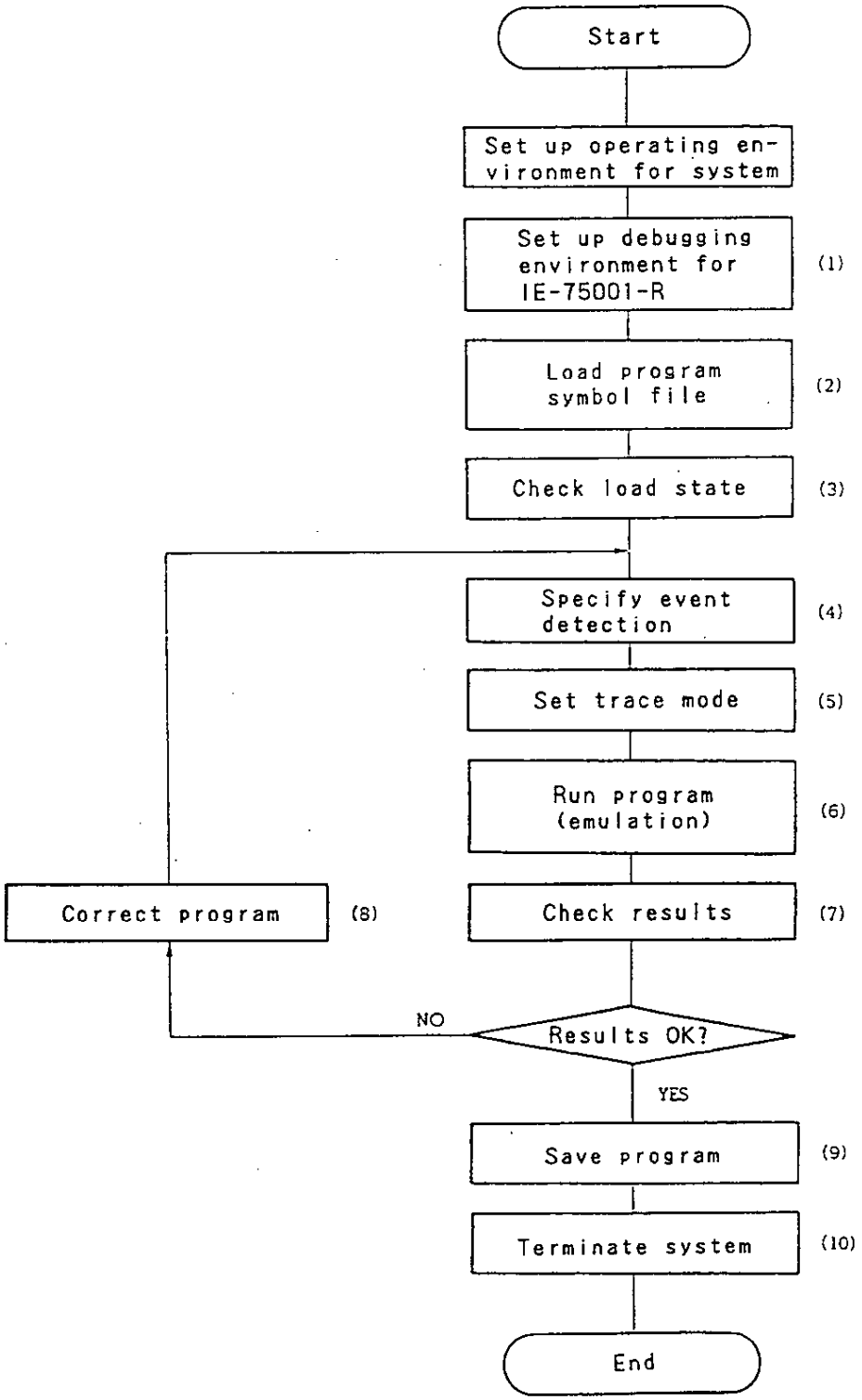
(d) Math command (MAT command)

The math command accepts a conditional expression for operation, then displays the operation results on the screen.

6.3 Basic Debugging Procedure

This section explains the basic procedure for debugging in the sequence of the flowchart shown below.

Fig. 6-10 Flowchart of Basic Debugging Procedure



(1) Setting up debugging environment for the IE-75001-R

When the system has been started, it enters the break mode and becomes ready for command input. Before debugging, a debugging environment must be set up with the following commands:

CLK:	Selects a clock source.
OUT:	Specifies the output of the event trigger signal.
LOD file D:	Loads debugging environment information stored in a file on the host machine. (When a program is loaded, the debugging environment information can be loaded together.)

(2) Loading a program and symbols

A program to be debugged and the symbols are loaded.

(a) The program is loaded with one of the following commands:

LOD file C:	Loads the object file on the host machine.
PGM:	Loads the program on the PROM programmer (PG series).

(b) The symbols are loaded with the following commands:

LOD file S:	Loads the symbols stored in the symbol file on the host machine.
SYM L:	Loads append symbols.

(3) Checking the load state

When debugging environment information, the program, and symbols have been loaded, the load state is checked with the following commands:

DAS: Checks the load state by disassembling the program.
RAM: Checks the data memory contents.
SYM D: Checks the symbols.
VRY: Compares the file contents with the contents of memory immediately after the program is loaded.

(4) Specifying event detection conditions

On the initial phase of debugging, an event is detected for each set point, and the program is debugged on a case-by-case basis.

Event detection conditions are set with the following commands:

BRA: Sets detection conditions about data memory.
BRS: Sets detection conditions about program memory.
PAS: Sets the pass counter.
BRM: Determines whether the detection conditions set with BRA and BRS are made valid when the program is executed.
DLY: Delays an event detection.

(5) Setting a trace mode

Output of the trace information required for debugging to the trace area (memory) is specified. The output data are used for checking results of execution.

The trace mode is specified with the following commands:

TRX: Sets conditions of qualified trace.
TRY: Sets conditions of sectional trace.
TRM ALL: Sets the unconditional trace mode.
TRM TRX: Sets the qualified-trace mode.
TRM TRY: Sets the sectional trace mode.

(6) Running the program

The program to be debugged is run, or emulated. One of the following execution modes is used.

RUN N: Nonbreak real-time execution
RUN B: Real-time execution under break conditions
RUN T: Step-by-step execution

(7) Checking results of execution

When the target device and tracer are stopped because an event is detected, the results of execution are checked with the following commands:

REG D: Checks the contents of general registers.
SPR D: Checks the contents of special registers.
MEM D: Checks the contents of program memory.
RAM D: Checks the contents of data memory.
TRD: Checks the contents of trace data.
TRF: Sets trace data retrieval conditions.
TRP: Manipulates the trace pointer.

(8) Correcting the program

If program correction is required, the following commands are used for modifying program and data contents.

If correction extends to large part of the program, it is more effective to modify the source program.

ASM: Corrects the program and data in mnemonic code.
MEM C: Corrects program memory contents in hexadecimal code.
RAM C: Corrects data memory contents in hexadecimal code.

(9) Saving the program, symbols, and debugging environment information

When the program has been debugged, the program, symbols, and debugging environment information are saved.

The program and debugging environment information can be saved in files on the host machine at a time. The following, however, describes how to save these data respectively.

(a) The program is saved with one of the following commands:

SAV file C: Saves the program in the object file on the host machine.
PGM: Saves the program in the PROM programmer (PG series).

(b) Append symbols are saved with the following command:

SYM S: Saves append symbols in the append symbol file on the host machine.

(c) The debugging environment information is saved with the following command:

SAV file D: Saves the debugging environment information in the debugging environment information file on the host machine.

(10) Terminating the system

Debugging work is completed and the system is terminated. Control is then passed to the OS, and the host machine is logically disconnected from the IE-75001-R.

To debug another program continuously, start debugging with step (1) (setting up a debugging environment).

The following command is used to terminate the system:

EXT: Terminating IE-75000-R control program

6.4 Examples of Basic Functions

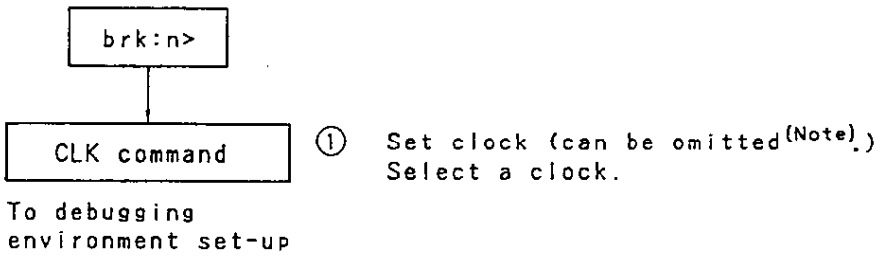
This section explains how to use the basic functions from the initialization of the IE-75001-R to the system termination sequentially.

Users who use the IE-75001-R for the first time should operate the emulator with referring to the explanations and examples in this section for understanding of overall operations and the basic functions.

6.4.1 Initialization of the IE-75001-R

When the system has been started, the break mode is entered (brk:n>). The IE-75001-R is then initialized.

(a) IE-75001-R initialization procedure



Note: When omitted, the IE-75001-R is emulated by internal clock (4.19 MHz).

(b) Sample operation

When the system has been started, the IE-75001-R is initialized.

```
brk:0>CLK <cr>                    ①
  IE                              ②
brk:0>CLK U <cr>                  ③
brk:0>CLK <cr>                    ④
  User
brk:0>
```

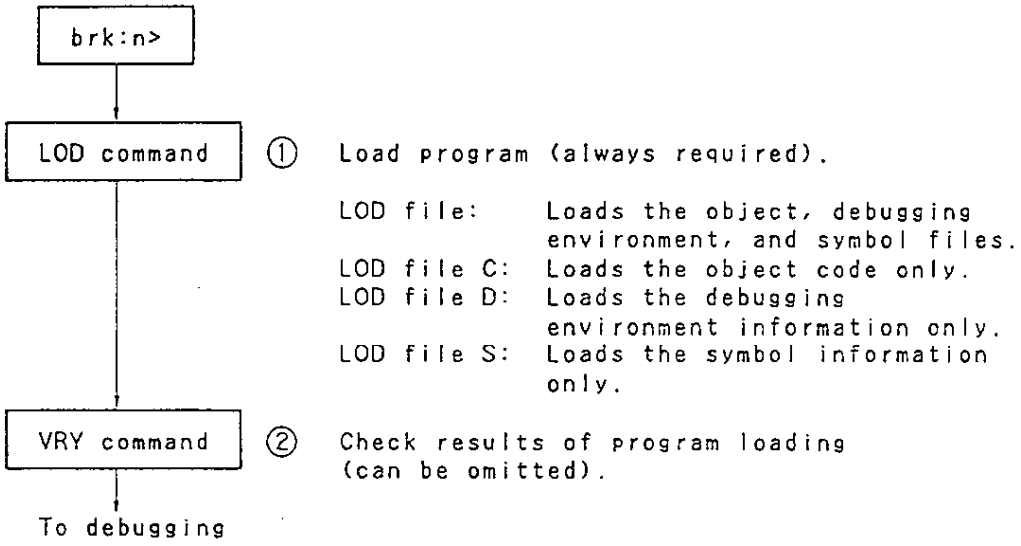
① Check the clock currently selected.
② The current clock is indicated.
③ Change the current clock to the target system clock(Note).
④ Check result of change.

Note: When changing the clock to the target system clock, the clock of the emulation board should also be changed.

6.4.2 Debugging environment setup

When the IE-75001-R has been initialized, setup of a debugging environment, including loading of the target program, is performed.

(a) Debugging environment setup procedure



(b) Sample operation

The object file, debugging environment file, and symbol file on the host machine are loaded into channel 1 (CH1) or 2 (CH2) of the IE-75001-R at a time.

```
brk:0>
brk:0>
brk:0>LOD B:SAMPLE <cr>                                    ①
  Debug condition load   (Y/N)?  Y <cr>                    ②
  Debug condition load complete                            ③
  object load complete                                     ④
  symbol table loading
  HEIKIN          load complete
  TASU            load complete                            ⑤
  WARU            load complete

brk:0>VRY B:SAMPLE.HEX <cr>                                ⑥
  object verify complete                                   ⑦
brk:0>■
```

① The batched loading of the file is specified. (No extension is required for the file name.)

② The loading of the debugging environment is determined and specified.

③ The debugging environment information has been loaded.

④ The object has been loaded.

⑤ The symbols have been loaded.

⑥ The object file is compared with the result of the loading.

⑦ The comparison is terminated normally.

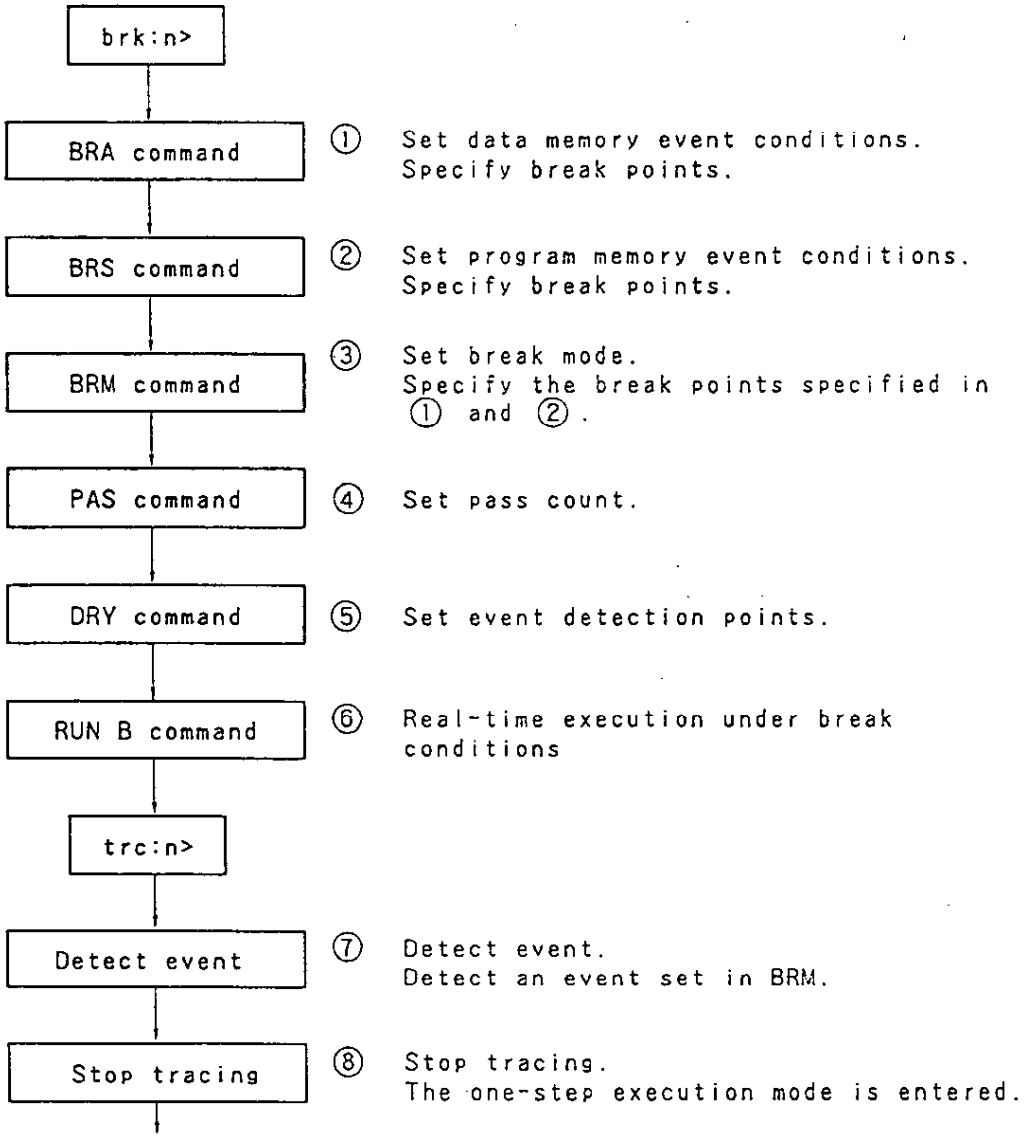
6.4.3 Target program execution

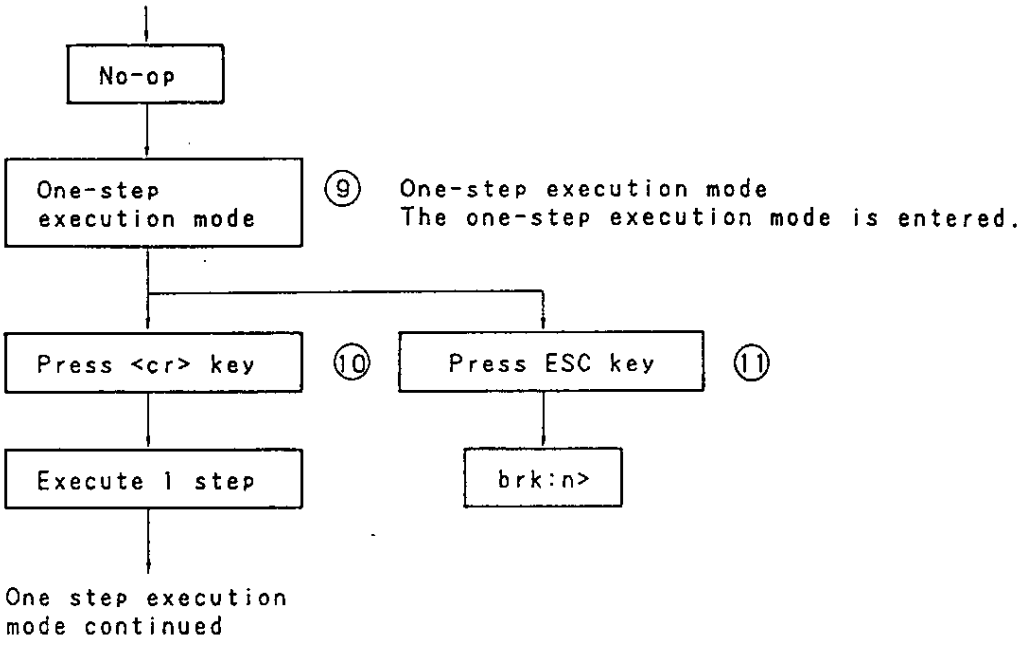
When initialization and debugging environment setup have been completed, it is time to debug the target program.

In this section, how the target program is executed is explained.

(1) Example for executing the target program with unconditional trace specification

(a) Procedure





⑩ One-step execution

When the <cr> key is pressed, one step is executed. The one-step execution mode then continues.

In this mode, trace information for each instruction is displayed on the screen.

Note: One-step execution erases the information previously written in trace memory.

⑪ The <ESC> key causes the break mode.

(b) Sample operation

Break specification is made so that a break occurs when data memory address 0100H or 0101H is accessed or when the instruction at program memory address 2000H is executed.

```
brk:0>BRA 1 MA=0100 V=00001111Y C=W8 <cr>                          ①
brk:0>BRA 2 MA=0101 V=0000XXXXY C=R8 <cr>                          ②
brk:0>BRS 1 P 2000 <cr>                                            ③
brk:0>BRM BRA1 BRA2 BRS1 <cr>                                      ④
brk:0>DLY M <cr>                                                   ⑤
brk:0>RUN B 100 <cr>                                               ⑥
Emulation start at 100                                             ⑦
trc:0>■                                                            ⑧
BRS1 break terminated                                              ⑨
XA HL DE BC XA' HL' DE' BC' RBS MBS RBE CY IST0 SP   PC        } (*)
00 00 00 00 00  00  00  00   0   0   0  0   0   000 0000      }
One step emulation standby                                         ⑩
```

* Registers not present in the target device are indicated with --.

① Set the following in BRA1 as an event detection condition:
Data memory address: 0100H (hexadecimal)
Data memory content (value): 00001111Y (binary)
Memory access condition:
Valid bits = 8 bits
I/O mode = write, or read-modify-write

② Set the following in BRA2 as an event detection condition:

Data memory address: 0101H (hexadecimal)

Data memory content (value):

0000XXXXY (binary mask specification)

Memory access condition:

Valid bits = 8 bits

I/O mode = write, or read-modify-write

Remark: X = undefined

Y = binary representation

③ Set the following in BRS1 as an event detection condition:

Program memory address: 2000H (hexadecimal)

④ Set the contents of the following event detection conditions in BRM:

BRA1, BRA2, BRS1

⑤ Set the event detection point in the middle frame in trace memory.

⑥ Specify program address 100H as start address of emulation.

⑦ Emulation has started at program address 100H.

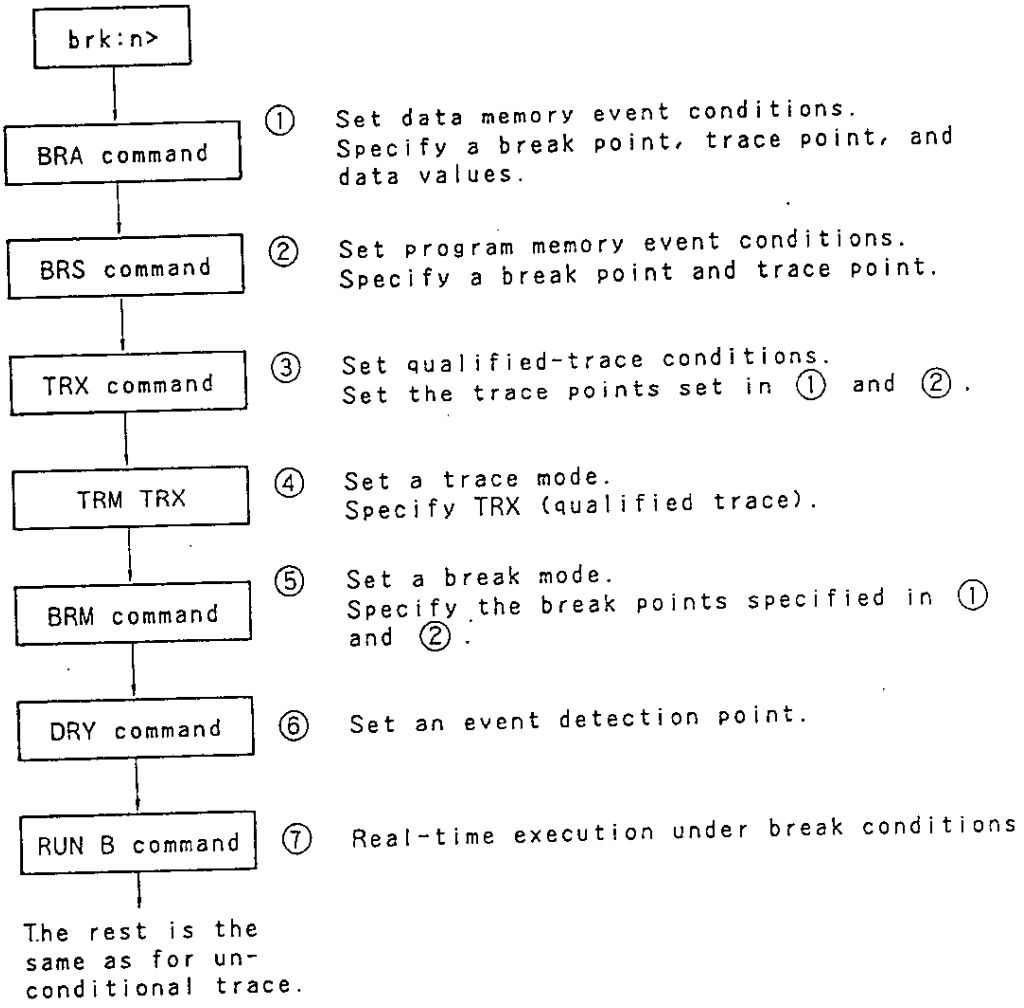
⑧ The system is ready for command input in the trace mode.

⑨ A break occurred by the event trigger source in BRS1 (program address 2000H).

⑩ The one-step execution mode has been entered.

(2) Example for executing the target program with qualified-trace specification

(a) Procedure



(b) Sample operation

In the following example, trace data are collected only when data memory address 0100H or 0101H is accessed or when the instruction at program memory address 1000H is executed, and a break occurs when the instruction at address 3000H is executed.

```
brk:0>BRA 1 MA=0100 V=00001111Y C=W8 <cr>                          ①
brk:0>BRA 2 MA=0101 V=0000XXXXY C=R8 <cr>                          ②
brk:0>BRS 1 P 1000 <cr>                                            ③
brk:0>BRS 2 3000 <cr>                                              ④
brk:0>TRX BRA1 BRA2 BRS1 <cr>                                      ⑤
brk:0>TRM TRX <cr>                                                 ⑥
brk:0>BRM BRS2 <cr>                                                ⑦
brk:0>DLY M <cr>                                                   ⑧
brk:0>RUN B 100 <cr>                                               ⑨
Emulation start at 100                                             ⑩
trc:0>■                                                            ⑪
BRS2 break terminated                                              ⑫
XA HL DE BC XA' HL' DE' BC' RBS MBS RBE CY ISTO SP   PC        }(*)
00 00 00 00 00  00  00  00   0   0   0  0   0   000 0000
One step emulation standby                                         ⑬
```

* Registers not present in the target device are indicated with --.

① Set the following in BRA1 as an event detection condition:

Data memory address: 0100H (hexadecimal)

Data memory content (value): 00001111Y (binary)

Data access condition:

Valid bits = 8 bits

I/O mode = write, or read-modify-write

Remark: Y: binary representation

② Set the following in BRA2 as an event detection condition:
Data memory address: 0101H (hexadecimal)
Data memory content (value):
0000XXXXY (binary mask specification)
Memory access condition:
Valid bits = 8 bits
I/O mode = read, or read-modify-write
Remark: X: undefined data
Y: binary representation

③ Set the following in BRS1 as an event detection condition:
Program memory address: 1000H (hexadecimal)

④ Set the following in BRS2 as an event detection condition:
Program memory address: 3000H (hexadecimal)

⑤ Set the following in TRX as trace points for qualified trace:
BRA1, BRA2, BRS1

⑥ Set qualified trace.

⑦ Integrate the event detection conditions, and set the following in BRM:
BRS2

⑧ Set the event detection point in the middle frame of trace memory.

⑨ Specify program address 100H as the emulation start address.

⑩ Emulation has started at program address 100H.

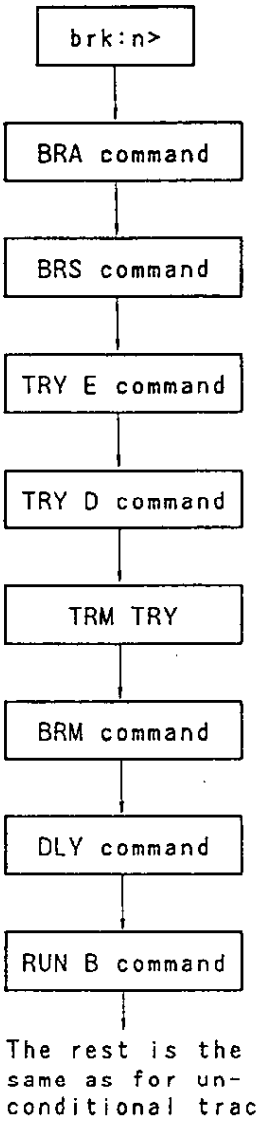
⑪ The system is ready for command input in the trace mode.

⑫ A break occurred by the event trigger source set in the BRS2 (program address 3000H).

⑬ The system enters the one-step execution mode.

(3) Example for executing the target program with sectional trace specification

(a) Procedure



① Set data memory event conditions. Specify a break point, trace point, and data values.

② Set program memory event conditions. Specify a break point and trace point.

③ Set a trace enable point. Specify the trace points specified in ① and ②.

④ Set a trace disable point. Specify the trace points specified in ① and ②.

⑤ Set a trace mode. Specify TRY (sectional trace).

⑥ Set a break mode. Specify the break points specified in ① and ②.

⑦ Set event detection points.

⑧ Real-time execution under break conditions

(b) Sample operation

In the following example, tracing starts when data memory address 0100H is accessed, and stops when data memory address 0200H is accessed, and a break occurs when the instruction at address 1000H is executed.

```
brk:0>BRA 1 MA=0100 V=00001111Y C=W8 <cr>                          ①
brk:0>BRA 2 MA=0200 V=0000XXXXY C=R8 <cr>                          ②
brk:0>BRS 1 P 1000 <cr>                                            ③
brk:0>TRY E BRA1 <cr>                                              ④
brk:0>TRY D RBA2 <cr>                                              ⑤
brk:0>TRM TRY <cr>                                                 ⑥
brk:0>BRM BRS1 <cr>                                                ⑦
brk:0>DLY M <cr>                                                   ⑧
brk:0>RUN B 100 <cr>                                               ⑨
Emulation start at 100                                             ⑩
trc:0>■                                                            ⑪
BRS1 break terminated                                              ⑫
XA HL DE BC XA' HL' DE' BC' RBS MBS RBE CY IST0 SP   PC        } (*)
00 00 00 00 00  00  00  00   0   0   0  0   0   000 0000      }
One step emulation standby                                         ⑬
```

* Registers not present in the target device are indicated with --.

① Set the following in BRA1 as an event detection condition:
Data memory address: 0100H (hexadecimal)
Data memory content (value): 00001111Y (binary)
Memory access condition:
Valid bits = 8 bits
I/O mode = write, or read-modify-write

② Set the following in BRA2 as an event detection condition:

Data memory address: 0200H (hexadecimal)

Data memory content (value):

0000XXXXY (binary mask specification)

Memory access condition:

Valid bits = 8 bits

I/O mode = read, or read-modify-write

Remark: X: undefined data

Y: binary representation

③ Set the following in BRS1 as an event detection condition:

Program memory address: 1000H (hexadecimal)

④ Set the following in TRY E as the trace start point of sectional trace:

BRA1

⑤ Set the following in TRY D as the trace stop point of sectional trace:

BRA2

⑥ Set sectional trace.

⑦ Integrate the event detection conditions, and set the following in BRM:

BRS1

⑧ Set the event detection point in the middle frame of trace memory.

⑨ Specify program address 100H as the emulation start address.

⑩ Emulation has started at program address 100H.

⑪ The system is ready for command input in the trace mode.

⑫ A break occurred by the trigger source in BRS1 (program address 1000H).

⑬ The system is in the one-step execution mode.

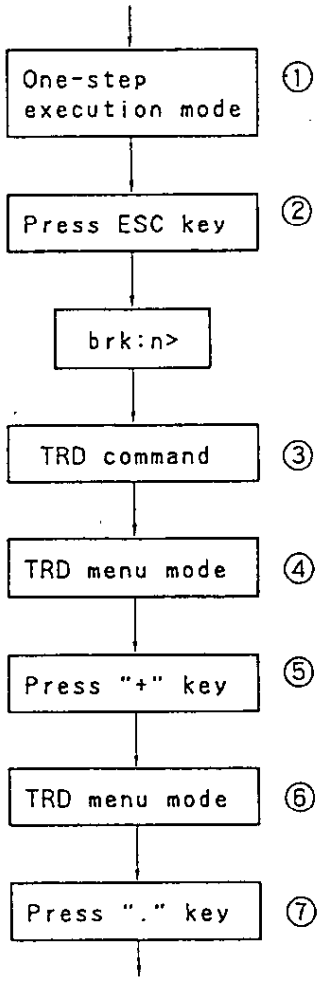
6.4.4 Checking results of execution

After a break occurs during execution of the target program because an event is detected, trace data, register values, special register values, and memory contents are checked to see that normal results are obtained.

(1) Checking trace data

The following shows a procedure for executing the target program with the RUN B command, and after an event is detected, checking trace information. For the procedure for entering the one-step execution mode after event detection, see Section 6.4.3.

(a) Procedure



① One-step execution mode immediately after a break

② Pressing the <ESC> key enters the break mode.

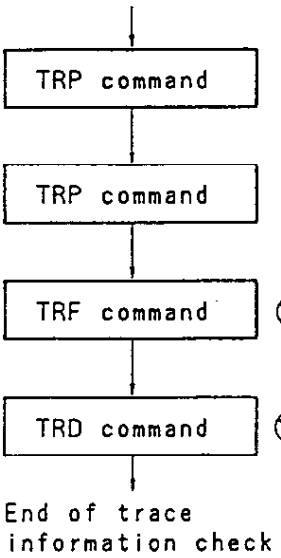
③ Display trace data.

④ TRD command menu mode

⑤ Display trace data following one page (12 lines) of trace data displayed.

⑥ TRD command menu mode

⑦ Terminate the TRD command.



⑧ Display the current pointer.

⑨ Point the pointer to the first trace data.

⑩ Set retrieval conditions.

⑪ Display retrieved trace data.

(b) Sample operation

```
One step emulation standby [ESC]                                          ①
brk:0>TRD DMEM <cr>                                                       ②

Frame  PA    PD          MA   MD  MRW  Label    Mnemonic                 ③
 00C   0104  8A          002  02  MRW           INCS     HL
 00D   0105  AA  4A       ---  --  ---           SKE      HA, HL
 00E                     002  02  MRD
 00F   0107  AB  01  02   ---  --  ---           BR       !0102H
 010                     ---  --  ---
 011                     ---  --  ---
 012   0102  89  11       ---  --  ---           MOV      XA, #0011H
 013                     000  11  MWR
 014   0104  8A          002  03  MRW           INCS     HL
 015   0105  AA  4A       ---  --  ---           SKE      XA, HL
 016                     002  03  MRD
 017   0107  AB  01  02   ---  --  ---           BR       !0102H
Total frame=1A1          (L/F/T/+/cr/-/Frame No./.)?+ <cr>                ④

Frame   PA   PD          MA   MD  MRW  Label    Mnemonic
 018                     ---  --  ---
 019                     ---  --  ---
 01A   0102  89  11       ---  --  ---           MOV      XA, #0011H
 01B                     000  11  MWR
 01C   0104  8A          002  04  MRW           INCS     HL
 01D   0105  AA  4A       ---  --  ---           SKE      XA, HL
 01E                     002  04  MRD
 01F   0107  AB  01  02   ---  --  ---           BR       !0102H
 020                     ---  --  ---
 021                     ---  --  ---
 022   0102  89  11       ---  --  ---           MOV      XA, #0011H
 023                     000  11  MWR
Total frame=1A1          (L/F/T/+/cr/-/Frame No./.)?  <cr>                ⑤
brk:0>TRP <cr>                                                            ⑥
  Total frame number   =1A1
  Display frame pointer=024                                               ⑦
brk:0>TRF MA=0 MD=11H MRW=MWR <cr>                                        ⑧
brk:0>TRD ALL $Q DMEM <cr>                                                ⑨
```

Frame	PA	PD			MA	MD	MRW	Label	Mnemonic	
!002	0102	89	11		---	--	---		MOV	XA, #0011H
003					000	11	MWR			
!00A	0102	89	11		---	--	---		MOV	XA, #0011H
00B					000	11	MWR			
!012	0102	89	11		---	--	---		MOV	XA, #0011H
013					000	11	MWR			
!01A	0102	89	11		---	--	---		MOV	XA, #0011H
01B					000	11	MWR			
!022	0102	89	11		---	--	---		MOV	XA, #0011H
023					000	11	MWR			
!02A	0102	89	11		---	--	---		MOV	XA, #0011H
02B					000	11	MWR			
!032	0102	89	11		---	--	---		MOV	XA, #0011H
033					000	11	MWR			
!03A	0102	89	11		---	--	---		MOV	XA, #0011H
03B					000	11	MWR			
!042	0102	89	11		---	--	---		MOV	XA, #0011H
043					000	11	MWR			
!04A	0102	89	11		---	--	---		MOV	XA, #0011H
04B					000	11	MWR			
!052	0102	89	11		---	--	---		MOV	XA, #0011H
053					000	11	MWR			
!05A	0102	89	11		---	--	---		MOV	XA, #0011H
05B					000	11	MWR			
!062	0102	89	11		---	--	---		MOV	XA, #0011H
063					000	11	MWR			
!06A	0102	89	11		---	--	---		MOV	XA, #0011H
06B					000	11	MWR			
!072	0102	89	11		---	--	---		MOV	XA, #0011H
073					000	11	MWR			
!07A	0102	89	11		---	--	---		MOV	XA, #0011H
07B					000	11	MWR			
!082	0102	89	11		---	--	---		MOV	XA, #0011H
083					000	11	MWR			
T0A3	0217	AB	02	04	---	--	---		BR	!0204H

① Terminate the one-step execution mode by pressing the <ESC> key.

② Specify the display of trace data to check them.

③ One page of trace data is displayed starting from the position pointed by the current pointer, then the menu mode is entered.
   . Immediately after emulation, the trace pointer points to the trigger frame. The TRD command displays the trigger frame and its preceding five lines and following five lines.
   . If there is no trigger frame, the pointer is pointed to the first frame.

④ To check the next page, enter + <cr> in the menu mode.

⑤ To terminate the menu mode of the TRD command, enter . <cr>.

⑥ Check the current pointer.

⑦ The pointer indicates to the next frame to the frame previously displayed.

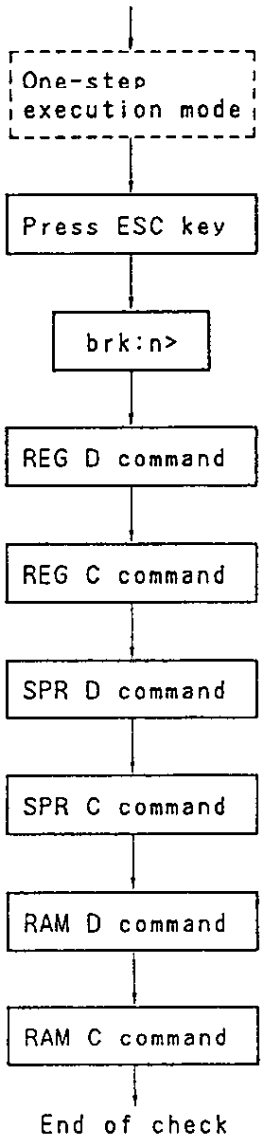
⑧ Set retrieval conditions to search the frame that wrote address 011H in the location at data memory address 0H.

⑨ Search the frames from the first frame for frames that satisfy the retrieval conditions, and display them.

(2) Checking register values, special register values, and memory contents

The following shows a procedure for starting target program execution with the RUN B command, after an event is detected, checking register values, special register values, and memory contents, and making correction if necessary.

(a) Procedure



① One-step execution mode immediately after a break

② Pressing the <ESC> key causes the break mode.

③ Display general registers.

④ Change general registers.

⑤ Display special registers.

⑥ Change special registers.

⑦ Display data memory contents.

⑧ Change data memory contents.

(b) Sample operation

```
brk:0>REG D <cr>                                                    ①

  XA HL DE BC XA' HL' DE' BC' RBS MBS RBE MBE CY IST1 IST0 SP   PC   ② (*)
  00 11 22 33 44  55  66  77   0   0   0   0  0   0    0   000 0178

brk:0>REG C <cr>                                                    ③

  XA      00 = 11 <cr>                                              ④
  HL      11 = 22 <cr>                                              ④
  DE      22 = 33 <cr>                                              ④
  BC      33 = <cr>                                                 ⑤ (*)
  XA'     44 = <cr>                                                 ⑤ (*)
  HL'     55 = <cr>                                                 ⑤ (*)
  DE'     66 = <cr>                                                 ⑤ (*)
  BC'     77 = <cr>                                                 ⑤ (*)
  RBS      0 = <cr>                                                 ⑤ (*)
  MBS      0 = <cr>                                                 ⑤ (*)
  SP     000 = <cr>                                                 ⑤ (*)
  PC    0178 = <cr>                                                 ⑤ (*)

brk:0>SPR D F8X <cr>                                                ⑥

  [F8X]
    SBS=1     BTM.33=.1  BTM=Wo  BT=00  DSPM=Wo DIMS=Wo             ⑦
    DIGS=Wo   KSF.3=.0   KSF=2

brk:0>SPR C BTM <cr>                                                ⑧

    BTM       Wo=1 <cr>                                             ⑨

brk:0>RAM D 100,11F $B <cr>                                         ⑩

    ADDR +0  +2  +4  +6  +8  +A  +C  +E  +10+12+14+16+18+1A+1C+1E   ⑪
    0100 00  01  02  03  04  05  06  07  08 09 0A 0B 0C 0D 0E 0F ...

brk:0>RAM C 100 $B <cr>                                             ⑫

    0100  00=10 <cr>                                                ⑬
    0102  01=11 <cr>
    0104  02=12 <cr>
    0106  03=  <cr>                                                 ⑭
```

* Registers not present in the target device are indicated with --.

① Specify display of register contents to check them.

② All general registers, control registers, and PSW flags in the current register bank are displayed.

③ Specify change of register contents.

④ Change the register contents as follows:
The XA register content, 00H, is changed to 11H.
The HL register content, 11H, is changed to 22H.
The DE register content, 22H, is changed to 33H.

⑤ For registers whose contents are to be left unchanged, press the <cr> key.

⑥ Specify display of special register contents and check them.

⑦ Group name F8X is specified, so the special registers that belong to F8X are displayed.

⑧ Specify a special register to be changed.

⑨ Change the content of special register BTM to 1.

⑩ Display 16 bytes of data in bytes starting at address 0100H to check data memory contents.

⑪ Sixteen bytes of data are displayed in bytes starting at data memory address 0100H.

⑫ Specify change of data memory contents in bytes from addresses 0100H to 0105H.

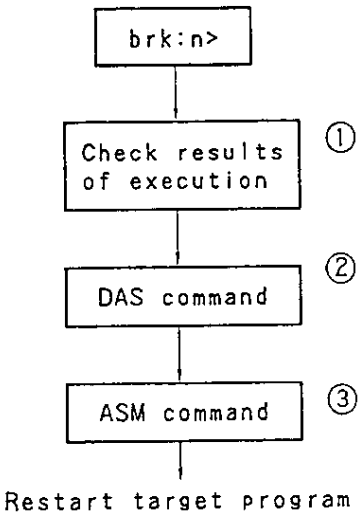
⑬ Change data memory contents.
Address 0100H Change 00H to 10H.
Address 0102H Change 01H to 11H.
Address 0104H Change 02H to 12H.

⑭ Terminate change of data memory contents.

6.4.5 Program correction

If the target program is found incomplete after results of execution are checked with trace data, register values, and memory contents, the program must be corrected.

(a) Procedure for correcting program



① Check trace data, register values, and memory contents.
See Section 6.4.4.

② Specify disassembly.
Disassemble program memory, then check the program.

③ Specify assembly.
Correct the program.

(b) Sample operation

In the following example, program memory contents are disassembled starting at program memory address 0100H, and the instruction at address 0104H is corrected.

```
brk:0>DAS 100 <cr>                                                    ①

  PA    Object       Mnemonic                                         ②
 0100   AA  10       MOV       @HL, XA
 0102   89  34       MOV       XA, #0034H
 0104   89  56       MOV       XA, #0056H
 0106   10           GET1      0020H
 0107   1F           GET1      003EH
 0108   89  00       MOV       XA, #0000H
 010A   92  80       MOV       0080H, XA
 010C   99  10       SEL       MB0
 010E   A3  30       MOV       A, 0030H
 0110   49           PUSH      XA

brk:0>ASM 104 <cr>                                                    ③

 0104   89  56       MOV       XA, #0056H       =MOV XA, #0078H <cr>   ④

        89  78                                                         ⑤
 0106   10           GET1      0020H            =END <cr>              ⑥

brk:0>■
```

① Disassemble program memory contents starting at address 0100H, and check the program.

② The instructions in the area starting at the specified address and ending at an address 16 bytes ahead are disassembled and displayed.

③ To correct the instruction at address 0104H, specify assembly of instructions from address 0104H.

④ Code currently set in memory and a disassembled image are displayed. Correct the instruction at address 0104H in mnemonic code.

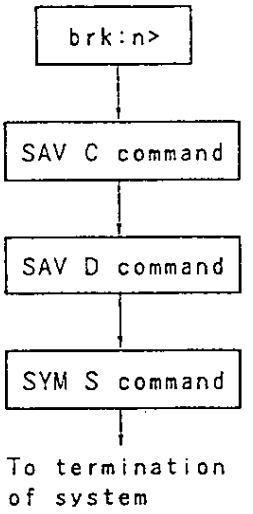
⑤ The object code of the results of correction is indicated.

⑥ The next instruction code and its disassembled image are displayed. Since the target instruction has been corrected, specify the end of assembly by entering END <cr>.

6.4.6 End of debugging

When debugging is completed or when debugging is stopped midway, the debugged program, debugging environment, and append symbols are saved so that debugging can be restarted in the same conditions as those present at the time of termination of previous debugging.

(a) Save procedure



① Save the program in an object file.

② Save the debugging environment information in a debugging environment information file.

③ Save the append symbols in an append symbol file.

In the above example, the program and debugging environment information are saved in separate steps ① and ②. These save operation, however, can be performed at one time using the SAV command.

(b) Sample operation

In the following example, object code, debugging environment information, and append symbols in the IE-75001-R are saved in the host machine.

```
brk:0>SAV SAMPLE C <cr>                    ①
 object save complete                      ②
brk:0>SAV SAMPLE D <cr>                    ③
 debug data save complete                  ④
brk:0>SYM S <cr>                           ⑤
brk:0>■
```

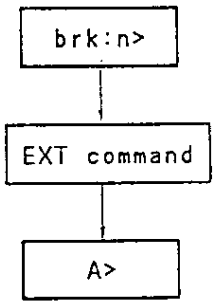
① Save program memory contents at locations from addresses 0H to 0FFFFH in a file named SAMPLE.HEX on the current drive of the host machine.
② The object code has been saved.
③ Save debugging environment information in a file named SAMPLE.DBG on the current drive of the host machine.
④ The debugging environment information has been saved.
⑤ Save append symbols in a file named IE75000.SYM on the current drive of the host machine.

6.4.7 System termination

Debugging operation is completed, the IE-75000-R control program is terminated, and the system is terminated.

Whenever application is ended, be sure to terminate the system.

(a) System termination procedure



① Terminate the IE-75000-R control program. Terminate the IE-75000-R control program, and return control to the OS (DOS).

② Control returns to the OS.

(b) Sample operation

The EXT command is entered to terminate the IE-75000-R control program and return control to the OS.

```
brk:0>EXT <cr>                                    ①
A>                                                ②
```

① Terminate the IE-75000-R control program.

② Control returns to the OS, and the OS displays the prompt.

CHAPTER 7 VERSION OF FIRMWARE ROM

7.1 Additional Functions

Upgrading the version (to Ver. 1.4) of
The version of the firmware ROM of the IE-75000-R was upgraded (to Ver. 1.4) on February 1993. If this new firmware ROM is used with the control program which was also upgraded (to Ver. 1.1) in the same period, the following new functions can be used. For detail, refer to the description of each command in Chapter 8 Command Description.

① A function to set a set-up file on start-up is added, so that the IE-75001-R can be automatically started up when started the second time and onward.
② NOT condition can be set when specifying an address of the BRA command.
③ NOT condition can be set when specifying a data value of the BRA command.
④ The BRA command can also be specified in the interactive mode.
⑤ The BNK command can be used also in the emulation mode.
⑥ An absolute address can be input with the TRP command.

7.2 Functional Differences by Version

The operations and functions of the IE-75001-R differ as indicated in the table below depending on the version of the firmware ROM and control program.

Table 7-1 Differences in Operation by Version of Firmware ROM and Control Program

Control program \ Firmware ROM	Ver. 1.0-1.3	*1 Ver. 1.4
Ver. 1.0	Operates	*2 Does not operate
*1 Ver. 1.1	Operates	*3 Operates

*1 Use the control program Ver. 1.1.

*2 The additional functions can be used.

CHAPTER 8 COMMAND EXPLANATION

This chapter explains all the IE-75001-R commands with their notation, the use conditions of the special keys, and the details of the functions. (Display examples are for the IE-75001-R connected with the IE-75000-R-EM.)

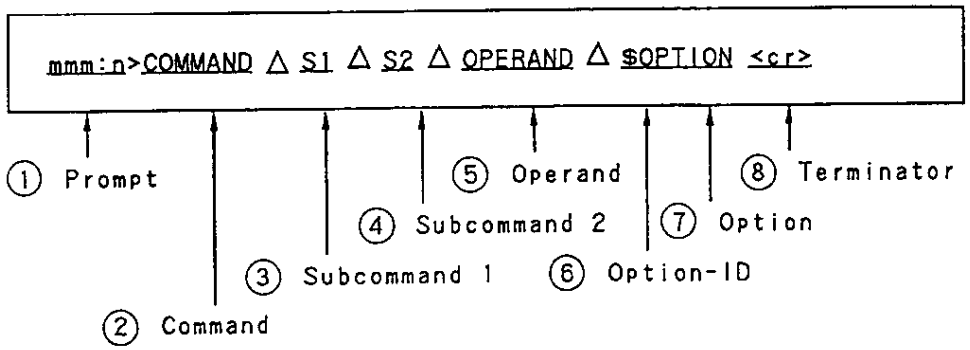
The commands are explained in alphabetical order.

8.1 Command Notation

8.1.1 Command format

(1) General command format

The general command format is shown below. A command must be input on a line.



① Prompt

This element is displayed on the monitor, and prompts the user to input a command.

mmm: System operation mode
n: Number of the serial port of the host machine

Prompt	System operation mode
brk:n>	Break mode
emu:n>	Emulation mode
trc:n>	Trace mode

② Command

This element represents the function of a command with three characters.

This element cannot be omitted.

③ Subcommand 1

This element modifies the function of a command.

Subcommand 1 and later elements may be omitted.

④ Subcommand 2

This element modifies the function of a command and subcommand 1. Subcommand 2 is specified only in some commands.

⑤ Operand

Numeric values, expressions, and so on are specified in this field.

⑥ Option-ID

This is an option identifier. When an option is specified, "$"(option-ID) must precede the option.

⑦ Option

This element modifies an operand.

⑧ Terminator

This element indicates the end of a command line.

(2) Command description rules

Description	Explanation
All-capital letters	Indicate that they must be input exactly as they are. Example: Main command, option, etc.
All-small letters	Indicate information to be supplied by the user. Example: Operand input
{Character strings}	Indicate that one of the enclosed character strings must be specified.
[Character string]	Indicates that the enclosed character string may be omitted.
(Character string)	Indicates that the enclosed character string is a subcommand or operand.
△	Indicates a space to be input. This is a separator for delimiting command elements.
¥	A backslash (\) is used instead in the IBM PC Series.

8.1.2 Explanation of command elements

(1) Main command

This element represents the function of a command with three characters. The command appearing in the format in all-capital letters must be input exactly as it is described. This element cannot be omitted.

Table 8-1 Commands

No.	Command	Function	System operation mode		
			brk	emu	trc
1	ASM	Assembles	Yes	No	No
2	BNK	Sets data memory banks	Yes	Yes	No
3	BRA	Sets data memory event conditions	Yes	Yes	No
4	BRK	Displays event information	Yes	Yes	No
5	BRM	Sets event mode	Yes	Yes	No
6	BRS	Sets program memory event conditions	Yes	Yes	No
7	CHK	Sets checkpoints	Yes	Yes	No
8	CLK	Selects clock	Yes	No	No
9	COM	Creates command files	Yes	Yes	Yes
10	CVD	Manipulates coverage measurements	Yes	No	No
11	CVM	Sets coverage measurement ranges	Yes	No	No
12	DAS	Disassembles	Yes	No	No
13	DIR	Displays directory	Yes	Yes	Yes
14	DLY	Sets event detection points	Yes	Yes	No
15	DOS	Passes control to OS	Yes	Yes	Yes
16	EXT	Exits control program	Yes	No	No
17	HIS	Displays command history	Yes	Yes	Yes
18	HLP	Displays help messages	Yes	Yes	Yes
19	LOD	Loads	Yes	No	No
20	LST	Redirects output	Yes	Yes	Yes
21	MAT	Computes	Yes	Yes	Yes
22	MEM	Manipulates program memory	Yes	No	No
23	MOD	Sets channel 2 mode	Yes	No	No
24	OUT	Sets external sense clip mode	Yes	Yes	No
25	PAS	Sets pass count	Yes	Yes	No
26	PGM	Sets terminal mode	Yes	No	No
27	RAM	Manipulates data memory	Yes	No	No
28	REG	Manipulates general registers	Yes	No	No
29	RES	Resets	Yes	Yes	Yes
30	RUN	Runs emulation	Yes	No	No
31	SAV	Saves	Yes	No	No

(to be continued)

Table 8-1 Commands (Cont'd)

No.	Command	Function	System operation mode		
			brk	emu	trc
32	SET	Switches emulated device mode	Yes	No	No
33	SPR	Manipulates special registers	Yes	No	No
34	STR	Redirects input	Yes	Yes	Yes
35	STP	Stops real-time emulations	No	Yes	Yes
	STP T	Stops real-time tracing	No	No	Yes
36	STS	Selects devices to be debugged	Yes	No	No
37	SYM	Manipulates symbols	Yes	No	No
38	SYS	Restarts the system	Yes	No	No
39	TRD	Displays trace data	Yes	Yes	No
40	TRF	Sets trace data retrieval conditions	Yes	Yes	No
41	TRG	Triggers tracer	No	Yes	No
42	TRM	Selects trace mode	Yes	Yes	No
43	TRP	Manipulates trace pointers	Yes	Yes	No
44	TRX	Specifies qualified-trace conditions	Yes	Yes	No
45	TRY	Specifies sectional-trace conditions	Yes	Yes	No
46	VRY	Verifies objects	Yes	No	No

(2) Subcommand

This element modifies the function of a command. There are two subcommands: subcommand 1 and subcommand 2. Subcommand 1 and later elements may be omitted.

(a) Subcommand 1

Parameter	Meaning	Parameter	Meaning
A	Append	M	Move, shift
B	Conditional	N	Unconditional
C	Change, chip	R	Read
D	Display	S	Size, step
E	Test	T	Step
F	Initialize, search	U	User
G	Search	V	Compare
H	Hardware	W	Write
I	IE-75001-R	X	Exchange
K	Erase	Z	Zero check
n	Register or bank number		

(b) Subcommand 2

Parameter	Meaning	Parameter	Meaning
P	Parallel	S	Sequential

(3) Operand

Numeric values, expression, and so on are specified in this field.

(a) Variable

The user inputs a numeric value or expression corresponding to the parameter.

Parameter	Meaning
addr	Address using an operational expression
addrx	Address using X coding
addrb	Address using bit specifications
bit	Single-bit numeric value
command	Command name
count	Decimal number
data	8-bit numeric value
expression	Expression, conditional expression
file	File name
group	Group name
module name	Module name for a symbol
option	Option
partition	Start and end addresses delimited by a comma, or address range using X coding
parameter list	List of parameters delimited by a space
register	Register name
string	String of characters or data items (data-string) delimited by a comma (,)
symbol	Symbol name
word	16-bit numeric value

(b) Constant

The user inputs the parameters exactly as they are described for each command. The table below lists such parameters.

Parameter	Meaning
BR?	Register name (BRA1-BRA4, BRS1-BRS2)
REG	Representative name of a general register
SPR	Representative name of a special register
TRD	Trace display
LST	List
CON	Close
F	First
M	Middle
L	Last
C	Object code
D	Debugging environment
S	Symbol file

(c) Operand prompt

In some commands, the following prompts are displayed. The table below lists such operand prompts. For details, see the explanation of each command.

Prompt	Meaning
C=	Status
MA=	Data memory address
MD=	Data memory data
MRW=	Data memory access status
E=	External sense clip data
EXT=	External sense clip data
PA=	Program memory address
PD=	Program memory data
Pn=	I/O port n data
V=	Value

(4) Option

This element modifies an operand. That is, additional information is given by an option. The operator must input the parameters for the option exactly as they are.

Parameter	Meaning
R	With register display
B	Byte specification (manipulate data memory command) Bit specification (trace command)
N	Nibble specification
I	Event cycle specification
M	Machine cycle specification

8.2 Coding Conventions for Numeric Values, Symbols, and Expressions

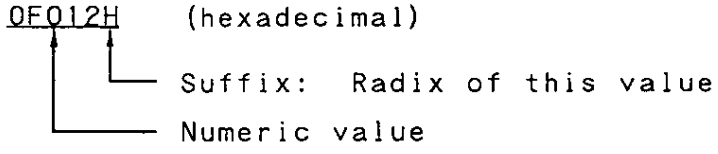
This section explains the conventions of coding numeric values, symbols, and expressions used as command operands of the IE-75001-R.

8.2.1 Coding conventions for numeric values

Numeric values are coded directly. The suffix placed after a numeric value indicates the radix of the value (hexadecimal, decimal, octal, or binary).

A numeric value can be replaced with a symbol or expression.

(1) Numeric representation



(a) The most significant digit of a numeric value must be a number (0 to 9).

(b) A sign, + or -, can be prefixed to a numeric value, and the sign must be followed by a number (0 to 9).

(c) When the most significant digit of a numeric value is not a number (0 to 9), the value is processed as a symbol.

(2) Size of a numeric value

Numeric values are processed according to the size specified in the command (in bits or in units of 4, 8, or 16 bits).

(3) Radix

Specify a suffix (H, T, Q, or Y) after a numeric value for the radix of the value (hexadecimal, decimal, octal, or binary).

(a) Normally, the radix is defined depending upon the functions of a command.

(b) When no suffix is specified for the radix, the defined radix (default) is assumed.

(c) When a radix which is different from the defined radix is used, specify a suffix corresponding to the radix.

Table 8-2 Maximum and Minimum Values for Each Radix

Radix	Minimum value	Maximum value			
	Common to all bit configuration	16 bits	8 bits	4 bits	2 bits
Hex.	0H	0FFFFH	0FFH	0FH	1H
Decimal	0T	65535T	255T	15T	1T
Octal	0Q	177777Q	377Q	17Q	1Q
Binary	0Y	1111111111111111Y	11111111Y	1111Y	1Y

Remark: For 0s and 1s, the suffix may be omitted.

8.2.2 Coding conventions for special numeric values (X coding)

A special numeric value indicates a special numeric representation. A numeric, symbolic, or expression representation indicates only one numeric value, but a special numeric representation indicates a group of several numeric values.

X is used to represent any numeric values in a special numeric representation. X represents 0 to F in hexadecimal notation, 0 to 7 in octal notation, or 0 or 1 in binary notation. No decimal number can be indicated using X.

The following explains the two coding conventions for special numeric values.

Coding as ranges
Coding as mask data

(1) Coding as ranges

When Xs are coded contiguously from the least significant digit, the special numeric representation indicates the range from the minimum value to the maximum value.

Note the following items:

- If the Xs are not contiguous, a mask data specification is assumed.

- When a numeric value in a special numeric representation (for a range) begins with X, 0 must be prefixed to the X. A value beginning with X is assumed to be a symbol.

Examples: Coding as ranges (for 16-bit data)

```
0XXXXH:       0H-0FFFFH
0XXXXXXQ:     0Q-177777Q
0XXXXXXXXY:   0Y-11111111Y
```

(2) Coding as mask data

Data using Xs at any digits of a numeric value are mask data. A number must precede X. If no number precedes X, 0 must be prefixed to the X.

Examples: Coding as mask data

```
0X00H:       0000H, 0100H, 0200H, 0300H,
             0400H, 0500H, 0600H, 0700H,
             0800H, 0900H, 0A00H, 0B00H,
             0C00H, 0D00H, 0E00H, 0F00H
01X1Q:       0101Q, 0111Q, 0121Q, 0131Q,
             0141Q, 0151Q, 0161Q, 0171Q
1X1010X1Y:   10101001Y, 11101001Y,
             10101011Y, 11101011Y
```

8.2.3 Coding conventions for symbols

Symbols are divided into local symbols and public symbols. Normally, the symbols are defined by loading them into the symbol table file.

The following explains the conventions for symbols and the maximum number of symbols.

(1) Symbol name

The following characters are valid for a symbol name.

A to Z, a to z, @, ?, _, 0 to 9

Note the following items:

. A symbol name must begin with a character other than numbers (0 to 9).

. Lower case alphabetic characters (a to z) are taken as upper case alphabetic characters (A to Z).

. A symbol name can contain up to eight characters.

. When nine or more characters are coded, only the first eight characters are valid.

(2) Symbol value

A symbol value is treated as 16-bit data.

(3) Module name

The coding conventions for a module name to be given to a symbol depend upon the type of symbol.

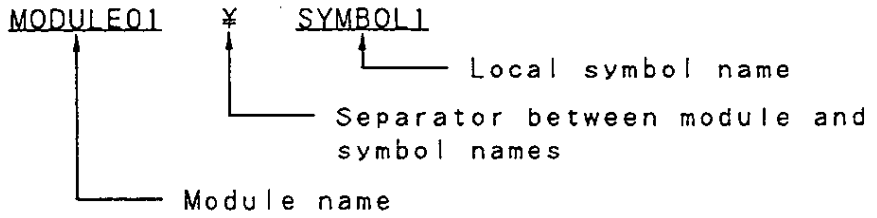
(a) For a public symbol

For a public symbol, specify a symbol name only. The module name (PUBLIC¥) need not be specified.

Remark: When an IBM PC Series is used as a host machine, ¥ is replaced with \.

(b) For a local symbol

For a local symbol, specify a module name before a symbol name as shown below.



When a current module is specified with an SYM M command, specify a symbol name only. A module name need not be specified.

Remark: When the name of a local symbol is the same as that of a public symbol, the public symbol is selected. See Section 8.4.37 for details.

(c) For an IE symbol

For an IE symbol, specify a symbol name only. The module name (IESYMBOL¥) need not be specified.

(4) Maximum number of symbols

The maximum number of symbols that can be registered in the IE-75001-R is as follows:

(a) Symbols in the symbol table file

Approximately up to 7000 symbols can be registered.

(b) IE symbols

Approximately up to 7000 symbols can be registered.

8.2.4 Coding conventions for expressions

An expression can be coded, instead of a numeric value, by connecting numeric values, symbols, or a numeric value and symbol with an operator.

(1) Operators

Operator	Priority	Remark
( ) * / + - AND OR XOR	↑ High | | Priority | ↓ Low	Up to 32 priority levels

Note the following items:

- Parentheses can be used for nesting up to 32 levels.
- Operators AND, OR, and XOR must be preceded by at least one space and followed by at least one space.
- No space is required between a numeric value or a symbol and an operator (,), *, /, +, or -.

(2) Operation

All operations are performed in 16-bit integers. If an intermediate result or the final result is longer than 16 bits, the 17th and later bits are discarded.

Operations are valid only for numeric values and symbols. An operation involving a reserved word or special numeric value causes an error.

8.3 Special Keys

The tables below lists the IE-75001-R special keys.

(1) Double key entry

Key entry	Description format	Function
[CTRL] + A	^A	Insert mode toggle switch
[CTRL] + C	^C	Control program forced termination
[CTRL] + H	^H	One character deletion
[CTRL] + I	^I	Space
[CTRL] + J	^J	End of line entry
[CTRL] + K	^K	Forced termination of an STR command
[CTRL] + L	^L	Temporary stop and restart of an STR command
[CTRL] + M	^M	End of line entry
[CTRL] + O	^O	Output switch of a COM command
[CTRL] + P	^P	Output switch of an LST command
[CTRL] + Q	^Q	Release of temporary stop by ^S
[CTRL] + S	^S	Temporary stop
[CTRL] + X	^X	One line deletion

(2) Single key entry

Key entry	Description format	Function
[BS]	<BS>	One character deletion
[↵]	<cr>	End of line entry
[DEL]	<DEL>	One character deletion
[ESC]	<ESC>	End of command execution
[TAB]	<TAB>	Space
[;]	;	Start of a comment
[!]	!	History calling
[←]	←	Cursor left move
[→]	→	Cursor right move

8.4 Command Explanation

8.4.1 Assemble command (ASM)

General format Format 1 ASM [△ addr]	
Radix	addr:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The ASM command assembles the program input in 75X Series mnemonic codes starting at a specified program address, into object codes and writes the object codes into the program memory.

Use of this command enables us to change and modify program using mnemonic codes. That is, the machine language need not be used.

The default value of addr is 000H.

(a) Assembler operation

① The assembler fetches object codes starting at a specified program address, disassembles them into 75X Series mnemonic codes, and displays them.

② The assembler displays the prompt (=) and waits for a source program in mnemonic codes to be input.

③ The assembler assembles the input mnemonic codes into object codes and writes them into the program memory. At the same time, the assembler displays the assembled object codes on the next line.

(b) Input data to the assembler

The assembler accepts two assembler control instructions (ORG addr and END) and mnemonic codes for a device to be debugged as input data. The table below shows such instructions and mnemonic codes.

Instruction or code	Function
ORG △addr	Changes an address to be assembled to an address specified in addr.
END	Terminates the processing of an assemble command.
Mnemonic code	Mnemonic codes for a device to be debugged ① The type of device is specified with an STS command. ② When an I/O instruction is to be used, an I/O address can be specified in the I/O reserved word.

(c) Error handling

If an error is found in input mnemonic code, the system displays the following error message and waits for re-entry of codes.

Message: "******** Error!! ********"

Programming:

Operand

addr: Specify the address of the program memory to start assembling mnemonic codes.

Valid address range: 0 to 0FFFFH

Caution: The valid address range depends on the target devices.

Programming note:

This assembler does not check the length of an object code. Thus, use an NOP instruction to pad the remaining bytes of a single-byte instruction to be changed to a double-byte instruction. If an NOP instruction is not used, the first byte of the next instruction to an instruction to be changed is painted.

Example:

This sample program changes a program using mnemonic codes starting at program address 1000H.

```
brk:0>ASM 1000H <cr>                                          ①

 Addr  Code    Label   Mnem. Operand
 1000  60              NOP             =MOV X, #0 <cr>        ②
 1000  9A 09                                                  ③
 1002  60              NOP             =END <cr>              ④

brk:0>■
```

① Specification to start assembling from address 1000H

② The contents at address 1000H are displayed and NOP is changed to MOV X, #0.

③ The object code is displayed.

④ The contents of the next instruction and address 1002H are displayed and assembling is terminated.

8.4.2 Set data memory bank command (BNK)

General format		
Format 1	BNK [△ n]	(0≦n≦0FH)
Radix	n:H	

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The BNK command changes the BNK contents to a specified bank number. In this case, specify the four high-order bits of the data memory address when a symbol in the operand is transformed during disassembling.
The n is set to 00H by default.

(a) BNK reference

The BNK contents are referenced only by assemble (ASM) commands and disassemble (DAS) commands, and the contents do not affect real-time execution.

(b) Determining the data memory access address

In 75X Series, the data memory access address is obtained by adding the following two elements (12 bits).

- Value of the operand in an instruction (low-order 8 bits)
- Value of the memory bank register (MBS) during execution of an instruction (high-order 4 bits)

(c) Displaying the data memory bank address

When the operand is omitted, the current data memory bank address is displayed.

(d) After emulation, value of BNK is set as follows:

1. When MBE = 0 -----> BNK = 0

2. When MBE = 1 -----> BNK = MBS value

Programming:

Operand

n: Specify a data memory bank address in hexadecimal notation.
Range: 0 ≦ n ≦ 0FH
Default: Current data memory bank address

Example:

This sample program changes the data memory bank address from 0 to 1.

```
brk:0>BNK <cr>                    ①
  00                              ②
brk:0>BNK 1 <cr>                  ③
```

① Specification to check the current bank address
② The current bank address is displayed.
③ The bank address is changed from 0 to 1.

8.4.3 Set data memory event condition command (BRA1 to BRA4)

General format

Format 1 BRA[△ n][△ MA= { addr / partition }][△ V=data1][△ C=status]
[△ E=data2]

(1≦n≦4)

Radix	addr:H	partition:H	data1:Y	data2:Y

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The BRA command sets an event condition. For example, when the programmer accesses the data memory under the conditions set with BRA commands, a break may be caused or trace data may be output.

(a) Number of event conditions to be set

One BRA command can set only one event condition. The maximum number of BRA commands to be specified is four. This number corresponds to the number of data memory event condition registers.

The conditions set in BRA commands are valid when they are selected by the set event mode command (BRM).

Programming:

Subcommand

n: Specify the number of a data memory event condition register.
Register number range: 1 to 4

Operands

MA=$\begin{matrix}\text{addrX}\\ \text{partition}\end{matrix}$: Specify the address of the data memory. The valid address range depends on the target devices. If "/" is added at the beginning of the addr, the program breaks when accessed any address other than specified one.
When BRA1 and BRA2 are specified, the address mask specification and partition specification are possible.
When BRA3 and BRA4 are specified, only the address mask specification is possible.

Valid address range: 0 to 0FFFH
Valid mask range: 0 to 0XXXH
Default: Previous value
Initial value: 0XXXH (hexadecimal mask specification)

V=data1: Specify a data value in the following format. If "/" is added at the beginning of the data, the program breaks when accessed any data value other than specified one:
Valid data value range: 0 to 0FFH
Valid mask range: 0 to 0XXXXXXXXY
Default: Previous value
Initial value: 0XXXXXXXXY (binary mask specification)

Remark: The number of valid bits follows the specification in status.

Example: When R4 is specified in status, only the four low-order bits are valid.

C=status: Specify a condition to access the data memory.

Format	Access condition	
	Number of valid bits	I/O mode
R4	4	Read, read-modify-write
W4	4	Write, read-modify-write
RW4	4	Read, write, read-modify-write
R8	8	Read, read-modify-write
W8	8	Write, read-modify-write
RW8	8	Read, write, read-modify-write
SP	Data of the address specified by the stack pointer	

Default: Previous value
Initial value: RW4

E=data2: Specify a data value to be input from the external sense clip in the following format. If "/" is added at the beginning of the data, the program breaks when a value except specified one is input:

Valid data value range: 0 to 0FFH
Valid mask range: 0 to 0XXXXXXXXY
Default: Previous value
Initial value: 0XXXXXXXXY (binary mask specification)

Programming note:

To trigger an event under the condition set with a BRA command, the data memory event registers (BRA1 to BRA4) must have been selected with the set event mode command (BRM). For details, see Section 8.4.5.

Notes 1. When a break condition is specified in the next instruction after the skip instruction has been executed, if the break condition is satisfied, the program does not break. The condition set is neglected.

Example:

```
Loop:MOV   A, @HL
       :
       :
     INCS  A
     BR    Loop
     MOV   X, A      <-- If the break condition is
                         set at this point, the
                         program does not break.
     MOV   XA, @HL   <-- In this example, set the
                         break condition at this
                         point.
```

2. For real-time execution under a break point (RUN B), a program breaks as follows:

   ① When the break condition is R4 or R8, a program breaks after a break condition is satisfied and the next instruction is executed.

   ② When the break condition is W4 or W8, a program breaks after the break condition is satisfied and the next two instructions are executed if the first instruction is a one-byte instruction or after the break condition is satisfied and the next one instruction is executed if the instruction is a two- or three-byte instruction.

Examples:

(a) Setting event conditions

This sample program sets conditions in BRA1 to BRA4 so that events are detected when data memory addresses 0100H to 0103H are accessed.

```
brk:0>BRA 1 MA=0100 V=00001111Y C=W8 <cr>          ①
brk:0>BRA 2 MA=0101 V=0000XXXXY C=R8 <cr>          ②
brk:0>BRA 3 MA=0102 V=0AH C=W8 <cr>                ③
brk:0>BRA 4 MA=0103 V=010H C=W8 <cr>               ④
brk:0>BRM BRA1 BRA2 BRA3 BRA4 <cr>                 ⑤
brk:0>RUN B <cr>                                   ⑥
```

① The following contents are set in BRA1 as an event detection condition.
Data memory address: 0100H (hexadecimal)
Data memory value: 00001111Y (binary)
Memory access condition:
Valid bits: 8 bits
I/O mode: Write, read-modify-write

② The following contents are set in BRA2 as an event detection condition.
Data memory address: 0101H (hexadecimal)
Data memory value: 0000XXXXY (binary mask specification)
Memory access condition:
Valid bits: 8 bits
I/O mode: Read, read-modify-write

③ The following contents are set in BRA3 as an event detection condition.

Data memory address: 0102H (hexadecimal)
Data memory value: 0AH (hexadecimal)
Memory access condition:
Valid bits: 8 bits
I/O mode: Write, read-modify-write

④ The following contents are set in BRA4 as an event detection condition.

Data memory address: 0103H (hexadecimal)
Data memory value: 010H (hexadecimal)
Memory access condition:
Valid bits: 8 bits
I/O mode: Write, read-modify-write

⑤ The event conditions set in BRA1 to BRA4 are set in the event mode register (BRM). This setting validates the event conditions.

⑥ This specification starts emulation at the current program address.

(b) Specifying NOT of address

```
brk:0>BRA 1 MA=/100H V=0XXXXXXXXY C=RW4 E=0XXXXXXXXY<cr>  ①
brk:0>
```

① The program breaks when accessing to address other than 100H in 4 bits.

(c) Specifying NOT of data value

```
brk:0>BRA 1 MA=/100H V=/0H C=RW4 E=0XXXXXXXXY<cr>
brk:0>
```

①

① The program breaks if data value is other than 0 when accessing to address 100H in 4 bits.

(d) Specifying NOT of external sense clip value

```
brk:0>BRA 1 MA=/100H V=0H C=RW4 E=/0FFH<cr>
brk:0>
```

①

① The program breaks if data value is 0 when accessing to address 100H in 4 bits and if the external sense clip value is other than 0FFH.

(e) When the subcommand and later elements are omitted

When the subcommand and later elements are omitted, the set contents of BRA1 to BRA4 are displayed.

```
brk:0>BRA <cr>

  BRA1:MA=0XXXH     V=0XXXXXXXXY   C=R8   E=0XXXXXXXXY
  BRA2:MA=0XXXH     V=0XXXXXXXXY   C=R8   E=0XXXXXXXXY
  BRA3:MA=0XXXH     V=0XXXXXXXXY   C=R8   E=0XXXXXXXXY
  BRA4:MA=0XXXH     V=0XXXXXXXXY   C=R8   E=0XXXXXXXXY
```

①
②

① The subcommand and later elements are omitted.
② The contents of BRA1 to BRA4 are displayed.

(f) When the operands and later elements are omitted

When the operands and later elements are omitted, the system enters the menu mode. Thus, event conditions can be set in the interactive mode.

```
brk:0>BRA 1 <cr>                                          ①

  MA 0H,0FFFH=0XXXH <cr>                                  ②

   V 10001Y=<cr>                                          ③

     4bit D-mem Read          (R4)                        ④
     4bit D-mem Write         (W4)
     8bit D-mem Read          (R8)
     8bit D-mem Write         (W8)
     4bit D-mem Read/Write    (RW4)
     8bit D-mem Read/Write    (RW8)
     Stack Pointer Access     (SP)

   C W8=R8 <cr>                                           ⑤

   E 0XXXXXXXXY=<cr>                                      ⑥

brk:0>
```

① The operands and later elements are omitted.
② The memory address is changed to an address for the mask specification.
③ The data value is not changed.
④ The guidance for the statuses is displayed.
⑤ The status is changed from W8 to R8.
⑥ The external sense clip data value is not changed.

8.4.4 Display event information command (BRK)

General format

Format 1 BRK

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The BRK command displays the previous break source, trace stop source, and the contents of an event condition register for real-time emulation.

(a) Operation mode and display contents

The contents of the event information to be displayed depends on the system operation modes.

Emulation mode: The trace stop source is displayed.
Break mode: The break source is displayed.

(b) Register displayed

The contents of the following event condition registers are displayed.

BRA1, BRA2, BRA3, BRA4, BRS1, BRS2

(c) Guide break

The IE-75001-R has the function to cause a break forcibly when the function which the target device does not have is executed.

① GDM: Guard data memory
When the data memory which is not specified is accessed

② GDSP: Guard stack pointer
When the stack pointer which is not specified is accessed

③ GDR: Guard register
When the register which is not specified is accessed

④ GDIO: Guard I/O
When the mode register I/O area which is not specified is accessed

Programming:

Enter the command only.

Examples:

(a) Displaying a break source

This sample program displays a source when a break is caused under the event condition set in BRA2 in the break mode.

```
brk:0>BRK <cr>

  Event Register Information
  (ena) BRA1:MA=123, 346H              V=0XXXXXXXXY  C=RW8   E=0XXXXXXXXY
  (dis) BRA2:MA=LABEL001, LABEL002     V=1Y          C=R4    E=11Y
  (dis) BRA3:MA=0XXXX1XXXXXXXY         V=1XXY        C=W8    E=1XXXY
  (dis) BRA4:MA=0XXXH                  V=0XXXXXXXXY  C=RW4   E=0XXXXXXXXY
  (dis) BRS1:Sequential
             PA=LABEL003               E=0XXXXXXXXY
             PA=0XXXX1XXXXXXXXXXX1     E=0XXXXXXXXY
             PA=0XXXXH                 E=11Y
  (dis) BRS2:PA=0XXXXH                 E=0XXXXXXXXY

  BREAK CAUSED BY BRA1
```

① ②

③

④

⑤

① Specification to display the event information

② The contents of the data memory event registers (BRA1 to BRA4) are displayed.

③ The contents of the program memory event register (BRS1) is displayed.

S (Sequential): A set condition with up to four points can be displayed.

P (Parallel): A set condition with up to seven points can be displayed.

④ The contents of the program memory event register (BRS2) is displayed.

Only one event condition can be displayed.

⑤ The event condition register name and the break source are displayed.

Register: BRA1, BRA2, BRA3, BRA4, BRS1, BRS2

Source: GDM, GDSP, GDR, GDIO, STP

Whether each event condition register is valid or not as an event detection condition (each register is set in BRM or not) is displayed.

(ena): Valid (set in BRM)
(dis): Invalid (not set in BRM)

(b) Displaying a trace stop source

This sample program displays a source when a break is caused under the event condition set in BRS1 in the emulation mode.

```
emu:0>BRK <cr>                                                          ①

  Event Register Information
  (dis) BRA1:MA=123, 346H              V=0XXXXXXXXY  C=RW8  E=0XXXXXXXXY   ②
  (dis) BRA2:MA=LABEL001, LABEL002     V=1Y          C=R4   E=11Y
  (dis) BRA3:MA=0XXXX1XXXXXXXY         V=1XXY        C=W8   E=1XXXY
  (dis) BRA4:MA=0XXXH                  V=0XXXXXXXXY  C=RW4  E=0XXXXXXXXY
  (ena) BRS1:Sequential                                                 ③
             PA=LABEL003               E=0XXXXXXXXY
             PA=0XXXX1XXXXXXXXXXXY     E=0XXXXXXXXY
             PA=0XXXXH                 E=11Y
  (dis) BRS2:PA=0XXXXH                 E=0XXXXXXXXY                    ④

  TRACER STOPPED BY BRS1                                                ⑤
```

① Specification to display the event information

② The contents of the data memory event registers (BRA1 to BRA4) are displayed.

③ The contents of the program memory event register (BRS1) is displayed.

S (Sequential): A set condition with up to four points can be displayed.

P (Parallel): A set condition with up to seven points can be displayed.

④ The contents of the program memory event register (BRS2) is displayed.
Only one event condition can be displayed.

⑤ The event condition register name and the trace stop source are displayed.
Register: BRA1, BRA2, BRA3, BRA4, BRS1, BRS2
Source: GDM, GDSP, GDR, GDIO, STPT

Whether each event condition register is valid or not as an event detection condition (each register is set in BRM or not) is displayed.

(ena): Valid (set in BRM)
(dis): Invalid (not set in BRM)

8.4.5 Set event mode command (BRM)

General format
Format 1 BRM[△ BR?][△ BR?][△ BR?][△ BR?][△ BR?][△ BR?]

brk:n>	o	emu:n>	o	trc:n>	x

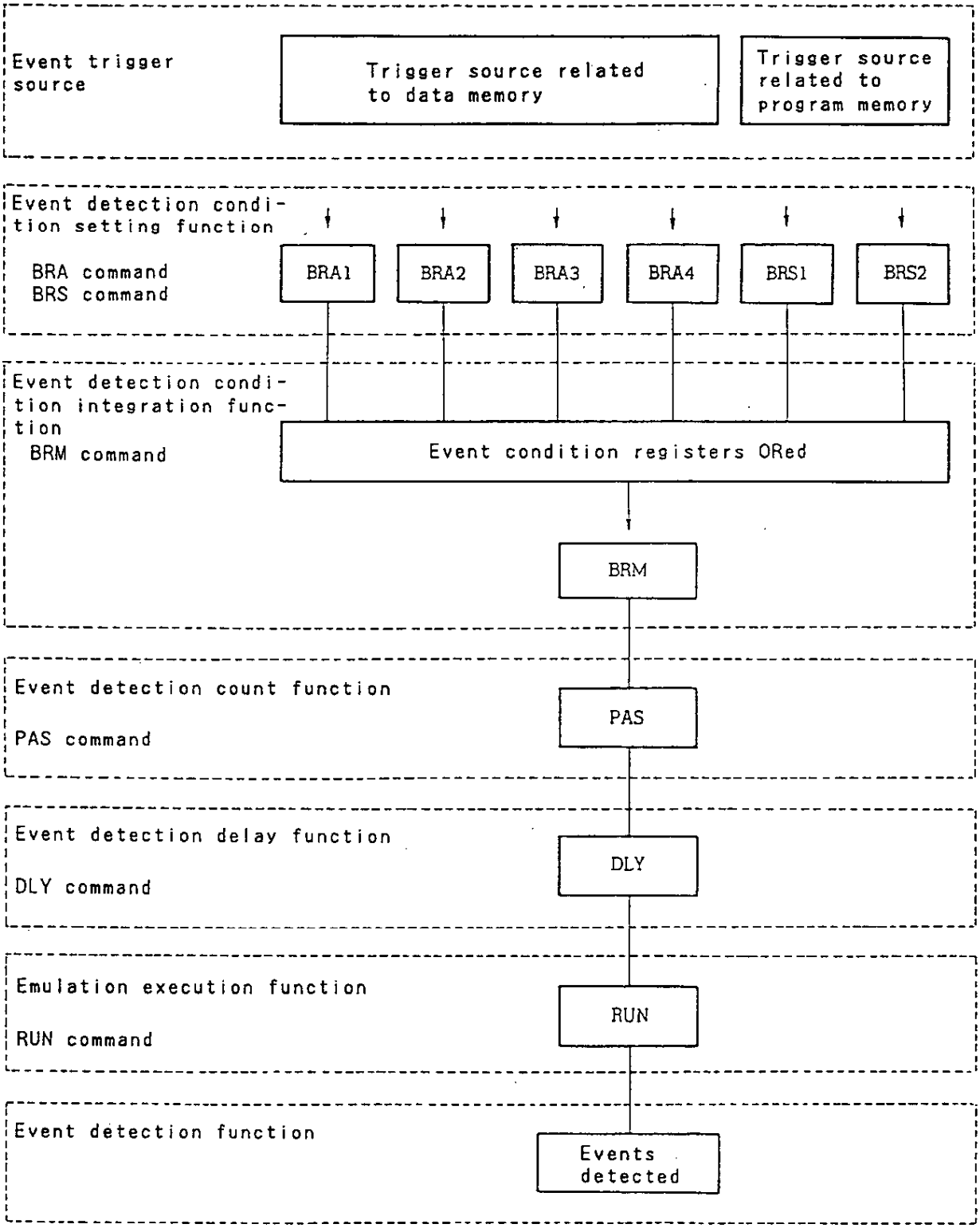
Function:

The BRM command integrates the event conditions set in the event condition registers as the final event conditions into the event mode register.

That is, the contents of the event condition registers specified by a BRM command are valid as the trace or break event detection condition. BRS1 is specified by default.

Figure 8-1 shows the general operation procedure from the setting of event detection conditions to the detection of events.

Fig. 8-1 Procedure to Set and Detect Event Conditions



(a) When the operand is omitted, the event condition register name selected in the current event mode register (BRM) is displayed.

(b) When BRA3 and BRA4 are specified with an OUT command, BRA3 and BRA4 cannot be specified with a BRM command.

Programming:

Operand

BR?: Up to six of the following event condition registers can be specified.

BRA1, BRA2, BRA3, BRA4, BRS1, BRS2, OFF

Remark: When OFF is specified, the event detection condition is invalidated and real-time execution is not affected.

Initial value: BRS1

Default: When the operand is omitted, the event condition register name set in the current event mode register (BRM) is displayed.

Example:

Setting event condition registers

This sample program selects the four event conditions which are set in the data memory event registers (BRA1 to BRA3) and the program memory event register (BRS1) in the event mode register (BRM).

```
brk:0>BRA 1 MA=0100 V=000001111Y C=W8 <cr>                    ①
brk:0>BRA 2 MA=0101 V=00000XXXXY C=R8 <cr>                    ②
brk:0>BRA 3 MA=0102 V=0AH C=W8 <cr>                           ③
brk:0>BRS 1 P 02000H 02500H,XXXX0000Y <cr>                    ④
brk:0>BRM BRA1 BRA2 BRA3 BRS1 <cr>                            ⑤

brk:0>BRM <cr>                                                ⑥
  BRA1 BRA2 BRA3 BRS1                                         ⑦
brk:0>RUN B <cr>                                              ⑧
```

① The event condition is set in BRA1.
② The event condition is set in BRA2.
③ The event condition is set in BRA3.
④ The event condition is set in BRS1.
⑤ All the event conditions set in BRA1 to BRA3 and BRS1 are selected in BRM.
⑥ Specification to display the selected contents (check)
⑦ The selected contents are displayed.
⑧ Specification to start emulation

8.4.6 Set program memory event condition command (BRS)

General formats

Format 1 BRS[△ 1] [△ S / △ P] [△ addrx[,data]]...
... [△ addrx[,data]]

Format 2 BRS △ 2 [△ addrx[,data]]

Radix	addrx:H data:Y

Function overview:

The BRS command sets an instruction executed at a specified program memory address or data input from the external sense clip, as a condition to trigger an event.

For example, when the program passes through the address specified in a BRS command, a break may be caused or trace data may be output.

Programming note:

1. When a break condition is set in the following instructions, if the break condition is satisfied, a program breaks after an instruction which is next to the instruction with the break condition is executed. When the following instructions are set consecutively, however, under the same conditions stated above, a program breaks after an instruction other than the following instructions is executed.

```
SET1   0FBX   (EI 0FBX)
CLR1   0FBX   (DI 0FBX)
SEL    0MBX
```

Example:

```
①  CLR1   0FB2.3   ←— Where a break condition is set
   MOV    A, #0H   ←— Where a program breaks

②  SET1   0FB4.3   ←— Where a break condition is set
   SET1   0FB5.0
   SEL    0MB0
   MOV    A, #0H   ←— Where a program breaks
```

2. When a break condition is set in an instruction which is skipped, a program breaks if the instruction is skipped and not executed.

(1) Setting multiple events

Format 1 BRS[△ 1] [△S / △P] [△ addrx[,data]]... ... [△ addrx[,data]]					
Radix	addrx:H data:Y				

brk:n>	o	emu:n>	o	trc:n>	x

Function:

Sets specified program memory addresses and specified data values to be input from the external sense clip in the event condition register having the number (1) specified in subcommand 1 as the event detection condition. Events are detected in the order specified in subcommand 2. Up to four sequential events or up to seven parallel events can be set at a time.

Programming:

(a) Subcommand 1

Specify the number (1) of a program memory event condition register.

(b) Subcommand 2

Specify the order in which events are detected.

S: Sequential event detection specification
When the program is executed in the order specified in the operand, events are detected.

P: Parallel event detection specification
When the program executes either of the conditions specified in the operand, events are detected.

(c) Operands

Specify events to be detected. The number of specifiable events varies with the detection mode (sequential or parallel).

addrx: Specify the address of a program for event detection.

Valid address range:	0 to 0FFFFH
Valid mask range:	0 to 0XXXXH
Number of events:	Up to four (sequential)
	Up to seven (parallel)

data: Specify a data value to be input from the external sense clip.

Valid address range:	0 to 0FFH
Valid mask range:	0 to 0XXXXXXXXY
Number of events:	Up to four (sequential)
	Up to seven (parallel)

(2) Setting a single event

Format 2 BRS[△ 2] [△ addrx[,data]]					
Radix	addrx:H data:Y				
brk:n>	o	emu:n>	o	trc:n>	x

Function:

Sets a specified program memory address and a specified data value to be input from the external sense clip in the event condition register having the number (2) specified in subcommand 1 as the event detection condition.

Only one event condition can be set. This format is used to set the start and end addresses for section tracing.

Programming:

(a) Subcommand

Specify the number (2) of a program memory event condition register.

(b) Operands

Specify events to be detected. Only one event can be specified.

addrx: Specify the address of a program for event detection.

Valid address range:	0 to 0FFFFH
Valid mask range:	0 to 0XXXXY
Number of events:	One

data: Specify a data value to be input from the external sense clip.

Valid address range:	0 to 0FFH
Valid mask range:	0 to 0XXXXXXXXY
Number of events:	One

Examples:

(a) Specification for multiple events

This sample program specifies multiple events as the break condition in the program memory event register (BRS1).

```
brk:0>BRS 1 P 2000H 2500H, 0XXXX0000Y <cr>                    ①

brk:0>BRM BRS1 <cr>                                           ②


brk:0>BRS <cr>                                                ③

  BRS1: Parallel

        PA=2000H            E=0XXXXXXXXY
        PA=2500H            E=0XXXX0000Y
  BRS2: PA=0XXXXH           E=0XXXXXXXXY

brk:0>RUN B <cr>                                              ④
```

① The event condition is set in BRS1.
② The event condition set in BRS1 is selected in BRM (the detection condition is settled).
③ Specification to display the selected contents (check)
④ Specification to start emulation

(b) Specification for a single event

This sample program sets the section trace start and end points in the program memory event registers (BRS1 and BRS2).

```
brk:0>BRS 1 S 2000H <cr>                    ①
brk:0>BRS 2 3000H <cr>                      ②
brk:0>TRY E BRS1 <cr>                       ③
brk:0>TRY D BRS2 <cr>                       ④
brk:0>TRM TRY <cr>                          ⑤
brk:0>RUN N <cr>                            ⑥
```

① The start address is set in BRS1.
② The end address is set in BRS2.
③ The trace valid point (start point) is set in BRS1.
④ The trace invalid point (end point) is set in BRS2.
⑤ Specification for section tracing
⑥ Specification to start emulation

(c) When operand and those that follow are omitted

when the operand and those that follow are omitted, the menu mode is set, and event condition can be set in the interactive mode.

```
brk:0>BRS 1 <cr>                                    ①
 PARALLEL (P)/SEQUENTIAL (S) P=P <cr>               ②
 PA, EXT 0XXXX, 0XXXXXXXXY     =300, 11100011 <cr>  ③
 PA, EXT 0XXXX, 0XXXXXXXXY     =400 <cr>
 PA, EXT 0XXXX, 0XXXXXXXXY     =. <cr>              ④
brk:0>
```

① Specifies omission of operand and those that follow

② Selects parallel event detection

③ Sets break point

④ Terminates with input of dot (.)

Note When inputting <cr> only, the previous value is valid. By inputting C <cr>, the value on the line can be cleared.

```
brk:0>BRS 1 <cr>                                    ①
 PARALLEL (P)/SEQUENTIAL (S) P=S <cr>               ②
 PA, EXT 0300, 011100011Y      =3FF <cr>
 PA, EXT 0400, 0XXXXXXXXY      =400, 00001111 <cr>
 PA, EXT 0XXXX, 0XXXXXXXXY     =410 <cr>
 PA, EXT 0XXXX, 0XXXXXXXXY     =400 <cr>
brk:0>
```

① Selects sequential event detection

② Sets break point

```
brk:0>BRS 2 <cr>
 PA, EXT 0XXXX, 0XXXXXXXXY          =500, 01010101 <cr>
brk:0>
```

① ②

① Specifies omission of operand and those that follow

② Sets event condition

(d) When deleting an event condition that was previously specified.

An event condition that was previously specified can be deleted.

```
brk:0>BRS 1
 PARALLEL (P) /SEQUENTIAL (S)  S=S<sr>
 PA, EXT  10H, 0XXXXXXXXY  =C<cr>
 PA, EXT  20H, 0XXXXXXXXY  =<cr>
 PA, EXT  20H, 0XXXXXXXXY  =, <cr>
brk:0>BRS<cr>

 BRS1:Sequential
      PA=20H            E=0XXXXXXXXY
      PA=30H            E=0XXXXXXXXY
      PA=10H            E=0XXXXXXXXY
      PA=0XXXX          E=0XXXXXXXXY
 BRS2:PA=0XXXX          E=0XXXXXXXXY

BRK:0>
```

① Selects sequential event detection.

② Inputs C <cr> when the event condition to be deleted is displayed.

8.4.7 Set checkpoint command (CHK)

General formats

Format 1 CHK [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [ΔREG]

Format 2 CHK [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [ΔSPR] [ΔSPR]
[ΔSPR] [ΔSPR]
[ΔSPR]

Format 3 CHK [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [ΔBR?] [Δaddr] [Δaddr]
[Δaddr] [Δaddr]
[Δaddr]

Operand 1 Operand 2

Radix	addr:H

Function overview:

The CHK command stops real-time execution started by a RUN command when it detects the contents of an event condition register specified in operand 1, in which checkpoints are set. Then it writes the check data specified in operand 2 to the trace memory. The check data are contents of general register, special register, or memory. After that, this command restarts real-time execution.

(a) During real-time execution DLY (the interval between an event detection and a break), check data are not written to trace memory even if check conditions specified by the CHK command are satisfied.

(b) If no operand is specified, currently selected checkpoint and check data are displayed.

(c) The user can check the data written in trace memory using a display trace data command (TRD).

(d) When BRA3 and/or BRA4 is specified by the OUT command, the CHK command cannot specify BRA3 and/or BRA4.

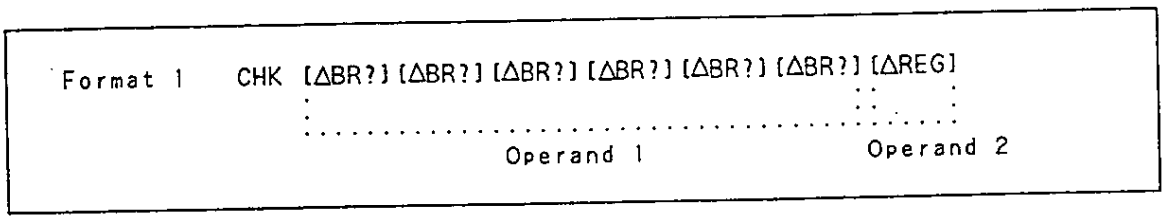
Programming note:

(a) It is necessary to have checkpoints already set in the event condition registers.

(b) When a condition which is set with the CHK command is satisfied at the time when an interrupt occurs, the CHK command is processed first, and then the interrupt is processed. When there are conditions of the CHK commands consecutively under the same case stated above, the consecutive CHK commands are processed first, and then the interrupt is processed.

Remark: See Section 8.4.39 for the screen display image.

(1) Specifying that check data of contents of general register be output



brk:n>	o	emu:n>	o	trc:n>	x

Function:

Writes the contents of general register to trace memory as check data when any one of event detection conditions selected in operand 1 is satisfied.

Programming:

(a) Operand 1

Select event condition registers in which checkpoints are set.

BR?: Specify up to six names of event condition registers (six points).

BRA1, BRA2, BRA3, BRA4, BRS1, BRS2, OFF

(b) Operand 2

Specify the representative name of the general register of which contents are to be output as check data.

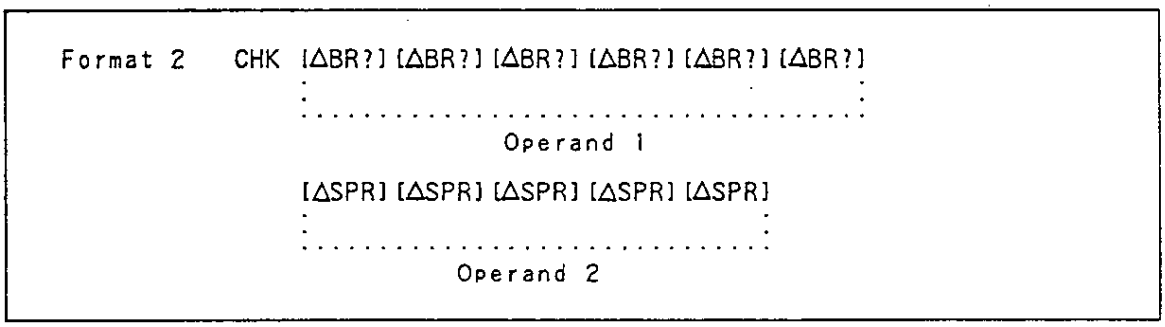
REG: Specify REG.

Contents of the following registers and flags can be output as check data.

But selection of registers depends on the target devices.

XA, HL, DE, BC (pair register)
XA', HL', DE', BC'
SP (including SBS), PC,
RBS (control register), MBS
CY, RBE, MBE, IST0 (flag), IST1

(2) Specifying that check data of contents of special register be output



brk:n>	o	emu:n>	o	trc:n>	x

Function:

Writes the contents of special registers specified in operand 2 to trace memory as check data when any one of event detection conditions selected in operand 1 is satisfied.

Programming:

(a) Operand 1

Select event condition registers in which checkpoints are set.

BR?: Specify up to six names of event condition registers (six points).

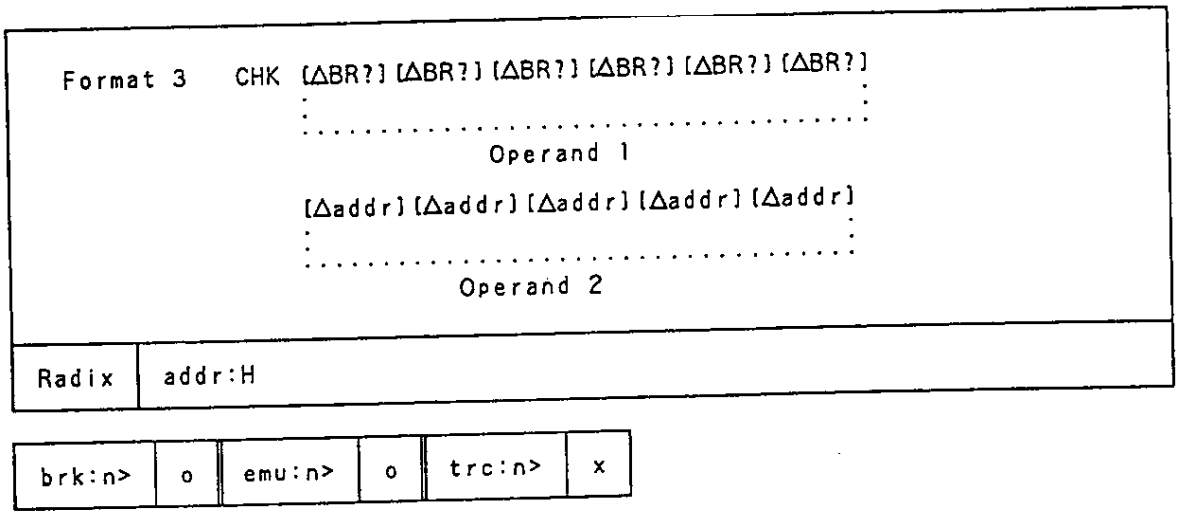
BRA1, BRA2, BRA3, BRA4, BRS1, BRS2, OFF

(b) Operand 2

Specify the special registers of which contents are to be output as check data.

SPR: Specify up to five names of special registers.

(3) Specifying that check data of contents of data memory be output



Function:

Writes the contents of the data memory specified in operand 2 to trace memory as check data when any one of event detection conditions selected in operand 1 is satisfied.

Programming:

(a) Operand 1

Select event condition register in which checkpoints are set.

BR?: Specify up to six names of event condition registers (six points).

BRA1, BRA2, BRA3, BRA4, BRS1, BRS2, OFF

(b) Operand 2

Specify the data memory address to be output as check data.

addr: Specify up to five addresses of the data memory. No mask can be specified.

Valid address range: Depends on the target devices.

Example:

(a) Specifying that check data of contents of a general register be output

This sample program selects BRA1 and BRA2 as checkpoints and specifies that the contents of the current general register be written as check data to trace memory.

```
brk:0>BRA 1 MA=0100H V=11110000Y C=W8 <cr>          ①
brk:0>BRA 2 MA=0101H V=00001111Y C=W8 <cr>          ②
brk:0>CHK BRA1 BRA2 REG <cr>                        ③
brk:0>RUN N <cr>                                    ④
```

① Checkpoints are set in BRA1.
② Checkpoints are set in BRA2.
③ BRA1 and BRA2 are selected and the contents of general register are specified to be output.
④ Emulation is started.

(b) Specifying that check data of contents of a data memory be output

This sample program selects BRS1 as a checkpoint and specifies that the contents of the current data memory be written as check data to trace memory.

```
brk:0>CHK <cr>                                          ①
  BRA 1 BRA2 REG                                        ②
brk:0>BRS 1 P 02000H 02500H <cr>                        ③
brk:0>CHK BRS1 0100H 0200H 0123H <cr>                   ④
brk:0>RUN N <cr>                                        ⑤
```

① Currently selected checkpoints are checked.
② Currently selected checkpoints are displayed.
③ Checkpoints are set in BRS1.
④ BRS1 is selected and the contents of data memory are specified to be output.
⑤ Execution of emulation is started.

8.4.8 Select clock command (CLK)

General format

Format 1 CLK [ΔU / ΔI]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The CLK command selects a clock in the target system or in the IE-75001-R as a clock source.

If no operand is specified, the name of currently set clock is displayed.

Note: When the target system clock is selected. Be sure to set the parts table (SX1, SX2) of the emulation board. If no parts table is set, clock is not applied to the emulation device.
Concerning the setting of the user clock, refer to "IE-75000-R-EM USER'S MANUAL"

Programming:

Operand

Specify the clock to be selected in the operands.

U: Select the clock in the target system.
I: Select the clock in the IE-75001-R.

Programming note:

When a clock source is selected, the system automatically executes the RES command.

Example:

Changing clock source

This sample program changes the clock source from the currently set clock in the IE-75001-R side to the clock in the target system.

```
brk:0>CLK <cr>                    ①
  IE                              ②
brk:0>CLK U <cr>                  ③
brk:0>CLK <cr>                    ④
  User
```

① Currently selected clock is checked.
② Currently selected clock is displayed.
③ Clock in the target system is selected.
④ Result is checked.

8.4.9 Create command file command (COM)

```
General formats
  Format 1     COM[ △d:file]
  Format 2     COM[ △LST:]
  Format 3     COM[ △CON:]
```

Function overview:

The COM command opens a file on the floppy-disk drive or a list output device specified in the operand to create a command file. And it closes these opened devices or files.

Output to a command file is started or stopped in the following procedures.

① Starting output of a command file

Enter the ^O (CTRL + O) key while the system is in the command input waiting status. The commands entered after this line are output to a file or a list device.

② Stopping output of a command file

Enter the ^O (CTRL + O) key while a command file is output and the system is in the command input waiting status. Output of commands to a file or a list device is stopped from this line.

(1) Specifying that a floppy-disk drive file be opened

Format 1 COM[△ d:file]

brk:n>	o	emu:n>	o	trc:n>	o

Function:

Opens the file on a floppy-disk drive specified in the operand to create a command file.

If the name of the file is already present, the following processes are performed according to the specification of file attributes.

(a) When the file attributes are DIR and R/W

The system displays the following message and allows the user to select to delete the existing file and create a new command file or to retain the existing file unchanged.

File already exists. Delete? (Y or N):____

Y: Delete the existing file and create a new command file.

N: Retain the existing file unchanged.

(b) When the file attribute is SYS or R/O

The system displays the following message and ignores the command.

File already exists.

Programming:

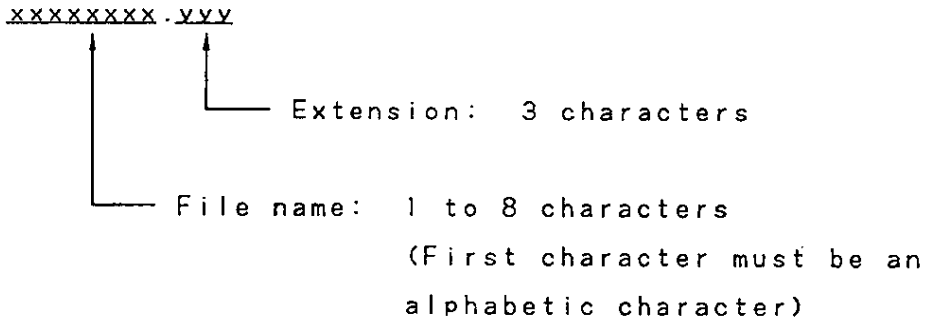
Operand

Specify the floppy-disk drive number to be opened and the file name.

d: Specify the floppy-disk drive number.

A: Floppy-disk drive-A
B: Floppy-disk drive-B

file: Specify the file name.



(2) Specifying that a list device be opened

Format 2	COM[△LST:]

brk:n>	o	emu:n>	o	trc:n>	o

Function:

Opens a list device specified in the operand to output commands and data. If the list device is being used by another process, the message indicated below is displayed and the command is ignored.

List device is used by other process.

Programming:

Operand

Specify LST:.

(3) Specifying that a command file output device be closed

Format 3	COM[△CON:]

brk:n>	o	emu:n>	o	trc:n>	o

Function:

Closes the opened file on a floppy-disk drive or the opened list output device.

Programming:

Operand

Specify CON:.

Example:

This sample program creates an command file in a file on a floppy-disk drive.

Command lines ②, ③, ④, ⑤, and ⑥ are output to the command file.

```
brk:0>COM B:SAMPLE.STR <cr>                          ①
brk:0>MEM F 0, 1FF 1, 2, 3, 4, 5, 6, 7 <cr>          ②(*)
brk:0>MEM D 010X <cr>                                ③
  0100 01 02 03 04 05 06 07...
brk:0>BRA 1 MA=100 <cr>                              ④
brk:0>BRM BRA1 <cr>                                  ⑤
brk:0>TRM ALL $M <cr>                                ⑥
brk:0>■                                              ⑦(*)
```

① A file named SAMPLE.STR in drive B is opened.

② Enter ^O (CTRL + O). Then enter MEM F 0, 1FF 1, 2, 3, 4, 5, 6, 7 <cr>.

③ ④ ⑤ ⑥ ② to ⑥ and the result are output to the command file.

⑦ Enter ^O. Close SAMPLE.STR.

* ^O is not displayed on the screen.

8.4.10 Manipulate coverage measurement command (CVD)

General formats	
Format 1 CVD[△ D] [partition]	
Format 2 CVD[△ K]	
Radix	partition:H

Function overview:

The CVD command displays and deletes (or initializes) the result of coverage measurement. To initialize means to return to the status before execution of coverage measurement.

The result of a coverage measurement is indicated by the following identifiers. Those identifiers indicate on which part of the target program the emulation is performed, and on which part it is not performed.

Identifier	Meaning	Priority at 64-byte display
X	Execution out of the range of coverage measurement	1
*	Execution within the range of coverage measurement	3
.	Non-execution within the range of coverage measurement	2
△	Non-execution out of the range of coverage measurement	4

Remark: For setting the coverage measurement range, see Section 8.4.11.

Programming note:

Coverage measurement is performed on the range specified by the set coverage measurement range command (CVM). It is necessary to have set the range before coverage measurement is performed.

(1) Specifying that the result of coverage measurement be displayed

Format 1 CVD[△ D] [partition]					
Radix	partition:H				
brk:n>	o	emu:n>	x	trc:n>	x

Function:

Displays the result of coverage measurement of a program memory represented by its start and end addresses. These addresses are specified in the operand.

The result is displayed in unit of one byte. So, one byte of object code is displayed as corresponding to one identifier (any of X, *, ., △).

(a) If subcommand and subsequent operand are not specified, the system displays the result of measurement on program memory starting from address 0 to address OFFFFH. In this case, it is displayed in unit of 64 bytes. That is, one identifier is displayed for corresponding 64 bytes of the object code.

(b) If no operand are specified, the system performs the same operation as above.

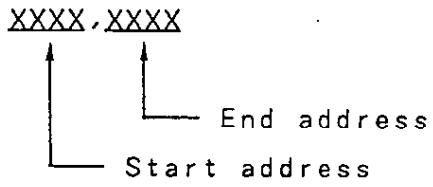
Programming:

(a) Subcommand

D: Specify D.

(b) Operand

partition: Specify the result of coverage measurement to be displayed in the start and end addresses.



(2) Specifying that the result of coverage measurement be deleted

Format 2 CVD △K

Function:

Deletes the identifiers of a coverage measurement result, and returns the target program to the status in which the emulation has not been performed.

So, measurement identifiers "X" is changed into "△", "*" is changed into ".".

Programming:

Subcommand

K: Specify K.

Example:

(a) Specifying that the range of coverage measurement result be displayed

This sample program displays the measurement result of a program memory of which range is indicated by the start address (address 100H) and the end address (address 1FFH).

```
brk:0>CVD D 100, 1FF <cr>                                              ①

 addr  0                 10  ..................................  30    ②
 0100  ********************  ..................................  ****  }③
 0140  ********************  ..................................  ****  }
 0180  **********........    ..................................  ...
 01C0  ..................    ..................................  ...   ④

brk:0>■
```

① Addresses 100H to 1FFH are specified to be displayed.

② The measurement result is displayed in unit of one byte.

③ The range where emulation has been executed is displayed.

④ The user can check that the measurement has been done to the address 18A (it was not executed after 18A).

(b) Specifying that the result of coverage measurement on the whole memory be displayed

This sample program displays the result of measurement on whole program memory in unit of 64 bytes by not specifying the subcommand and operands.

```
brk:0>CVD <cr>                                                    ①

 addr  000 100  200 300 400 ..........................  F00      ②
 0000  ********************  ..........................  ****
 1000  ... ********.......   ..........................  ...
 2000  ..................   ..........................  ...
   .
   .
   .
 F000

brk:0>■
```

① The CVD command is entered without specifying subcommand and operands.

② After this line, the measurement result is displayed in unit of 64 bytes.
(The user can check that the measurement has been executed from address 0 to 0FFF and from address 1100 to 12FF).

(c) Specifying that the result of coverage measurement be deleted

This sample program deletes the result of coverage measurement. The measurement identifiers "*" (Already executed) is changed into "." (Not executed).

```
brk:0>CVD K <cr>
brk:0>■
```
①

① The result of coverage measurement is specified to be deleted.

8.4.11 Set coverage measurement range command (CVM)

General formats

Format 1	CVM[△ A] [partition]
Format 2	CVM[△ D]
Format 3	CVM[△ K] [partition]

Radix	partition:H

Function overview:

The CVM command sets the range of coverage measurement, displays that range, or cancels the range setting.

The coverage function of IE-75001-R is called C0 coverage. This is a function to add identifiers to the portion of target program where a real-time emulation was executed by a RUN command. By these identifiers, the user can easily determine on which portion of target program the emulation has been executed, on which portion it has not been executed, or whether the portion is out of the range of emulation.

Identifier	Meaning
X	Execution out of the range of coverage measurement
*	Execution within the range of coverage measurement
.	Non-execution within the range of coverage measurement
Δ	Non-execution out of the range of coverage measurement

Remark: To check the contents of the result of coverage measurement, use a CVM command. See Section 8.4.10.

(1) Setting or adding a coverage measurement range

Format 1 CVM[Δ A] [partition]	
Radix	partition:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Sets or adds the range of coverage measurement of a program memory represented by its start and end addresses specified in the operand.

If subcommand and subsequent operand are not specified, the command performs the same function as the display coverage measurement range command (CVMD).

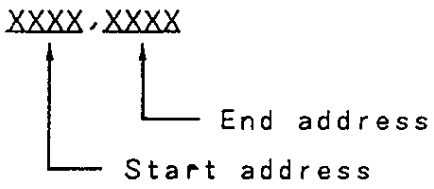
Programming:

(a) Subcommand

A: Specify A.

(b) Operand

partition: Specify the range of coverage measurement by the start and end addresses.



(2) Displaying the range of coverage measurement

Format 2 CVM[△ D]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Displays the range of coverage measurement.

When it is immediately after loading of an object, the loaded area is set as coverage measurement range.

Programming:

Subcommand

D: Specify D.

(3) Canceling the range of coverage measurement

Format 3 CVM[△ K] [partition]	
Radix	partition:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Cancels the range of coverage measurement of a program memory represented by the start and end addresses specified in the operand.

If no operand is specified, the whole program memory (addresses 0 to 0FFFFH) is canceled from the coverage measurement range.

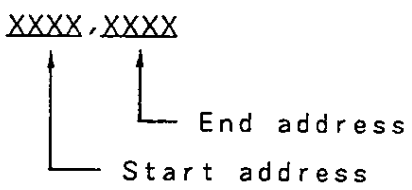
Programming:

(a) Subcommand

K: Specify K.

(b) Operand

partition: Specify the range of coverage measurement to be canceled in the start and end addresses.



Example:

(a) Setting the range of coverage measurement

This sample program sets ranges of coverage measurement of a program memory indicated by the start address (address 0H) and end address (address 1FFFH). Then it adds ranges 3000 to 3FFFH and 5500 to 5FFFH.

```
brk:0>CVM A 0, 1FFF <cr>                    ①
brk:0>CVM A 3000, 3FFF <cr>                 ②
brk:0>CVM A 5500, 5FFF <cr>                 ③
brk:0>■
```

① Addresses 0H to 1FFFH are specified as a measurement range.

② Addresses 3000 to 3FFFH are added to the measurement range.

③ Addresses 5500 to 5FFFH are added to the measurement ranges.

(b) Specifying that the coverage measurement ranges be displayed

This sample program displays the ranges of coverage measurement set in (a) above so that the user can check it.

```
brk:0>CVM D <cr>                                    ①
    0000,1FFF 3000, 3FFF 5500, 5FFF                 ②
brk:0>■
```

① The coverage measurement ranges are specified to be displayed.

② The ranges are displayed.

(c) Specifying that the coverage measurement ranges be deleted

This sample program deletes the ranges of coverage measurement set in (a) above so that the user can check it.

```
brk:0>CVM K 3000, 3FFF <cr>                         ①
brk:0>CVM D <cr>                                    ②
  0000,1FFF 5500, 5FFF                              ③
brk:0>CVM K <cr>                                    ④
brk:0>CVM D <cr>                                    ⑤
  nothing                                           ⑥
brk:0>■
```

① Addresses 3000 to 3FFFH are specified to be deleted from the measurement range.

② The coverage measurement range is specified to be displayed.

③ The currently set ranges are displayed.

④ The whole memory is deleted from the measurement range.

⑤ The coverage measurement range is displayed and checked.

⑥ The message to show that no range is set

8.4.12 Disassemble command (DAS)

General format
Format 1 DAS [△addr / △partition]

Radix	addr:H partition:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The DAS command converts (disassembles) the object code of the specified program memory into mnemonics of the 75X Series and displays them on the screen.

Using this and an assemble command frees the user from reading complicated machine languages and facilitates the Omodification and correction of programs.

(a) Disassembling the instruction sets

The instruction sets and the names of the SPR reserved words which can be disassembled depend on the target devices.

(b) Disassembling data

In case of disassembling the instructions to manipulate a data memory, symbol conversion is performed only when the following conditions are all satisfied.

① There is a symbol of data attribute (D).

② The higher 4 bits of the value of the symbol above are equal to the value specified by the BNK command.

③ The lower 8 bits of the value of the symbol are same as the data memory address specified in the operand of that instruction.

(c) In address specification, the instructions are disassembled and displayed.

(d) In partition specification, the data indicated by the start address and end address are disassembled and displayed.

(e) If no operand is specified, the instructions up to at the seventeenth byte as counted from the line at which the previous disassembling terminated are disassembled and displayed.

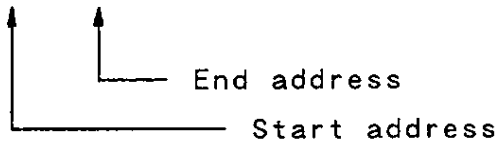
Programming:

Operands

addr: Specify the start address of disassembling.
Valid address range: 0 to 0FFFFH

partition: Specify the range to be disassembled by the start and end addresses.

XXXX, YYYY (XXXX<YYYY)



XXXX: 0 to 0FFFFH
YYYY: 0 to 0FFFFH

Example:

This sample program disassembles a program memory starting from address 100H.

```
brk:0>DAS 100 <cr>                                          ①

 Addr  Code       Label      Mnem.   Operand
 0100  9A09                  MOV     X,#0000H               ②
 0102  71                    MOV     A,#0001H
 0103  9A2F                  MOV     B,#0002H
 0105  9A3E                  MOV     C,#0003H
 0107  9A4D                  MOV     D,#0004H
 0109  9A5C                  MOV     E,#0005H
 010B  9A6B                  MOV     H,#0006H
 010D  9A7A                  MOV     L,#0007H
 010F  60                    NOP
 0110  60                    NOP

brk:0>■
```

① Program memory from address 100H is specified to be disassembled.

② Disassembled result from address 100H is displayed.

8.4.13 Display directory command (DIR)

General format

Format 1 DIR[△file]

brk:n>	o	emu:n>	o	trc:n>	o

Function:

The DIR command displays all contents of the directory of current disk drive or the directory of a file.

If no operand is specified, all contents of the directory of current disk drive are displayed.

Programming:

Operand

file: Specify a file name.

Example:

This sample program displays the contents of the directory of the current drive.

```
brk:0>DIR <cr>

 Directory=A:¥SS
<.           > <..          >   COMMAND COM   SAMPLE1 EXE   SAMPLE2 EXE
 IE75000   HLP   IE75000   EXE   ASM01   DAT   ASM02   DAT   DEMO1   EXE
 SAMPLE3   EXE   SAMPLE    MAC   ERROR   EXE   ERRSUB  EXE   DUMP    EXE
 DISPLAY   EXE   EXTERNAL  MAC   FILE    LST

brk:0>■
```

8.4.14 Set event detection point command (DLY)

General format

Format 1 DLY [△F | △M | △L]

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The DLY command sets the trace stop timing for an event condition detection during execution of a real-time emulation.
There are following three relationships between the event detection points (trigger points) and the trace information which is to be output to the trace memory. The user can select any one of these three relations.
L (Last) is set by default.

(a) Setting F (First)

The event detection point is set at the beginning of the trace memory.

When event is detected, the trace data which were output immediately before the detection are discarded and the trace data at the time of the event detection (the trigger frame) are output at the beginning of the trace memory. Then the output of the trace data is continued until the trace stops when the trace memory becomes full.

The user can check the trace data after the trigger frame.

(b) Setting M (Middle)

The event detection point is set at the middle of the trace memory.

The trace data at the time of the event detection (the trigger frame) are set at the middle of the trace memory. Tracing is stopped when the latter half of the trace memory becomes full.

The user can check the trace data before and after the event detection point.

(c) Setting L (Last)

The event detection point is set at the last of the trace memory.

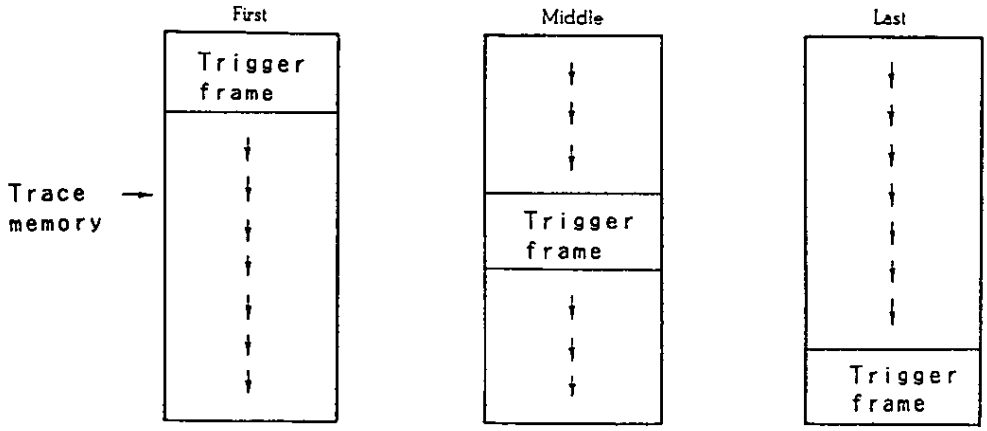
When event is detected, the trace data at the time of the event detection (the trigger frame) are output to the trace memory and the system immediately stops the tracing.

The trigger frame is output at the end of the trace memory.

The user can check the trace data before the trigger frame.

The relationships described above are illustrated as shown below.

Fig. 8-2 Points of Trigger Frames



(d) Specifying no operand

If no operand is specified, the setting status of event detection point is displayed.

Programming:

Operands

F: Specify the event detection point at the beginning of the trace memory.
M: Specify the event detection point at the middle of the trace memory.
L: Specify the event detection point at the last of the trace memory.

Example:

This sample example displays and checks the currently set event detection point, then sets it at the middle of the trace memory and displays its contents.

```
brk:0>DLY <cr>                    ①
  FIRST                           ②
brk:0>DLY M <cr>                  ③
brk:0>DLY <cr>                    ④
  MIDDLE                          ⑤
brk:0>■
```

① Current setting is specified to be displayed.
② Current setting is displayed.
③ The event detection point is set at the middle of the trace memory.
④ Current setting is specified to be displayed.
⑤ Current setting is displayed.

8.4.15 Run DOS command command (DOS)

General format

Format 1 DOS

brk:n>	o	emu:n>	o	trc:n>	o

Function:

The DOS command temporarily transfers control to the operating system (DOS) without terminating the IE-75000-R control program. By this means, various command functions can be available.

To return control to the IE-75000-R control program, use the EXIT command of the operating system.

Programming:

Enter the command only.

Programming note:

Do not insert nor extract a floppy disk while the control has been transferred to the operating system by this command. Otherwise, the contents of the floppy disk may be destroyed.

This function can be available only when COMMAND.COM is executed. Retrieve COMMAND.COM from the data of COMSPEC at first. If it is not found, refer to the PATH data, then retrieve COMMAND.COM.

If COMMAND.COM is found in the retrieval, use the COMMAND.COM. If it is not found, you cannot use this function.

Example:

This sample program transfers control to the operating system through the DOS command, uses various operating system commands, then returns control to the IE-75000-R control program.

```
brk:0>DOS <cr>                                    ①

A>TIME <cr>                                       ②
:
:
A>EXIT <cr>                                       ③

brk:0>                                            ④
```

① Specified that control be transferred to the operating system.

② A prompt of the operating system is displayed and the commands of the operating system become available.

③ Specified that control be returned from the operating system to the IE-75000-R control program.

④ The control is returned to the IE-75000-R control program, and a prompt of the IE-75000-R control program is displayed.

8.4.16 Exit control program command (EXT)

General format

Format 1 EXT

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The EXT command terminates the processing of the IE-75001-R control program and transfers control to the operating system (DOS).

When terminating a job, always use this command.

Programming:

Enter the command only.

Programming note:

(a) To terminate a job, always use this command.
(b) This command can be used in the break mode only.

Example:

This sample program finishes all the debugging work, and returns control to the operating system.

```
brk:0>EXT <cr>                                    ①
A>■                                               ②
```

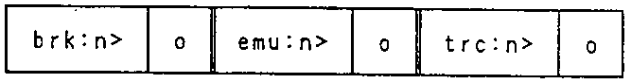
① Specified that control be returned to the operating system (DOS).

② Control is returned to the operating system, and the operating system displays the prompt.

8.4.17 Display command history command (HIS)

General format

Format 1 HIS



Function:

The HIS command displays and reexecutes the commands stored in command history memory.

(a) Displaying commands

This command displays commands stored in history memory, reading them while adding numbers to them sequentially from the oldest to the latest.

Since history memory contains up to 20 lines of commands including the latest command, the HIS command can display up to 20 lines.

(b) Reexecuting a command

The commands stored in history memory can be reexecuted after being called up as follows.

(i) Calling by number specification

To call up a particular command, follow the format below.

!n <cr> (n: Command number)

(ii) Calling the latest command

To call up the latest command, follow the format below.

!! <cr>

(iii) Reexecuting the command

After the called command is displayed, the cursor is displayed at the end of the command line.

Then, enter <cr>.

Programming:

Enter the command only.

Example:

(a) Displaying the contents of command history

This sample program displays the commands stored in command history memory.

```
brk:0>HIS <cr>                                    ①

1  LOD TEST                                       ②
2  CLK
    :
    :
20 HIS

brk:0>■
```

① Specified that command history be displayed.

② Commands are displayed with command numbers added to each. (The 20th line is the latest command).

(b) Re-executing a command

This sample program calls up the CLK command, which command number is 2, after being checked in (a) above, and re-executes it to check the clock status.

```
brk:0>!2 <cr>                                    ①
  CLK <cr>                                       ②
  IE                                             ③
brk:0>■
```

① Specified that the command with number 2 be called.

② <cr> is entered, and the CLK command is reexecuted.

③ Clock status is displayed.

8.4.18 Help command (HLP)

```
General format
  Format 1    HLP[ △command]
```

brk:n>	o	emu:n>	o	trc:n>	o

Function:

The HLP command displays a list of commands and how to use those commands.

Executing this command turns the system to the command input waiting status in the help mode (HLP>), and entering a command in this mode makes the system to display how to use the command.

To return the system from help mode to the normal command waiting status, enter <cr> only.

Programming:

Operand

Command: Specify the command to be displayed.
Default: Display the list of commands.

Programming note:

No command entered during the help mode is executed.

Example:

(a) Use of the Help command

This sample program displays how to use the directory command.

```
brk:0>HLP DIR <cr>                                              ①

  (Directory Dump Command)

  DIR[_d:][file name.ext]        display file directory

      d          =drive number       (omission=default drive)
                  use character A-P

      file name =file name is organized by 8 characters

      ext        =extend character is organized by 3 characters
                  use character of file name and ext
                  (0-9, A-Z, Special character omit=< >.,;:=[]

 HLP>■                                                          ②
```

① Specified to display how to use the DIR command.

② The system is turned into help mode.

(b) Displaying a list of commands

This sample program displays a list of the commands which can be used in the IE-75001-R.

```
brk:0>HLP <cr>                                                    ①

    Command Table
  ASM       BNK       BRA       BRK       BRS
  BRM       CHK       CLK       COM       CVD
  CVM       DAS       DIR       DLY       DOS
  EXT       HIS       HLP       LOD       LST
  MAT       MEM       MOD       OUT       PAS
  PGM       RAM       REG       RES       RUN
  SAV       SET       SPR       STR       STP
  STS       SYM       TRD       TRF       TRG
  TRM       TRP       TRX       TRY       VRY

HLP>■
```

① Specified to display the list of commands.

(c) Displaying the use of the command

When a command is entered in the help mode, the system displays how to use that command.

```
HLP>DIR <cr>

  (Directory Dump Command)

  DIR[_d:][file name.ext]        display file directory

      d          =drive number       (omission=default drive)
                  use character A-P

      file name =file name is organized by 8 characters

      ext        =extend character is organized by 3 characters
                  use character of file name and ext
                  (0-9, A-Z, Special character omit=< >.,;:=[]

  HLP>■
```

①

① A command is entered in the help mode.

8.4.19 Load command (LOD)

```
General format
  Format 1    LOD △file[ △ module¥----][ △ option]
```

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The LOD command downloads an object file, a debug environment file, or a symbol file from a host machine by specifying a file name.

There are following two methods of download.

Method	Description	Selection
High-speed download	Download from Centronix I/F output of a host machine to Centronics I/F input of IE (CH4)	Turn on power to the IE-75001-R, start the control program, then select high-speed download.
Normal download	Download from RS-232-C I/F channel of a host machine to RS-232-C I/F channel of IE (CH1)	Do not select high-speed down-load after turning on power to the IE-75001-R and starting the control program.

Programming:

Operands

file: File name without an extension of an object file, a symbol file, or a debug environment file.

module¥: Mode name when a symbol file is loaded.

option: Any one of C, D, or S.

option	Function
C	Load the object file specified in the operand with an extension ".HEX".
D	Load the debug environment file specified in the operand with an extension ".DBG". The following commands environment can be set by the environment file. BNK, BRA1, BRA2, BRA3, BRA4, BRK, BRM, BRS1, BRS2, CHK, CLK, DLY, MOD, OUT, PAS, PGM, REG, SET, STS, TRF, TRM, TRX, TRY
S	Load the symbol file specified in the operand with an extension ".SYM". If module name is specified after the specified file name, only the symbol file of the specified module is loaded.
Default	Perform all the processes of options C, D, and S.

Programming note:

When an object file is loaded, the storage addresses of that object file is automatically set as a coverage measurement range (See Section 8.4.11).

Example:

(a) Specifying that each file be loaded at the same time

This sample program loads an object file, a debug environment file, and symbol files at the same time.

```
brk:0>LOD SAMPLE <cr>

  Debug condition load (Y/N)? Y <cr>
  Debug condition load complete
  object load complete
  symbol table loading
  PUBLIC load complete
  MOD01 load complete
  MOD02 load complete

brk:0>■
```

(b) Specifying that the file of the specified module be loaded

This sample program loads an object file, a debug environment file, and files of the specified module.

```
brk:0>LOD SAMPLE PUBLIC¥, MOD02¥ <cr>

  Debug condition load (Y/N)? Y <cr>
  Debug condition load complete
  object load complete
  symbol table loading
  PUBLIC load complete
  MOD02 load complete

brk:0>■
```

(c) Specifying that an environment file be loaded

This sample program loads only an environment file.

```
brk:0>LOD SAMPLE D <cr>

  debug condition load complete

brk:0>■
```

8.4.20 Redirect output command (LST)

```
General formats

  Format 1    LST[ △d:file]
  Format 2    LST[ △LST:]
  Format 3    LST[ △CON:]
```

Function overview:

The LST command opens a file or list device so that command execution results can be output to the specified output device as well as to the console.

Output to an output device is started or stopped in the following procedures.

(a) Starting output to the device

Enter the ^P (CTRL + P) key while the system is in the command input waiting status. Command execution results after this line are output to the specified device.

(b) Stopping output to the device

Enter the ^P key while command execution results are output to the specified device and the system is in the command input waiting status. Output of command execution results is stopped from this line.

(1) Specifying that a file be opened

Format 1 LST[△ d:file]

brk:n>	o	emu:n>	o	trc:n>	o

Function:

Opens the file specified in the operand to create a redirect file.

If the name of the file is already present, the following processes are performed according to the specification of file attributes.

(a) When the file attributes are DIR and R/W

The system displays the following message and allows the user to select to delete the existing file and create a new redirect file or to retain the existing file unchanged.

File already exists. Delete? (Y or N):___

Y: Delete the existing file and create a new redirect file.
N: Retain the existing file unchanged.

(b) When the file attribute is SYS or R/O

The system displays the following message and ignores the command.

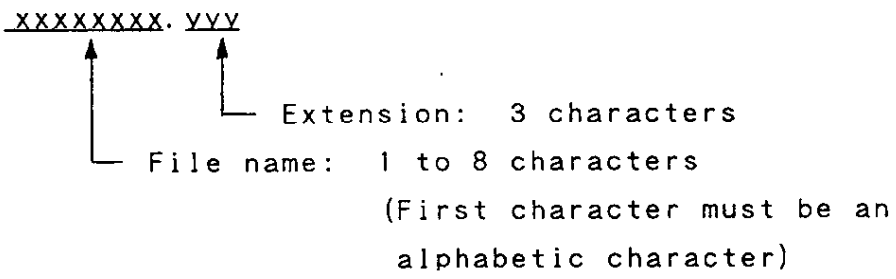
File already exists.

Programming:

Operands

Specify the number of the drive to be opened and the file name.

d: Specify the drive number.
file: Specify the file name.



(2) Specifying that a list device be opened

Format 2 LST[△ LST:]

brk:n>	o	emu:n>	o	trc:n>	o

Function:

Opens a list device specified in the operand to output command execution results.

If the list device is being used by another process, the message indicated below is displayed and the command is ignored.

List device is used by other process.

Programming:

Operand

Specify LST:.

(3) Specifying that a redirect file output device be closed

Format 3 LST[△ CON:]

brk:n>	o	emu:n>	o	trc:n>	o

Function:

Closes the opened file or the opened list output device.

Programming:

Operand

Specify CON:.

Example:

This sample program opens a file as an output device. Following lines ② to ⑦ are output to a file named SAMPLE.TXT.

```
brk:0>LST B:SAMPLE.TXT <cr>                                    ①
brk:0>MEM F 0, 1FF 1, 2, 3, 4, 5, 6, 7, 8 <cr>                 ②(*)
brk:0>MEM D 080X <cr>                                          ③
  0800  01 02 03 04 05 06 07 ...                               ④
brk:0>BRA A=100 <cr>                                           ⑤
brk:0>BRM BRA1 <cr>                                            ⑥
brk:0>TRM ALL $M <cr>                                          ⑦
brk:0>■                                                        ⑧(*)
```

① A file named SAMPLE.TXT in drive B is opened.

② Enter ^P (CTRL + P). Then, enter MEM F 0, 1FF 1, 2, 3, 4, 5, 6, 7, 8 <cr>.

③ ④ ⑤ ⑥ ⑦ ② to ⑦ and the result are output to SAMPLE.TEXT.

⑧ Enter ^P. Close SAMPLE.TXT.

* ^P is not displayed on the screen.

8.4.21 Math command (MAT)

General format

Format 1 MAT △ expression

brk:n>	o	emu:n>	o	trc:n>	o

Function:

The MAT command evaluates an expression representation coded in the operand, and displays the result in hexadecimal, decimal, octal, and binary in this order.

Programming:

Operand

expression: Allow words, bytes, and symbols to be combined by the operators indicated below.

(,)	High
*, /	↑
+, -	Priority
AND	↓
OR, XOR	Low

Programming note:

An operation is performed on a 16-bit integer basis. When an operation result is longer than 16 bits, only the low-order 16 bits are used.

Example:

This sample program illustrates an operation using the MAT command.

```
brk:0>MAT 5H+7H AND 17Q <cr>                    ①
  0CH,12T,14Q,1100Y                             ②
brk:0>■
```

① Expression is input.
② Operation results are displayed.

8.4.22 Manipulate program memory command (MEM)

General formats

Format 1 MEM △C[△ addr]

Format 2 MEM[△ D[{ △ addr / △ partition }]

Format 3 MEM △E[△ partition]

Format 4 MEM △ [F / G] △ partition △ data-string

Format 5 MEM △ [M / X / V] △ partition △ addr

Radix	addr:H partition:H data-string:H

Function overview:

The MEM command changes, displays, tests, initializes, searches, copies, exchanges, or compares the contents of the program memory in unit of 8 bits.

(1) Changing the contents of a program memory

Format 1 MEM △ C[addr]					
Radix	addr:H				
brk:n>	o	emu:n>	x	trc:n>	x

Function:

Changes contents of a program memory.

Programming:

Operand

addr: Specify the memory change start address.
No mask can be specified.

Valid address range: 0 to 0FFFFH
Default: The address at which the previous change operation ended is used.
Initial value: 0H

Example:

This sample program changes memory contents starting at address 100H.

```
brk:0>MEM C 100 <cr>                                    ①

  0100   00   11 <cr>
  0101   11   22 <cr>
  0102   22   33 <cr>
  0103   33   <cr>                                      ②
  0104   44   55 <cr>
  0105   55   66 <cr>
  0106   66   77 <cr>
  0107   77   88 <cr>
  0108   88   . <cr>                                    ③

brk:0>MEM C <cr>                                        ④

  0108   88   99 <cr>                                   ⑤
  0109   99   0AA <cr>
  010A   AA   0BB <cr>                                  ⑥
  010B   BB   0CC <cr>
  010C   CC   . <cr>                                    ⑦

brk:0>■
```

① Memory change start address

② Values are changed (with address 103 unchanged).

③ A period (. <cr>) is entered to terminate memory contents change.

④ Specification of memory change address is omitted.

⑤ Address at which the previous change ended.

⑥ Values are changed.

⑦ A period (. <cr>) is entered to terminate memory contents change.

(2) Displaying memory contents

Format 2	MEM[△D[{ △addr / △partition }]
Radix	addr:H partition:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Displays the contents of a program memory. They are displayed in hexadecimal and ASCII forms. The memory contents from 20H to 70H are displayed as indicated in the table below. 00H to 1EH and 7EH to FFH are displayed as the ASCII character period (.).

		Upper 4 bits 2	3	4	5	6	7
	0	(SP)	0	@	P	'	P
	1	!	1	A	Q	a	q
	2	"	2	B	R	b	r
	3	#	3	C	S	c	s
	4	$	4	D	T	d	t
	5	%	5	E	U	e	u
	6	&	6	F	V	f	v
Lower	7	'	7	G	W	g	w
4 bits	8	(	8	H	X	h	x
	9	)	9	I	Y	i	y
	A	*	:	J	Z	j	z
	B	+	;	K	[	k	{
	C	,	<	L	\	l	|
	D	-	=	M	]	m	}
	E	.	>	N	^	n	
	F	/	?	O	_	o	

Programming:

Operands

addr: Specify the address or the range of addresses of the contents of a program memory to be displayed.

partition: The address range is specified by mask or partitions.

Valid address range: 0 to 0FFFFH

Valid mask range: 0 to 0XXXXH

Partition specification:
Specified range is displayed.

When only start address is specified:
Contents of memory of 11 lines starting at the specified address are displayed.

Default: The contents at 11 addresses are displayed. These addresses starts from the next address after the previous one.

Initial value: 0H

Example:

This sample program displays contents of memory from address 100H.

```
brk:0>MEM D 100,17F <cr>                                              ①

  0100 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ................
  0110 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?
  0120 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F @ABCDEFGHIJKLMNO
  0130 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F PQRSTUVWXYZabcde
                                    .
                                    .
                                    .
  0170 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F !"#S%&'0*+,¯./

brk:0>MEM D 108 <cr>                                                  ②

  0108                        08 09 0A 0B 0C 0D 0E 0F          ........
  0110 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?
                                    .
                                    .
                                    .
  01A0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF ................

brk:0>MEM D <cr>                                                      ③

  01B0 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF ................
  01C0 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF ................
  01D0 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF ................
                                    .
                                    .
                                    .
  0250 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?

brk:0>■
```

① Memory display start address and end addresses are specified.

② Only memory display start address is specified.

③ No memory display address is specified.

(3) Testing a program memory

Format 3 MEM △ E[△ partition]	
Radix	partition:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Tests the program memory specified in the test range.

(a) If no test range is specified, the program memory area from address 0 to address 0FFFFH is tested.

(b) The data of the memory tested are saved after execution of memory test.

(c) When a memory test detects an error at an address, the subsequent addresses are not tested.

Programming:

Operand

partition: Specify test range of program memory.

The test range can be specified by mask or partitions.

In partition specification, the start and end addresses must be delimited with a comma (,).

Valid address range: 0 to 0FFFFH

Valid mask range: 0 to 0XXXXH

Default: Test the program memory area from 0 to 0FFFFH.

Example:

(a) This sample program tests the program memory of addresses 0H to 0FFFH.

```
brk:0>MEM E 0XXX <cr>                    ①
  Complete                               ②
brk:0>■
```

① Addresses of the memory to be tested are specified.

② Result is displayed.

(b) This sample program tests the program memory of addresses 1000H to 1FFFH and detects an error at address 152AH.

```
brk:0>MEM E 1000, 1FFF <cr>              ①
  152A                                   ②
brk:0>■
```

① Addresses of the memory to be tested are specified.

② Result is displayed.

(4) Initializing a memory

Format 4 MEM △ F △ partition △ data-string	
Radix	partition:H data-string:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Sets specified initialization data in a specified range of memory.

Programming:

Operands

partition: Specify the memory initialization range.

The test range can be specified by mask or partitions.

In partition specification, the start and end addresses must be delimited with a comma (,).

Valid address range: 0 to 0FFFFH
Valid mask range: 0 to 0XXXXH

data-string: Set initialization data.

Up to 10 data strings can be specified.
Delimit data and data with commas (,).

Data can be specified as a mask data, of which up to 10 data strings can be specified.

Example:

(a) This sample program initializes the range of 100H to 1FFH by data strings 1, 2, 3, 4, 5, 6, 7, 8, 9, and 0.

```
brk:0>MEM F 100, 1FF 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 <cr>

brk:0>MEM D 100, 1FF <cr>

  0100 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06 ................
  0110 07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02 ................
  0120 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
                                    .
                                    .
                                    .
  01E0 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00 ................
  01F0 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06 ................

brk:0>■
```

① ②

① Specified initialization range and data strings

② Specified that memory contents after initialization be displayed.

(b) This sample program initializes memory by mask data.

```
brk:0>MEM F 100, 11F 3X <cr>                                          ①

brk:0>MEM D 100, 13F <cr>                                             ②

  0100 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 1234567890123456
  0110 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 7890123456789012
  0120 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  0130 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04 ................

brk:0>■
```

① Specified initialization range and mask data

② Specified that memory contents after initialization be displayed.

Remark: As shown in this sample program, when memory is initialized using mask data 3X, low-order four bits are not changed.

(5) Searching memory contents

Format 4 MEM △ G △ partition △ data-string	
Radix	partition:H data-string:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Searches a specified range of memory for specified data strings.

(a) When a data string is detected, the address where that data string is detected is displayed.

If two or more data are detected in the search range, all addresses where they are detected are displayed.

(b) Up to 10 data strings can be specified as search data, and mask can be specified.

Programming:

Operands

partition: Specify the memory search range.

The search range can be specified with partitions or mask.

In partition specification, the start and end addresses must be delimited with a comma (,).

Valid address range: 0 to 0FFFFH
Valid mask range: 0 to 0XXXXH

data-string: Specify search data.

Up to 10 data strings can be specified.
Delimit data and data with commas (,).

Example:

This sample program searches the contents of memory 100H to 1FFH which are assumed to be as follows.

```
0100 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36
0110 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32
0120 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
0130 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04
0140 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00
0150 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06
0160 07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02
0170 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
0180 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04
0190 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00
01A0 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06
01B0 07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02
01C0 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
01D0 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04
01E0 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00
01F0 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06
```

```
brk:0>MEM G 100, 1FF 30, 31, 32, <cr>          ①

  0109
  0113                                         ②
  011D

brk:0>■
```

① Addresses 100H to 1FFH are searched for successive data strings 30, 31, and 32.

② Addresses where data string was detected

(6) Copying memory contents

Format 5 MEM Δ M Δ partition Δ addr					
Radix	partition:H addr:H				
brk:n>	o	emu:n>	x	trc:n>	x

Function:

Copies the contents of memory in a specified copy source range to a specified copy destination address and subsequent addresses.

(a) A copy source range may overlap a copy destination address.

(b) Only one address can be specified as a copy destination address.

Programming:

Operands

partition: Specify the copy source range.

The copy source range can be specified with partitions or mask.

In partition specification, the start and end addresses must be delimited with a comma (,).

Valid address range: 0 to 0FFFFH
Valid mask range: 0 to 0XXXXH

addr: Specify one copy destination address.

Valid address range: 0 to 0FFFFH

Example:

This sample program copies contents of addresses 100H to 13FH to addresses starting at 180H.

```
brk:0>MEM M 100, 13F 180 <cr>                                          ①

brk:0>MEM D 100, 1FF <cr>                                              ②

  0100 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 1234567890123456
  0110 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 7890123456789012  ③
  0120 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  0130 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04 ................
  0140 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00 ................
  0150 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06 ................
  0160 07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02 ................
  0170 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  0180 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 1234567890123456
  0190 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 7890123456789012  ④
  01A0 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  01B0 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04 ................
  01C0 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00 ................
  01D0 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06 ................
  01E0 07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02 ................
  01F0 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................

brk:0>■
```

① Copy source range and copy destination address are specified.

② Memory contents after copy are specified to be displayed, so that the user can check that the contents of memory ③ are copied to ④.

(7) Exchanging memory contents

Format 5 MEM △ X △ partition △ addr	
Radix	partition:H addr:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Exchanges the contents of a memory in a specified memory exchange range with the contents of a memory starting at a specified memory exchange target address.

Cautions 1. No overlap is allowed between an exchange range and exchange target address.

2. Only one address can be specified as an exchange target address.

Programming:

Operands

partition: Specify the memory exchange range.
The exchange range can be specified with partitions or mask.
In partition specification, the start and end addresses must be delimited with a comma (,), and no mask can be specified.
Valid address range: 0 to 0FFFFH
Valid mask range: 0 to 0XXXXH

addr: Specify one exchange target address.
Valid address range: 0 to 0FFFFH

Example:

This sample program exchanges contents of addresses 100H to 17FH which are assumed to be as follows.

```
0100 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36
0110 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32
0120 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
0130 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04
0140 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00
0150 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06
0160 07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02
0170 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
```

```
brk:0>MEM X 100, 13F 140 <cr>                                    ①

brk:0>MEM D 100, 17F <cr>                                        ②

  0100 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00 ................
  0110 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06 ................  } ③
  0120 07 08 09 00 01 02 03 04 05 06 07 08 09 00 01 02 ................
  0130 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  0140 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 1234567890123456  } ④
  0150 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 7890123456789012
  0160 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08 ................
  0170 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04 ................

brk:0>■
```

① Contents of addresses 100H to 13FH are exchanged with contents of addresses starting at 140H.

② Memory contents after exchange are specified to be displayed, so that the user can check that the contents of memory ③ are exchanged with that of ④.

(8) Comparing memory contents

Format 5 MEM △V △partition △addr					
Radix	partition:H addr:H				

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Compares the contents of a memory in a specified comparison range with the contents of a memory starting at a specified comparison target address.

Notes 1. A comparison range may overlap a comparison target address.

2. For each mismatch found, the address and data are displayed.

3. Comparison is continued until the end address of a specified comparison range is reached.

Programming:

Operands

partition: Specify the memory comparison range.

The comparison range can be specified with partitions or mask.

In partition specification, the start and end addresses must be delimited with a comma (,), and no mask can be specified.

Valid address range: 0 to 0FFFFH
Valid mask range: 0 to 0XXXXH

addr: Specify one comparison target address.

Valid address range: 0 to 0FFFFH

Example:

This sample program compares contents of a memory starting at address 100H which are assumed to be as follows.

```
0100 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 00
0110 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36
0120 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32
0130 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
0140 09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04
0150 05 06 07 08 09 00 01 02 03 04 05 06 07 08 09 01
0160 31 32 33 34 35 41 37 38 39 30 31 32 33 34 35 36
0170 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32
0180 03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08
0190 09 00 01 02 2F 04 05 06 07 08 09 00 01 02 03 04
```

```
brk:0>MEM V 100, 14F 150 <cr>                    ①

  010F   00  015F  01
  0115   36  0165  41                             ②
  0144   03  0194  2F

brk:0>■
```

① Contents of addresses 100H to 14FH are compared with addresses starting at 150H.

② Result of comparison is displayed.

8.4.23 Set channel 2 mode command (MOD)

General format

Format 1 MOD[△MODE= [CHAR / FLOW]][△BAUD= [19200 / 9600 / 4800 / 2400 / 1200 / 600 / 300]

[△LONG= [7 / 8]][△PAR= [NON / EVEN / ODD]][△STOP= [1 / 2]]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The MOD command sets the operation status of serial channel 2. If the operands are omitted, the desired operation status can be set interactively.

Initially, the status is set as follows: one-character handshaking, 9600 baud, character length of eight bits, no parity bit, and stop bit length of two bits.

Programming:

Operands

MODE: Select a handshaking mode.
CHAR: Hardware handshaking mode
FLOW: Software handshaking mode
Initial value: Software handshaking mode

BAUD: Select one of the following baud rates:
19,200, 9600, 4800, 2400, 1200, 600, and 300
Initial value: 9600

LONG: Select a character length.
7: Character length of seven bits
8: Character length of eight bits
Initial value: Character length of eight bits

PAR: Select a parity bit.
NON: No parity bit
EVEN: Even parity bit
ODD: Odd parity bit
Initial value: No parity bit

STOP: Select a stop bit length.
1: Stop bit length of one bit
2: Stop bit length of two bits
Initial value: Stop bit length of two bits

Default: Set the desired operation status interactively.

Examples:

(a) Setting the channel 2 mode

This sample program sets MODE, BAUD, LONG, PAR, and STOP.

```
brk:0>MOD MODE=CHAR BAUD=4800 LONG=8 PAR=NON STOP=2 <cr>

brk:0>■
```

①

① The operation status is set as follows: one-character handshaking, 4800 baud for the baud rate, eight bits for the character length, no parity bit, and two bits for the stop bit length.

(b) Interactively setting the channel 2 mode

Setting the desired operation status interactively when the operands are omitted

```
brk:0>MOD <cr>

  Mode Char=FLOW <cr>
  Baud 4800=9600 <cr>
  Long 8=<cr>
  Par  NON=EVEN <cr>
  Stop 2=1 <cr>

brk:0>■
```

① ② ③ ④ ⑤ ⑥

① The operation status of channel 2 is set interactively.

② The mode is changed to the buffer control mode.

③ The baud rate is changed to 9600 baud.
④ The character length remains unchanged.
⑤ Parity is changed to even parity.
⑥ The stop bit length is changed to one.

8.4.24 Set external sense clip mode command (OUT)

```
General formats
  Format 1      OUT[ △OFF]
  Format 2      OUT[ △ON △$T]
  Format 3      OUT[ △ON △BRA? [ △BRA?]]
```

Function:

The OUT command sets the external sense clip mode. The external sense clip can be used as follows:

(a) OFF

All eight pins of the external sense clip are used for input.

(b) ON $T

Seven pins (bits 1 to 7) of the external sense clip are used for input and one pin (bit 0) is used for external event output.

(c) ON BRA? [BRA?] BRA?:BRA3 or BRA4

The pins of the external sense clip are divided into two groups: bits 0 to 3 and bits 4 to 7. Each group contains four pins. Each group of pins is used for outputting data read from or written into the data memory specified by BRA3 or BRA4.

Note: The set external sense clip mode command only sets the use mode of the external sense clip. This command does not check whether each pin of the external sense clip is connected to a pin for the desired use. Carefully connect the pins of the external sense clip. The external sense clip I/O level is a TTL level.

(1) Setting the external sense clip input mode

Format 1	OUT[△ OFF]

brk:n>	o	emu:n>	o	trc:n>	x

Function:

Sets all eight pins of the external sense clip to be used for input.

(a) The input data are written by the tracer. A TRD command can be used to trace the change of data.

(b) An external signal level set at setting of the event register can be used for an event condition.

Programming:

Operand

Specify OFF.

Default: The current external sense clip mode appears.

(2) Setting the external sense clip I/O mode

Format 2	OUT[△ ON △ $T]

brk:n>	o	emu:n>	o	trc:n>	x

Function:

Sets seven pins (bits 1 to 7) of the external sense clip to be used for input and one pin (bit 0) to be used for output.

(a) The input data are written by the tracer. A TRD command can be used to trace the change of data.

(b) The output pin (bit 0) can be used for external event output. (Note)

Note: Connect a pull-up resistor because bit 0 is used for open-collector output.

Programming:

(a) Operand 1

Specify ON.

(b) Option

Specify $T.

(3) Setting the external sense clip output mode

Format 3	OUT[△ ON △ BRA? [△ BRA?]]

brk:n>	o	emu:n>	o	trc:n>	x

Function:

Divides the pins of the external sense clip into two groups: bits 0 to 3 and bits 4 to 7. Each group contains four pins. And sets each group of pins to be used for outputting data read from or written into the data memory specified by BRA3 or BRA4.

(a) The groups of pins of the external sense clip correspond to BRAs as follows:

BRA3: Bits 4 to 7
BRA4: Bits 0 to 3

(b) When BRA3 or BRA4 is specified in a command other than an OUT command, BRM, TRX, TRY, or CHK, it cannot be specified in the OUT command. Conversely, when BRA3 or BRA4 is specified in an OUT command, it cannot be specified in other commands.

Programming:

(a) Operand 1

Specify ON.

(b) Operand 2

BRA?: Specify BRA3 or BRA4. This operand must be specified.

Programming note:

Set BRA3 and/or BRA4 before executing this command.

The initial value of the external sense clip for the OUT command is the memory address specified when the command is set.

Examples:

(a) Displaying the external sense clip output mode

The current setting is displayed when the operand(s) are omitted.

```
brk:0>OUT <cr>
  OFF

brk:0>OUT <cr>
  ON $T

brk:0>OUT <cr>
  ON BRA3
```

① ② ③

① When OFF is set
② When ON and $T are set
③ When ON and BRA3 are set

(b) Specifying the external sense clip mode

These sample programs set the input, I/O, and output modes.

```
brk:0>OUT OFF <cr>                                          ①
  ***WARNING***
  ALL BIT INPUT MODE

brk:0>OUT ON $T <cr>                                        ②
  ***WARNING***
  BIT0 OUTPUT MODE

brk:0>OUT ON BRA3 <cr>                                      ③
  ***WARNING***
  BIT4-7 OUTPUT MODE

brk:0>OUT ON BRA4 <cr>                                      ④
  ***WARNING***
  BIT0-3 OUTPUT MODE

brk:0>OUT ON BRA3 BRA4 <cr>                                 ⑤
  ***WARNING***
  ALL BIT OUTPUT MODE

brk:0>OUT ON BRA4 <cr>                                      ⑥
  USE BRA4 OTHERS COMMAND (BRM, TRY, CHIC)
```

① When the input mode is specified
② When the I/O mode is specified
③ When BRA3 is specified in the output mode
④ When BRA4 is specified in the output mode
⑤ When BRA3 and BRA4 are specified in the output mode
⑥ When BRA4 is already specified in another command

8.4.25 Set pass count command (PAS)

General format Format 1 PAS[△ count]	
Radix	count:T

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The PAS command sets the pass count of event detection integrated by the event mode register (BRM). If the operand is omitted, this command displays the current setting.

Programming:

Operands

count: Specify the pass count with a decimal number.
Valid pass count range: 1 to 255
Initial value: 1

Default: The current setting appears.

Example:

This sample program sets the pass count to 5.

```
brk:0>PAS 5 <cr>                    ①
brk:0>PAS <cr>                      ②
  5                                 ③
brk:0>■
```

① The pass count is specified.
② The pass count is omitted.
③ The current setting appears.

8.4.26 Set terminal mode command (PGM)

```
General formats
  Format 1      PGM
  Format 2      PGM △C
```

Function overview:

The PGM command sets the terminal mode for remote operation of a PROM programmer, the PG-1000 or PG-1500, connected to channel 2 of the IE-75001-R.

Remark: Connecting the PG series programmer using the MOD command (See Section 4.2 and refer to the User's manual for each PG series programmer.)

(1) Connection with the PG-1500

① Connect between channel 2 (CH2) of the IE-75001-R and the PG-1500 with the RS-232-C straight cable.

② Specify the RS-232-C interface for channel 2 of the IE-75001-R using the MOD command and for the PG-1500 using the MODE key.

③ Open the cover for the RS-232-C mode setting section at the side of the IE-75001-R and set the slide switch for selecting a CH2 modem or terminal mode to the terminal mode position.

④ Perform the operations on the PG-1500 panel in the following key order:

(Power on) → RESET → FUNCTION → REMOTE → SET

(2) Connection with the PG-1000

① Connect between channel 2 (CH2) of the IE-75001-R and the PG-1000 with the RS-232-C straight cable.

② Specify the RS-232-C interface for channel 2 of the IE-75001-R using the MOD command and for the PG-1000 using the MODE key.

③ Open the cover for the RS-232-C mode setting section at the side of the IE-75001-R and set the slide switch for selecting a CH2 modem or terminal mode to the terminal mode position.

④ Perform the operations on the PG-1000 panel in the following key order:

(Power on) → RESET → DEVICE SELECT → A-ENT → SERIAL

(1) Setting the terminal mode

Format 1 PGM

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Controls communication between channel 2 of the IE-75001-R and a PROM programmer, PG-1000 or PG-1500.

When a PGM command is executed, the terminal enters the PGM mode. In the PGM mode, an object code can be uploaded to and downloaded from the PG-1000 or PG-1500 connected to channel 2.

Programming:

Specify the command only.

Examples:

(a) Connecting/disconnecting the PG-1500

```
brk:0>PGM <cr>                                    ①
  Beginning of PGM mode
PG>                                               ②
  .
  .
  .
PG>^[CTRL] Z                                      ③
Exit PGM mode (Y/N)Y <cr>                         ④
brk:0>■                                           ⑤
```

① Connect the PG-1500 using the PGM command.

② Enter desired commands for the PG-1500 (Note) in the PG-1500 connection mode (prompt PG>).

③ and ④ Entering ^Z (CTRL + Z) and Y <cr> disconnects the PG-1500.

⑤ The emulator exits the PGM mode.

Note: The following operator commands can be used when the PG-1500 is connected.

Table 8-3 Commands for the PG-1500

Command	Format	Function
RR	PG>RR R_S_ADR,R_E_ADR,PG_S_ADR,CONV <cr>	Reads PROM contents.
RS	PG>RS sub <cr> sub=C/R/A	Selects a device.
RV	PG>RV R_S_ADR,R_E_ADR,PG_S_ADR,CONV <cr>	Compares PROM with memory in PG-1500.
RW	PG>RW R_S_ADR,R_E_ADR,PG_S_ADR,CONV <cr>	Writes on PROM.
RZ	PG>RZ <cr>	Checks whether PROM contents are erased.
MC	PG>MC PG_S_ADR <cr>	Changes contents of memory in PG-1500.
MD	PG>MD PG_S_ADR,PG_E_ADR <cr>	Displays contents of memory in PG-1500.
MF	PG>MF PG_S_ADR,PG_E_ADR,INT_DATA <cr>	Initializes memory in PG-1500.
LI	PG>LI <cr>	Transmits data from IE-75001-R to PG-1500.
SI	PG>SI PG_S_ADR,PG_E_ADR <cr>	Transmits data from PG-1500 to IE-75001-R.
??	PG>?? <cr>	Command help

For details, refer to "PG-1500 User's Manual."

Remark:
- PG_S_ADR: PG-1500 start address
- PG_E_ADR: PG-1500 end address
- R_S_ADR: PROM start address
- R_E_ADR: PROM end address
- CONV: Specification of address division
- INT_DATA: Initialization data

(b) Uploading object codes to the PG-1500 (LI command)

The contents of mapped memory in IE-75001-R are transferred to the buffer in the PG-1500. If the transfer range is omitted, all the contents of the mapped memory are transferred. To stop data transfer, press the ESC key.

```
PG>LI <cr>

 Partition=YYYY, ZZZZ <cr>
PG>■
```

①

① Enter the transfer start address for memory in the IE-75001-R in the YYYY field and the transfer end address for memory in the IE-75001-R in the ZZZZ field.

(c) Downloading object codes from the PG-1500 (SI command)

The contents of the buffer in the PG-1500 are transferred to mapped memory in the IE-75001-R. The load bias of the IE-75001-R cannot be omitted. To stop data transfer, press the ESC key.

```
PG>SI XXXX, YYYY <cr>

 Bias=ZZZZ <cr>
   complete
PG>■
```

①

②

① Enter the transfer start address for the buffer in the PG-1500 in the XXXX field and the transfer end address for the buffer in the PG-1500 in the YYYY field.

② Enter the load bias of the IE-75001-R in the ZZZZ field.

(d) Connecting/disconnecting the PG-1000

```
brk:O>PGM <cr>                                    ①
  Beginning of PGM mode
*I <cr>                                           ②
*■                                                ③
 .
 .
* CTRL Z                                          ④
Exit PGM mode (Y/N)Y <cr>                         ⑤
brk:O>■                                           ⑥
```

① Connect the PG-1000 using the PGM command.

② Entering I <cr> enters the intelligent mode. Letter I does not appear, however.

③ Enter desired commands for the PG-1000 (Note) in this mode (prompt *).

④ and ⑤ Entering ^Z (CTRL + Z) and Y <cr> disconnects the PG-1000.

⑥ The emulator exits from the PGM mode.

Note: The following operator commands can be used when the PG-1000 is connected.

Table 8-4 Commands for the PG-1000

Command	Format	Function
A	*As,e,r <cr>	Sets parameters.
E	*Er <cr>	Changes the content of the buffer in the PG-1000. (The format is as follows.) *Er <cr> r XX- YY- XX- YY- XX- <cr>  Data input format - Enter data. (When non-hexadecimal data is entered, a question mark (?) appears.) - Press the space key if the data need not be changed.)
F	*Fr,re,d <cr>	Initializes the buffer in the PG-1000 with d.
I	*I <cr>	Enters the intelligent mode (for echo back to the display).
O	*Or,re <cr>	Displays the contents of the buffer in the PG-1000. Display format: r 00 00 00 00 00 00 00
R	*Rr,re <cr>	Transfers PROM contents to the buffer in the PG-1000.
S	*S <cr>	Select PROM
T	*T <cr>	Enters the transient mode (for no echo back to the display).
V	*Vs,e,r <cr>	Compares PROM contents with PG-1000 buffer contents. A question mark (?) appears in case of data mismatch.
W	*Ws,e,r <cr>	Writes PG-1000 buffer contents into PROM. A question mark (?) appears in case of a writing error.
Y	*Y <cr>	Displays current parameters.
Z	*Z <cr>	Checks whether data is already written on PROM. A question mark (?) appears only if data is already written. (Nothing appears unless data is already written.)
L	*L bias <cr> partition= i,ie <cr>	Transfer data at addresses i to ie mapped in IE-75001-R to addresses i+bias to ie+bias in the buffer of the PG-1000.
P	*Pr,re <cr> Bias=i <cr>	Transfer data mapped at addresses r to re in the PG-1000 buffer to addresses r+i to re+1 mapped in the IE-75001-R.

For details, refer to "PG-1000 User's Manual."

Remark:
- s: PROM start address
- e: PROM end address
- r: PG-1000 buffer start address
- re: PG-1000 buffer end address
- i: IE-75001-R start address
- ie: IE-75001-R end address
- d: Data

Note: Conditions causing an error

s > e

r > re

i > ie

Non-hexadecimal data entry

(e) Uploading object codes to the PG-1000 (L command)

The contents of mapped memory in IE-75001-R are transferred to the buffer in the PG-1000. If the transfer range is omitted, all the contents of the mapped memory are transferred. To stop data transfer, press the ESC key.

```
*LXXX <cr>                                    ①
 Partition=YYYY, ZZZZ <cr>                    ②
*■
```

① Enter the load bias of the IE-75001-R in the XXX field.

② Enter the transfer start address for memory in the IE-75001-R in the YYYY field and the transfer end address for memory in the IE-75001-R in the ZZZZ field.

(f) Downloading object codes from the PG-1000 (P command)

The contents of the buffer in the PG-1000 are transferred to mapped memory in the IE-75001-R. The load bias of the IE-75001-R cannot be omitted. To stop data transfer, press the ESC key.

```
*PXXXX, YYYY <cr>                    ①

 Bias=ZZZZ <cr>                      ②
   complete
*■
```

① Enter the transfer start address for the buffer in the PG-1000 in the XXXX field and the transfer end address for the buffer in the PG-1000 in the YYYY field.

② Enter the load bias of the IE-75001-R in the ZZZZ field.

(2) Changing the current control characters

```
Format 2      PGM △C
```

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Enables the control characters in the PGM mode to be changed to the desired characters interactively.

(a) The following 16 characters are available:

^A (CTRL + A (01H))
^B (CTRL + B (02H))
^E (CTRL + E (05H))
^F (CTRL + F (06H))
^G (CTRL + G (07H))
^N (CTRL + N (0EH))
^O (CTRL + O (0FH))
^P (CTRL + P (10H))
^R (CTRL + R (12H))
^T (CTRL + T (14H))
^U (CTRL + U (15H))
^V (CTRL + V (16H))
^W (CTRL + W (17H))
^X (CTRL + X (18H))
^Y (CTRL + Y (19H))
^Z (CTRL + Z (1AH))

(b) When the control characters are changed to the valid characters, the terminal enters the PGM mode. There is no need to execute a PGM command after change of control characters.

(c) A character can be set for the control character of only one function.

(d) When the <DEL> key or ^H is entered, a control character is changed to its default value.

(e) When the <ESC> key is entered to cause an interrupt, the character change made until that time is invalidated.

Programming:

Operand

Specify C.

Examples:

(a) Example of changing characters

This paragraph describes how to change the character and how to use the DEL key.

```
brk:0>PGM C <cr>

  Termination of "PGM"          ... ^Z
  Beginning   of "HEX LOAD"     ... ^A
  Beginning   of "HEX SAVE"     ... ^E
  Termination of "SYM LOAD"     ... ^N
  Termination of "LOAD"         ... ^B
  Termination of "SAVE"         ... ^F
  Break       of "LOAD/SAVE"    ... ^W

  Termination of "PGM"          ... ^Z

  Beginning   of PGM mode
```

① ② ③ ④

① Change current control character command

② The default values appear.

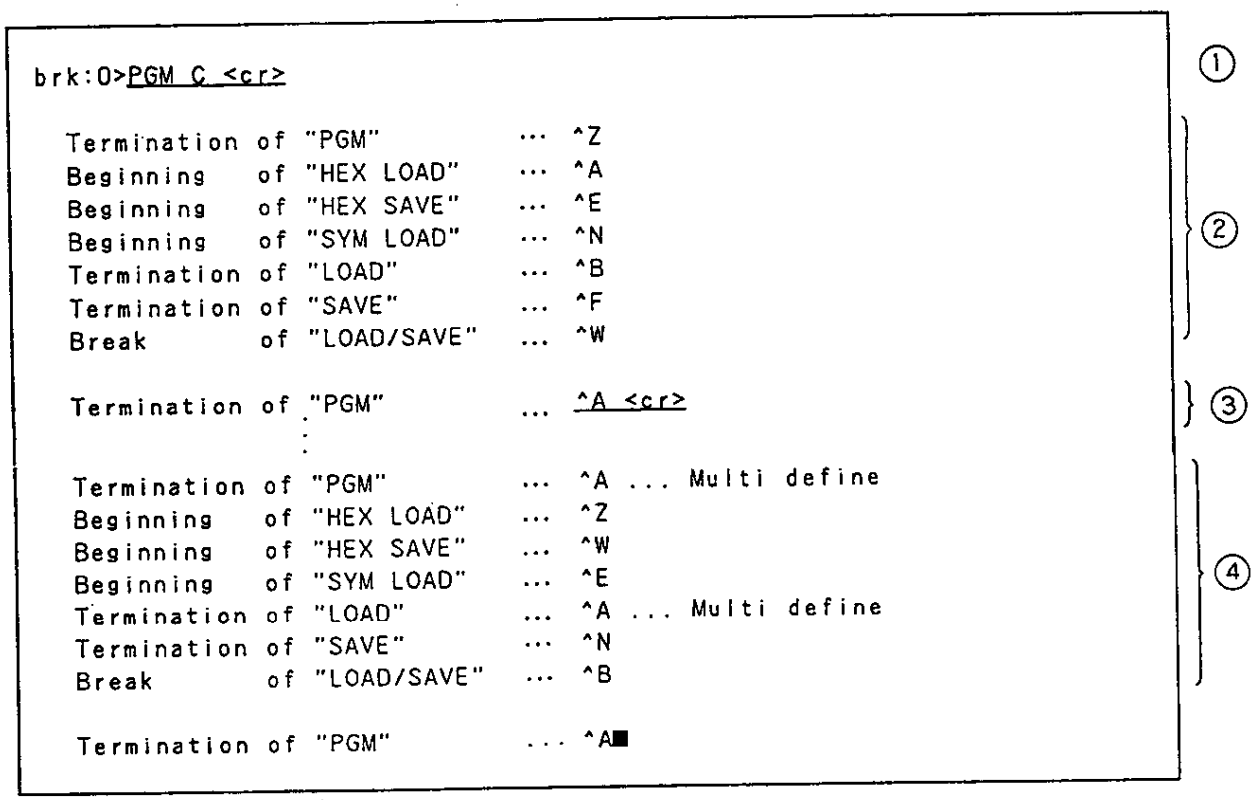
③ Displays the currently set value of Termination of "PGM" and waits for input. To change the setting, hit CTRL key the character key to be set in pressing, and then hit the carriage return key. To restore the initial setting, hit DEL key (or "CTRL + H") followed by carriage return key. Then the newly set character is displayed on the screen. If the current setting is to be left unchanged, only input carriage return. When the setting of Termination of "PGM" is finished, the next Beginning of "HEX LOAD" is

displayed on the bottom line on the screen. Set this line in the same manner as the first line. When the setting of Beginning of "HEX LOAD" is finished, the next Beginning of "HEX SAVE" is displayed on the bottom line of the screen. In this manner, after one setting has been finished, the next line is displayed. Subsequently, set Termination of "SYM LOAD", Termination of "LOAD", Termination of "SAVE", and Break of "LOAD/SAVE".

④ If there is no problem in connection with input, the message Beginning of PGM mode is displayed, and the PGM is set.

Caution: If setting 3 is aborted by pressing the ESC key, the changed data becomes invalid.

(b) In this example, an attempt is made to set a character for two or more control characters by mistake.



① Change current control character command

② The default values appear.

③ Setting of control character (Refer to preceding page.)

④ If an erroneous character has been set (in this example, "CTRL + A" is input at two places), the PGM mode is not set, but the in-circuit emulator enters in the input wait status. Correct the setting.

8.4.27 Manipulate data memory command (RAM)

General formats

Format 1 RAM △C[{ △addr[{ △$B / △$N }] / △addrb }]

Format 2 RAM[△D[{ { △addr / △partition } { △B / △N } / △addrb }]]

Format 3 RAM △E[△partition]

Format 4 RAM { △F / △G } △partition △data-string[{ △$B / △$N }]

Format 5 RAM { △M / △X / △V } △partition △addr[{ △$B / △$N }]

Format 6 RAM △L

Format 7 RAM △S[△partition] [,partition]···[,partition]

Radix	addr:H addrb:H partition:H data-string:H

Function overview:

The RAM command changes, displays, tests, initializes, searches, copies, exchanges, and compares data in data memory and loads a file into data memory and saves data in data memory into the file.

Programming notes:

(a) The address range which can be specified in an RAM command depends on the target devices. A device to be debugged is selected using an STS command.

(b) When the emulated device is reset by an RES command, data in data memory become undefined. When the target device is reset, the data also become undefined. Carefully execute an RES command.

(1) Changing data in data memory

Format 1 RAM △C[{ △ addr { △$B / △$N } / △ addrb }]					
Radix	addr:H addrb:H				
brk:n>	o	emu:n>	x	trc:n>	x

Function:

Changes data in the area starting at the address specified for the change start address in data memory in byte or nibble units, whichever is specified. Data in data memory can also be changed bit-by-bit.

Programming:

Operands

addr: Specify the change start address of data in data memory.

A partition or mask cannot be used in specifying the address.

Valid address range: Depends on the target devices.
Default: The value specified just before is used.
Initial value: 0H

$B: Changes data in byte units.
In this case, only even addresses can be specified.

$N: Changes data in nibble units. (Default)

addrb: Specify the address and bit number of data in data memory to be changed.

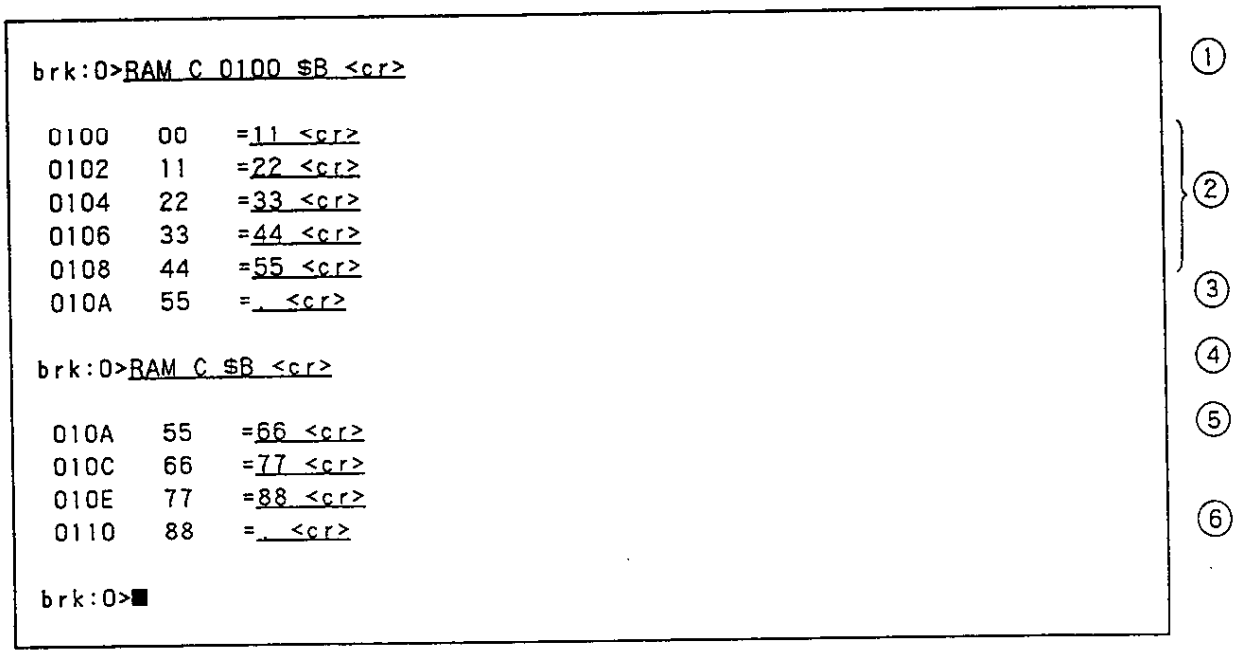
A bit attribute symbol cannot be specified.

Format: address.bit-number
Address: 0 to 0FFFH
Bit: 0 to 3

Examples:

(a) Changing data in data memory in byte units

This sample program changes data in the area starting at address 0100H in data memory in byte units.



① A change data-in-memory command is specified.

② Change values are entered.

③ A period (. <cr>) is entered to terminate change of data in memory.

④ The change start address is omitted.

⑤ Address of data specified just before (Address 0 at power-on)

⑥ A period (. <cr>) is entered to terminate change of data in memory.

(b) Changing data in data memory in nibble units

This sample program changes data in the area starting at address 0100H in data memory in nibble units.

```
brk:0>RAM C 0100 $N <cr>                                    ①

 0100      1  =2 <cr>
 0101      1  =2 <cr>
 0102      2  =3 <cr>                                       ②
 0103      2  =3 <cr>
 0104      3  =4 <cr>
 0105      3  =. <cr>                                       ③

brk:0>RAM C $B <cr>                                         ④

 0105      3  =4 <cr>                                       ⑤
 0106      4  =5 <cr>
 0107      4  =5 <cr>
 0108      5  =. <cr>                                       ⑥

brk:0>■
```

① An RAM command is specified.
② Change values are entered.
③ A period (. <cr>) is entered to terminate change of data in memory.
④ The change start address and option are omitted.
⑤ Address of data specified just before (Address 0 at power-on)
⑥ A period (. <cr>) is entered to terminate change of data in memory.

(c) Changing data at a specified address bit-by-bit

This sample program changes the value of bit 1 at address 0100H in data memory.

```
brk:0>RAM C 100.1 <cr>                    ①
  0100.1  1   =   0 <cr>                  ②
brk:0>■
```

① An RAM command is specified.

② The change value is entered.

(2) Displaying data in data memory

Format 2 RAM [△D [{ { △ addr / △ partition } [{ △ $B / △ $N }] / △ addrb }]]

Radix	addr:H	partition:H	addrb:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Displays data in a specified address range in byte or nibble units, whichever is specified. Data in data memory can also be displayed bit-by-bit.

Programming:

Operands

addr partition:

Specify the address or address range of data in data memory to be displayed.

A partition or mask can be used in specifying the address range.

When a partition is used, separate the start and end addresses with a comma (,).

Valid address range:
Depends on the target devices.

When only the start address is specified:
Eleven lines of data from the specified address are displayed.

Default: Eleven lines of data from the address following that specified just before are displayed.

Initial value: 0H

$B: Displays data in byte units.
In this case, only even addresses can be specified.

$N: Displays data in nibble units. (Default)

addrb: Specify the address and bit number of data in data memory to be displayed.

A bit attribute symbol cannot be specified.

Format: address.bit-number
The valid address range depends on the target devices.
Bit: 0 to 3

Examples:

(a) Displaying data in data memory in byte units

This sample program displays data in the area starting at address 0100H in data memory in byte units.

```
brk:0>RAM D 100,17F $B <cr>

 ADDR   +0 +2 +4 +6 +8 +A +C +E +10+12+14+16+18+1A+1C+1E
 0100    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F .........
 0120    ......
  .
  .
 0160    ......

brk:0>■
```

(b) Displaying data in data memory in nibble units

This sample program displays data in the area starting at address 0100H in data memory in nibble units.

```
brk:0>RAM D 100,17F $N <cr>

 ADDR     +0+1+2+3+4+5+6+7+8+9+A+B+C+D+E+F
 0100      0 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0
 0110      ........
 0120
 0130
 0140
 0150
 0160
 0170      ........

brk:0>■
```

(c) Displaying data at a specified address bit-by-bit

This sample program displays the value of bit 1 at address 0100H in data memory.

```
brk:0>RAM D 100. 1 <cr>
  0100. 1 0
brk:0>■
```

(3) Testing data in data memory

Format 3 RAM △ E[△ partition]	
Radix	partition:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Tests data in a specified test range in data memory.

When the test range is omitted, this command tests data in all data memory area of the target device.

Programming:

Operand

partition: Specify the test range of data in data memory.

A partition or mask can be used in specifying the test range.

When a partition is used, separate the start and end addresses with a comma (,).

Valid address range:
Depends on the target devices.

Default: Tests data in all data memory area of the target device.

Initial value: 0H

Programming notes:

(a) The tested data in memory are saved after memory testing.

(b) If an error is detected during memory testing, data at the address where the error is detected and subsequent addresses are not tested.

Examples:

(a) This sample program tests data at addresses 0H to 100H in data memory.

```
brk:0>RAM E0, 100 <cr>

  Complete

brk:0>■
```

(b) After a partition is omitted, data in all data memory area of the target device are tested.

```
brk:0>RAM E <cr>                                    ①
  110                                               ②
brk:0>■
```

① Data in all data memory of the target device are tested.

② The address at which an error is detected is displayed.

Remark: A test stops at the address at which an error is detected.

(4) Initializing data in data memory

Format 4	RAM △F △partition △string-data { △$B / △$N }

Radix	partition:H	data-string:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Initializes data memory in a specified address range to a specified data string in specified units.

Programming:

Operands

partition: Specify the initialize range of data memory.

A partition or mask can be used in specifying the initialize range.

When a partition is used, separate the start and end addresses with a comma (,).

Valid address range:
Depends on the target device.

data-string: Specify initialize data.

Up to 10 data items can be specified.
Separate the data items with a comma (,).

Mask data can be specified.

$B: Initializes data memory in the address range in byte units.

In this case, only even addresses can be specified.

$N: Initializes data memory in the address range in nibble units. (Default)

In this case, a mask cannot be used in specifying initialize data.

Example:

This sample program initializes addresses 100H to 1FFH in data memory to data strings 0 to 9.

```
brk:0>RAM F 100, 1FF 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, <cr>
brk:0>■
```

①

① partition: Addresses 100H to 1FFH
data-string: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Units: Omitted. (Nibble)

(5) Searching data memory

Format 4 RAM △G △partition △data-string[{ △$B / △$N }]

Radix	partition:H	data-string:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Searches a specified range of data memory for a data string in specified units.

(a) When this command detects the data string, it displays the address(es) at which the data string is detected.

(b) Up to 10 search data items can be specified. A mask can be used only when $B is specified.

Programming:

Operands

partition: Specify the search range of data memory.

A partition or mask can be used in specifying the search range.

When a partition is used, separate the start and end addresses with a comma (,).

Valid address range:
Depends on the target device.

data-string: Specify up to 10 data items. Separate the data items with a comma (,).

Mask data can be specified only when $B is specified.

$B: Searches the address range in byte units.
In this case, only even addresses can be specified.

$N: Searches the address range in nibble units. (Default)

Example:

This sample program searches addresses 100H to 1FFH in data memory for data strings 0 to 9.

```
brk:0>RAM G 100 1FF 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 <cr>

  0109
  0113
  011D

 brk:0>■
```

① ②

① partition: Addresses 100H to 1FFH
data: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Units: Omitted. (Nibble)

② The addresses at which the data string is detected are displayed.

(6) Copying data in data memory

Format 5 RAM △M △partition △addr[{△$B / △$N}]

Radix	partition:H	addr:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Copies data in a specified copy source range into the area starting at a copy destination address in specified units.

(a) An address in the copy source range can be specified for the copy destination address.

(b) Only one address can be specified for the copy destination address.

Programming:

Operands

partition: Specify the copy source range.

A partition or mask can be used in specifying the copy source range.

When a partition is used, separate the start and end addresses with a comma (,).

Valid address range:
Depends on the target devices.

addr: Specify a copy destination address.
Valid address range: Depends on the target devices.

$B: Copies data in byte units.
In this case, only even addresses can be specified.

$N: Copies data in nibble units. (Default)

Example:

This sample program copies data at addresses 100H to 13FH in data memory into the area starting at address 180H.

```
brk:0>RAM M 100, 13F 180 <cr>
brk:0>■
```

①

① partition: Addresses 100H to 13FH
addr: Address 180H
Units: Omitted. (Nibble)

(7) Exchanging data between two areas in data memory

Format 5 RAM △X △partition △addr [{△$B / △$N}]
Radix | partition:H addr:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Exchanges data between a specified exchange range and the area starting at a specified exchange destination address.

Notes 1. An address in the exchange range cannot be specified for the exchange destination address.

2. Only one address can be specified for the exchange destination address.

Programming:

Operands

partition: Specify the exchange range.

A partition or mask can be used in specifying the exchange range.

When a partition is used, separate the start and end addresses with a comma (,).

Valid address range:
Depends on the target devices.

addr: Specify an exchange destination address.
Valid address range: Depends on the target devices.

$B: Exchanges data in byte units.
In this case, only even addresses can be specified.

$N: Exchanges data in nibble units. (Default)

Example:

This sample program exchanges data between the area at addresses 100H to 1FFH and the area starting at address 200H in data memory.

```
brk:0>RAM X 100, 1FF 200 <cr>
```

①

① partition: Addresses 100H to 1FFH
addr: Address 200H
Units: Omitted. (Nibble)

(8) Comparing data in data memory

Format 5 RAM △ V △ partition △ addr [△ $B / △ $N]

Radix	partition:H addr:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Compares data in a specified comparison range with data in the area starting at a specified comparison destination address.

(a) An address in the comparison range can be specified for the comparison destination address.

(b) When a mismatch is found by the comparison, the differing datum and its address are displayed.

(c) The comparison continues to the end of the comparison range.

Programming:

Operands

partition: Specify the comparison range in data memory.

A partition or mask can be used in specifying the comparison range.

When a partition is used, separate the start and end addresses with a comma (,).

The valid address range depends on the target devices.

addr: Specify a comparison destination address.

A partition or mask cannot be used.

The valid address range depends on the target devices.

$B: Compares data in byte units.
In this case, only even addresses can be specified.

$N: Compares data in nibble units. (Default)

Example:

This sample program compares data at addresses 100H to 14FH with data in the area starting at address 150H in data memory.

```
brk:0>RAM V 100, 14F 150 <cr>

  010F 00 015F 01
  0115 36 0165 41
  0144 03 0194 2F

brk:0>■
```

① ②

① partition: Addresses 100H to 14FH
addr: Address 150H
Units: Omitted. (Nibble)

② Differing data items and their addresses found by the comparison are displayed.

(9) Loading a file into data memory

Format 6 RAM ΔL

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Loads file IE75000.RAM in the current drive into data memory.

(a) The file is loaded in the area specified by an RAM S command.

(b) If file IE75000.RAM is not found in the current drive or if the file format is invalid, an error occurs.

Example:

This sample program loads the file normally.

```
brk:0>RAM L <cr>                                    ①
  Object load complete                              ②
brk:0>■
```

① The file is loaded into data memory.
② The terminated state is displayed.

(10) Saving data in data memory into a file

Format 7 RAM △S [△partition][,partition][,partition][,partition][,partition][,partition]	
Radix	partition:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Saves data in up to five save ranges into the current drive with file name IE75000.RAM.

(a) When the address is omitted, all data in data memory are saved.

(b) When file IE75000.RAM already exists in the current drive, the following message is displayed.

File already exists. Delete? (Y or N) :
Y: The file is deleted and a new file is opened.
N: The file remains unchanged.

Programming:

Operand

partition: Specify an address range of data in data memory to be saved.

A partition or mask can be used in specifying the address range.

When a partition is used, separate the start and end addresses with a comma (,).

Valid address range:
Depends on the target devices.

Examples:

(a) This sample program saves data at addresses 0H to 1FFH in data memory.

```
brk:O>RAM S 0, 1FF <cr>                    ①
  Object save complete                     ②
 brk:O>■
```

① Data in data memory are saved.
② The terminated state is displayed.

(b) After the operand is omitted, all data memory area of the target device is saved.

```
brk:O>RAM <cr>                             ①
  File already exists. Delete? (Y or N) :Y <cr>   ②
  Object save complete
brk:O>■                                    ③
```

① The operand is omitted.
② Y <cr> is entered.
This operation is required when file IE75000.RAM already exists in the current drive.
③ The terminated state is displayed.

8.4.28 Manipulate general register command (REG)

```
General formats
  Format 1   REG △C[ △register]
  Format 2   REG[△ D[ { △ ALL      } ]]
                      { △ register }
```

Function overview:

The REG command changes and displays the contents of general registers and control registers. Each flag in the PSW can be changed and displayed using its name. The registers which can be changed and displayed depend on the target devices.

(1) Changing the contents of a register

Format 1 REG △ C[△ register]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Changes the contents of general registers and control registers.

The contents of a specified register in the register bank pointed by the register bank selector (RBS) are changed.

(a) The contents of a specified general register or control register can be changed.

When PSW is specified, individual flags can be changed sequentially from the first flag. When the name of a flag is specified, only the flag is changed.

(b) When the register name is omitted, the contents of the general registers and control registers can be changed sequentially.

Programming:

Operand

register: Specify one of the following registers and flags.

Control registers:	PC, SP, PSW, RBS, and MBS
General registers:	XA, HL, DE, BC, XA', HL', DE', BC', X, A, H, L, D, E, B, and C
PSW flags:	CY, RBE, MBE, IST0, and IST1

Default: The contents of the general registers and control registers can be changed sequentially.

Examples:

(a) Changing register contents

When the register name is omitted, the contents of the general registers and control registers are changed sequentially.

```
brk:0>REG C <cr>

  XA     00 = 11 <cr>
  HL     11 = 22 <cr>
  DE     22 = 33 <cr>
  BC     33 = <cr>
  XA'    44 = <cr>
  HL'    55 = <cr>
  DE'    66 = <cr>
  BC'    77 = <cr>
  RBS     0 = <cr>
  MBS     0 = <cr>(*)
  SP    000 = <cr>
  PC   0000 = <cr>

brk:0>■
```

① ② ③

① The contents of all general registers and control registers other than the PSW are changed.

② The contents of the registers are changed.

③ The contents of the registers remain unchanged.

* Whether to use a 3- or 2-byte is determined according to the target devices.

(b) Specifying the PSW

When control register PSW is specified, each flag in the PSW can be changed.

```
brk:0>REG C PSW <cr>                                    ①

  RBE  0 = 1 <cr>                                       ②
  MBE  0 = 1 <cr>
  CY   0 = 1 <cr>
  IST1 0 = <cr>                                         ③
  IST0 0 = <cr>

brk:0>■
```

① All flags in the PSW are changed.
② The values of the flags are changed.
③ The values of the flags remain unchanged.

(c) When a register name is specified

When the name of a general or control register is specified, the contents of the register can be changed.

```
brk:0>REG C DE <cr>                                     ①

  DE 00=11 <cr>

brk:0>■
```

① The contents of register DE are changed.

(2) Displaying the contents of a register

Format 2 REG[△ D[{ △ ALL / △ register }]]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Displays the contents of general registers and control registers.

The contents of a specified register in the current register bank are displayed.

(a) When ALL is specified for the operand, the contents of all general registers, control registers, and PSW flags in all register banks for the target device are displayed.

(b) When PSW is specified for the operand, the values of the PSW flags are displayed.

(c) When the operand is omitted, the contents of all general registers, control registers, and PSW flags in the current register bank are displayed.

Programming:

Operands

ALL: Displays the contents of all general registers, control registers, and PSW flags in all register banks for the target device.

register: Specify one of the following registers and flags.

Control registers:	PC, SP, PSW, RBS, and MBS
General registers:	XA, HL, DE, BC, XA', HL', DE', BC', X, A, H, L, D, E, B, and C
PSW flags:	CY, RBE, MBE, IST0, and IST1

Default: Displays the contents of all general registers, control registers, and PSW flags in the current register bank.

Examples:

(a) When the operand is omitted

The contents of all general registers, control registers, and PSW flags in the current register bank are displayed.

```
brk:O>REG D <cr>

  XA HL DE BC XA' HL' DE' BC' RBS MBS RBE MBE CY IST1 IST0 SP   PC
  00 00 00 00 00  00  00  00   0   0   0   0  0   0     0  000 0000

brk:O>■
```

Remark: -- is displayed for the contents of a register not provided for the target device.

(b) When PSW is specified for the register name

The values of all PSW flags are displayed.

```
brk:0>REG D PSW <cr>

  RBE MBE CY IST1 IST0
   0   0  0    0    0

brk:0>■
```

(c) When a register name is specified for the operand

```
brk:0>REG D PC <cr>

  PC 0000

brk:0>■
```

(d) When ALL is specified for the operand

The contents of all general registers, control registers, and PSW flags in all register banks for the target device are displayed.

```
brk:0>REG D ALL <cr>

  RBS MBS RBE MBE CY IST1 ISTO SP   PC
   0   0   0   0  0    0    0  000 0000

       XA HL DE BC XA' HL' DE' BC'
  RB0  00 00 00 00 00  00  00  00
  RB1  00 00 00 00 00  00  00  00
    .
    .
    .
  RB14 00 00 00 00 00  00  00  00
  RB15 00 00 00 00 00  00  00  00
```

① ②

① The contents of all registers in all register banks are specified to display.

② The display contents depend on the target devices.

8.4.29 Reset command (RES)

General format

Format 1 RES[△ H]

brk:n>	o	emu:n>	o	trc:n>	o

Function:

The RES command resets the emulation device or the entire IE-75001-R system.

Programming:

Operand

H: Resets the entire IE-75001-R system.
Default: Resets the emulation device.

Examples:

(a) After H is specified for the operand, the entire IE-75001-R system is reset.

```
brk:0>RES H <cr>                                                    ①

 IE-75000/1-R Monitor V1.4 [1 Aug 92]
   Copyright (C) 1989, 1992 by NEC Corporation

Self check ok

Target CPU      uPD75108/108A/108F/P108/P108B/112/112F/116/116F/P116   ②
Program Memory  0-FFFFH
Data Memory     00H-1FFH, F80H-FFFH
Memory Bank     0-1, 15
Register Bank   0-3
Power on target system (Y/N) Y <cr>
 Do you use high speed down load mode? (Y/N)=Y <cr>                 ③
```

① The entire IE-75001-R system is reset.
② The target device is displayed.
③ The high-speed download mode is used.

(b) Resetting the emulation device

When the operand is omitted, the emulation device is reset.

```
brk:0>RES <cr>                                                      ①


brk:0>■
```

① The operand is omitted.

8.4.30 Run emulation command (RUN)

General formats	
Format 1	RUN △N[△addr]
Format 2	RUN △B[△addr]
Format 3	RUN △T[△addr] { ,expression / ,word } [△TRD] [△REG]
Radix	addr:H

Function overview:

The RUN command starts the execution of a program by the emulated device.

A subcommand can be specified to select one of the following three execution functions.

(a) RUN△N: Nonbreak real-time execution

Starts the execution of a target program from a specified address and traces the flow of the program. Stops the tracer when an event is detected.

The emulated device does not stop.

(b) RUN△B: Real-time execution under break conditions

Starts the execution of a user program from a specified address and traces the flow of the program. Stops the tracer and emulated device when an event is detected.

(c) RUN Δ T: Step execution

Starts the step execution of a user program from a specified address. Stops the emulation device and tracer when a specified number of instructions are executed or when a specified condition is satisfied.

For RUN Δ B or RUN Δ T, the emulated device and tracer are stopped when an event is detected, when a specified number of instructions are executed, or when a specified condition is satisfied. At this time, the system enters the one-step execution mode(Note). (See Figure 8-3.)

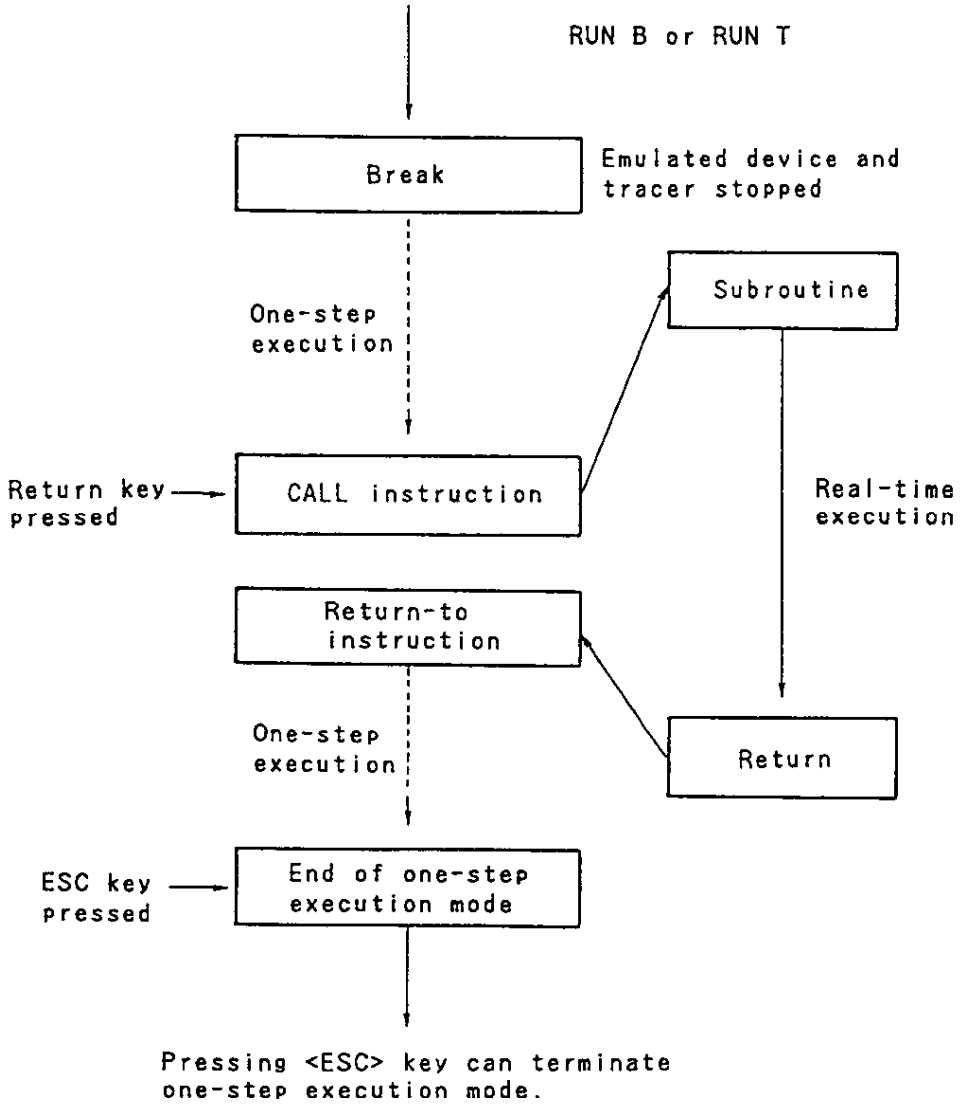
Note: One-step execution mode .

In the one-step execution mode, the target program is executed step-by-step when the <cr> (return) key is pressed. Every time a step of the program is executed, the disassembled instruction which is executed and the contents of registers in the current bank are displayed. When /<cr> (slash and return) is entered after the CALL, CALLA, CALLF, or CALLT instruction is executed, the subroutine to which a branch is made is executed in real time. Then the return-to instruction is executed, and the system enters the one-step execution mode. Nothing is executed during real-time execution. When the <ESC> key is pressed, the one-step execution mode is terminated, the system waits for a command to be entered.

Programming note:

When an interrupt request occurs in one-step execution, a request flag (IRQx) is set but an interrupt is not received. One-step execution initializes trace data acquired before the execution. The system is not in the STOP or HALT state in the one-step execution mode.

Fig. 8-3 One-Step Execution Mode



(1) Nonbreak real-time execution

Format 1 RUN △ N[△ addr]	
Radix	addr:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Starts the execution of the target program from the address specified for the operand and traces the flow of the program. Stops only the tracer when an event is detected or when an STP T command is entered.

The emulated device stops only when a fail-safe break occurs or when an STP command is entered to terminate the CPU forcibly.

(a) When the operand is omitted, the current value of the PC is used for the execution start address.

(b) The tracer and emulated device can be forcibly stopped as follows:

	Current mode	
	trc:n>	emu:n>
STP command	. Stops both the tracer and emulated device. . The system enters the brk:n> mode.	. Stops the emulated device. . The system enters the brk:n> mode.
STP T command	. Stops the tracer. . Does not stop the emulated device. . The system enters the emu:n> mode.	. Cannot be used.

(c) When the tracer is stopped by detection of an event or an STP T command, the cause of the tracer stop is displayed.

(i) Event detection

BRA1, BRA2, BRA3, BRA4, BRS1, or BRS2

(ii) Tracer stop specification

STP T

(d) The emulated device stops when a fail-safe break occurs or when an STP command is entered to terminate the CPU forcibly. In this case, the cause of the stop of the emulated device and the contents of registers in the current bank are displayed.

(i) Fail-safe break

GDM, GDIO, GDR, or GDSP

(ii) Forced termination specification

STP

Programming:

Operand

Specify the start address of the target program to be executed.

addr: Target program execution start address
Valid address range: 0 to 0FFFFH

Examples:

(a) Nonbreak real-time execution

```
brk:0>RUN N 100 <cr>

  Emulation Start at 0100                                          ①

trc:0>■

  [        ] tracer stop                                           ②

emu:0>■                                                           ③

emu:0>STP <cr>                                                    ④

  STP break terminated

  XA HL DE BC XA' HL' BC' RBS MBS RBS MBE CY IST1 ISTO SP   PC
  00 00 00 00 00  00  00   0   0   0   0  0    0    0  000 0000

brk:0>■

brk:0>RUN N 100 <cr>

  Emulation Start at 0100

trc:0>■                                                           ⑤

  [        ] tracer stop                                          ⑥

emu:0>■                                                           ⑦

emu:0>TRG <cr>

  Tracer Start

trc:0>■                                                           ⑧
```

① The system waits for a command in the trace mode.

② One of the following causes of the tracer stop is displayed: BRA1 to BRA4, BRS1, BRS2, or STPT.

③ The system waits for a command in the emulation mode.

④ A break is caused by an STP command.

⑤ The system waits for a command in the trace mode.

⑥ One of the following causes of the tracer stop is displayed: BRA1 to BRA4, BRS1, BRS2, or STPT.

⑦ The tracer is restarted by a TRG command.

⑧ The system waits for a command in the trace mode.

(b) In this example, a break is forcibly caused when RUN N is executed.

```
trc:0>■                                                          ①

  [      ] break terminated                                      ②

  XA HL DE BC XA' HL' BC' RBS MBS RBS MBE CY IST1 IST0 SP   PC
  00 00 00 00 00  00  00   0   0   0   0  0    0    0  000 0000

brk:0>■
```

① The system waits for a command in the trace mode.

② One of the following causes of the forced break is displayed:
GDM, GDIO, GDR, or GDSP

(2) Real-time execution under break conditions

Format 2 RUN △ B[△ addr]	
Radix	addr:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

Starts the execution of the target program from the address specified for the operand. Stops both the emulated device and tracer when an event is detected. Also stops them when a fail-safe break occurs or when an STP command is entered to terminate them forcibly.

(a) When the operand is omitted, the current value of the PC is used for the execution start address.

(b) An STP command can be entered to stop the emulated device forcibly.

An STP command stops both the tracer and emulated device and the system enters the brk:n> mode from the trc:n> mode.

(c) When the emulated device is stopped, the cause of the stop and the contents of registers in the current bank are displayed.

(i) Event detection

BRA1, BRA2, BRA3, BRA4, BRS1, or BRS2

(ii) Forced termination specification

STP T

(iii) Fail-safe break

GDM, GDIO, GDR, or GDSP

(d) When the emulated device is stopped by detection of an event, the cause of the stop and the contents of registers in the current bank are displayed, and the system enters the one-step execution mode.

Programming:

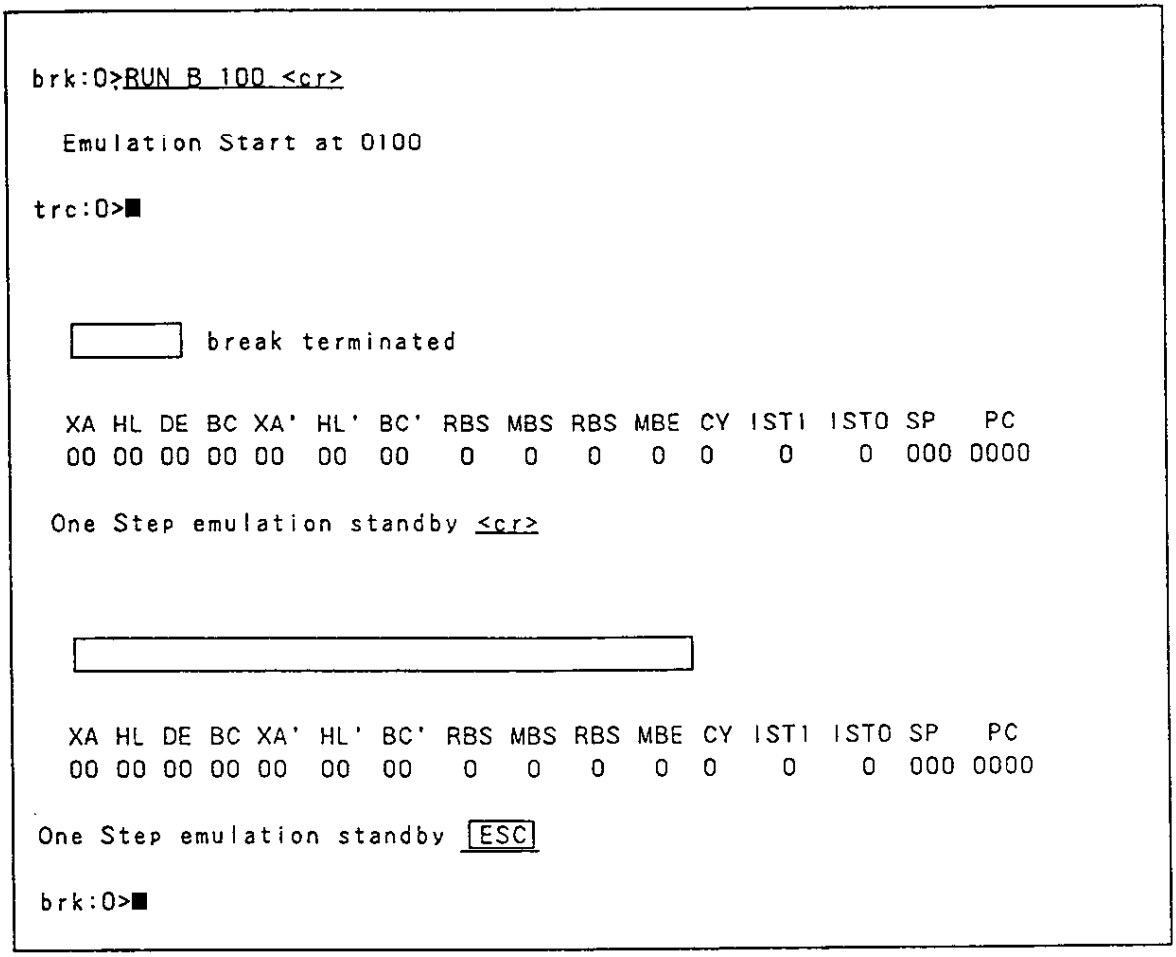
Operand

Specify the start address of the target program to be executed.

addr: Target program execution start address
Valid address range: 0 to 0FFFFH

Examples:

(a) In this example, a break is caused during RUN B, and the system enters the one-step execution mode.



① The execution of the program starts at address 100.

② The system waits for a command in the trace mode.

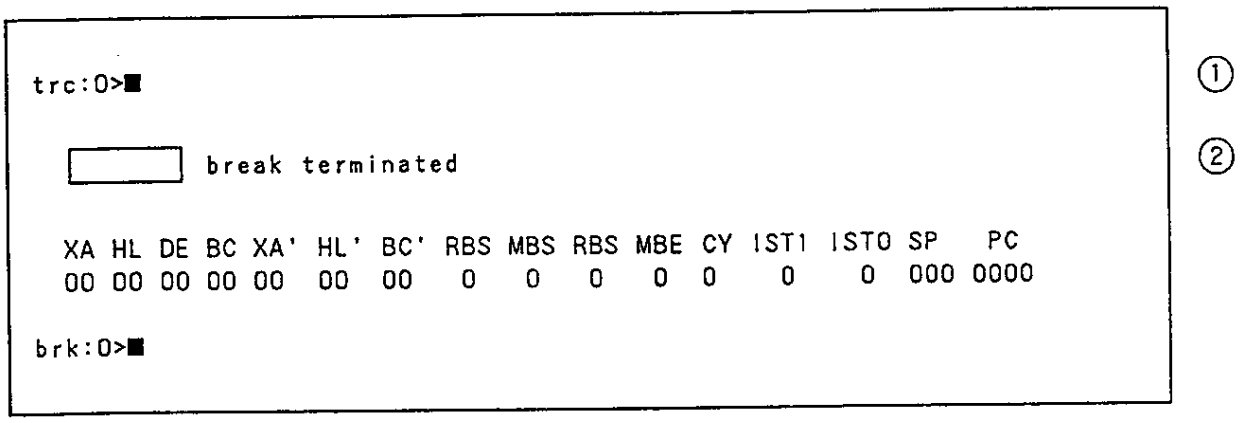
③ One of the following causes of the break is displayed:
BRA1 to BRA4, BRS1, or BRS2

④ The <cr> key is pressed.

⑤ The trace results are displayed in the same way when a TRD command is executed.

⑥ The <ESC> key is pressed.

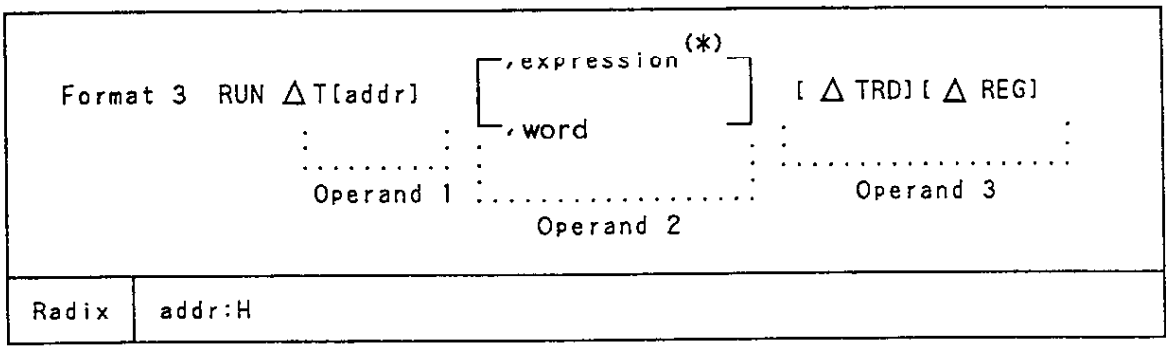
(b) In this example, a forced break is caused during RUN B.



① The system waits for a command in the trace mode.

② One of the following causes of the forced break is displayed:
GDM, GDIO, GDR, or GDSP

(3) Step execution



* expression: register(Note 1) { = > < => >= =< <= >< <> } { Mask data(Note 2) / Bit data / 4-bit data / 8-bit data / 16-bit data }

Notes 1. register: X, A, H, L, D, E, B, C, XA, HL, DE, BC, XA', HL', DE', BC', PC, SP, RBS, MBS, CY, RBE, MBE, IST1, or IST0

2. Mask data: For mask data, only the following conditions can be used: =, ><, and <>.

Function:

Starts step execution of the target program from the address specified for operand 1 and continues the execution until the condition specified for operand 2 is satisfied or a specified number of steps are executed.

During step execution, the disassembled instruction which is executed or the contents of registers in the current bank are displayed according to the specification of operand 3.

Programming notes:

. When operand 1 is omitted, the current value of the PC is used for the execution start address.

. When operand 2 is omitted, only one step of the program is executed.

. When operand 3 is omitted, the disassembled instruction which is executed and the contents of registers in the current bank are not displayed during step execution.

. The emulated device stops when the register condition is satisfied or when a specified number of steps are executed. At this time, the contents of registers in the current bank are displayed, and the system enters the one-step execution mode.

. Step execution is not made in real time. Therefore, BRM and TRM are invalidated during step execution. When the system enters the one-step execution mode, the trace data are initialized. After step execution, trace data cannot be displayed by a TRD command.

. When a fail-safe break occurs or when the <ESC> key is entered to terminate step execution forcibly, the cause of the stop and the contents of registers in the current bank are displayed, and the system enters the brk:n> mode.

(i) Fail-safe break

GDM, GDIO, GDR, or GDSP

(ii) Forced termination specification

ESC

Programming:

(a) Operand 1

Specify the start address of step execution.

addr: Step execution start address
Valid address range: 0 to 0FFFFH

Default: The current value of the PC is used for the step execution start address.

(b) Operand 2

Specify the termination condition of step execution.

expression: Specify the name of a register and its condition to terminate step execution using the contents of the register.

register:
X, A, H, L, D, E, B, C, XA, HL, DE, BC, XA', HL', DE', BC', PC, SP, RBS, MBS, CY, RBE, MBE, IST1, or IST0

Remark: The registers which can be specified depend on the target devices

Relational operator:

=, >, <, =>, >=, =<, ><, or <>

Data to be compared:

Mask data
Bit data
4-bit data
8-bit data
16-bit data

Remark: For mask data, only the following relational operators can be used: =, ><, and <>.

word: Specify the number of steps to terminate step execution when a specified number of steps are executed.

Valid number range: 1 to 65535T

Default: It is assumed that 1 is specified for the number of steps.

(c) Operand 3

Specify data to be displayed during step execution. Data are displayed for each step.

TRD: Displays the disassembled instruction which is executed.

REG: Displays the contents of registers in the current bank.

Default: Displays nothing.

Example:

In this example, trace data are displayed.

```
brk:0>RUN T 100, A=1 TRD REG <cr>                                    ①

  Emulation Start at 0100

                                                                     ②
  XA HL DE BC XA' HL' BC' RBS MBS RBS MBE CY IST1 IST0 SP   PC
  00 00 00 00 00  00  00   0   0   0   0  0    0    0  000 0000
                          .
                          .
                          .
                          .
                          .
  Terminated
  One Step emulation standby <cr>                                    ③

                                                                     ④
                          .
                          .
  At forced termination

    [      ] break terminated                                        ⑤

  XA HL DE BC XA' HL' BC' RBS MBS RBS MBE CY IST1 IST0 SP   PC
  00 00 00 00 00  00  00   0   0   0   0  0    0    0  000 0000

brk:0>■
```

① The flow of the program is traced until the contents of register A become 1.

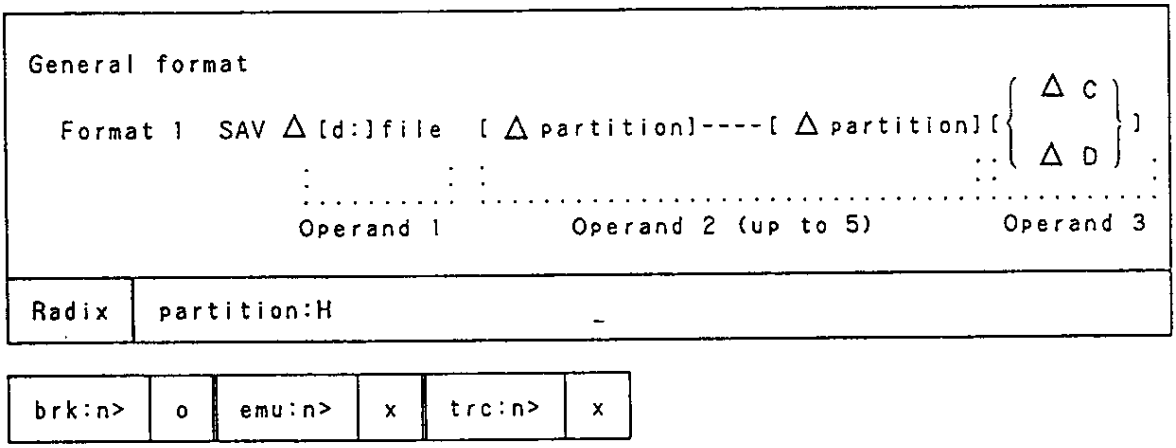
② An instruction is traced and disassembled.
Trace data are displayed in the same way when a TRD command is executed.
This processing repeats until the contents of register A become 1 or a forced break condition is satisfied.

③ When the condition (register A = 1) is satisfied, the <cr> key is pressed.

④ The system enters the one-step execution mode in the same way when RUN B is executed.

⑤ One of the following causes of the forced break is displayed:
ESC, GDM, GDIO, GDR, or GDSP

8.4.31 Save command (SAV)



Function:

The save command saves object codes and/or a debugging environment to a file whose name is specified in operand 1 on the host machine connected with channel 1 of the IE-75001-R.

(a) The object codes in a program memory area specified in operand 2 are saved to file filename.HEX on the host machine in the Intel standard hexadecimal format.

(b) The debugging environment set in the command is saved to file filename.DBG on the host machine. Debugging environments that can be saved by the SAV command are those set up with the following commands.

BRA1, BRA2, BRA3, BRA4, BRK, BRM, BRS1, BRS2, CHK, CLK, DLY, MOD, OUT, PAS, PGM, REG, STS, TRF, TRM, TRX, TRY

(c) When operand 2 is not specified, data at addresses 0 to 0FFFFH in program memory are saved.

(d) Either the object codes or the debugging environment is saved by specifying operand 3. When operand 3 is not specified, both the object codes and the debugging environment are saved.

(e) When a file name is already entered, the following processing is performed by specifying file attributes.

(i) DIR and R/W attributes

The system displays the following message and gives users a choice of deleting the existing file and creating a new file, or leaving the existing file as is.

File already exists. Delete? (Y or N) :_

Y: Deletes the existing file and creates a new file.

N: Leaves the existing file as is.

(ii) SYS and R/O attributes

The system displays the following message and ignores the command.

File already exists.

Programming:

(a) Operand 1

Specify the name of a file to which object codes and/or a debugging environment is to be saved.

file: A file name without an extension

(b) Operand 2

partition: Specifies a program memory area containing data to be saved.

Address format:
Start address, end address

Valid address range: 0 to 0FFFFH

Number of operands: Up to five

Default:
64K-byte data at 0 to 0FFFFH in program memory are saved.

(c) Operand 3

Specify the type of data to be saved.

C: Only the object codes are saved.

D: Only the debugging environment is saved.

Default: Both the object codes and the debugging environment are saved.

Examples:

(a) Without addresses

This sample program saves all data in the coverage measurement range of program memory (same as that of the loaded object file) and the debugging environment to files SAMPLE.HEX and SAMPLE.DBG respectively on the current drive in the host machine.

```
brk:0>SAV SAMPLE <cr>

  Object save complete
  debug date save complete

brk:0>■
```

(b) With addresses

This sample program saves data at addresses 0 to 1FFFFH and 3000H to 30FFH to file SAMPLE.HEX on the current drive in the host machine.

```
brk:0>SAV SAMPLE 0, 1FFF 3000,30FF C <cr>
  File already exists. Delete?  (Y or N) :Y <cr>          ①
  Object save complete
brk:0>■
```

① When file SAMPLE.HEX already exists

(c) When file SAMPLE.HEX already exists and has attribute SYS or R/O

```
brk:0>SAV SAMPLE C <cr>
  File already exists
```

8.4.32 Switch emulated device mode command (SET)

General format

Format 1 SET [{ △ STACK / △ SLWAIT } { △ OFF / △ ON }]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SET command changes the number of bytes per stack used when an emulated device interrupt occurs. It also switches wait times used after a reset in the emulated device. When operands are not specified, the current STACK and SLWAIT settings are displayed with ON or OFF.

Programming:

(a) Operand 1

Select the number of bytes per stack or the wait time.

STACK: The setting of the number of bytes per stack used on an emulated device interrupt is selected.

SLWAIT: The setting of the wait time used after a reset in the emulated device is selected.

(b) Operand 2

Set a value for STACK or SLWAIT.

STACK:
OFF: The number of bytes per stack is set to 2.
ON: The number of bytes per stack is set to 3.

If the target device is specified as any of the uPD75217, uPD75218, uPD75P218, uPD75236, uPD75237, uPD75238, uPD75P238, uPD75P336, uPD75517, uPD75518, and uPD75P518 (internal ROM 24- or 32-byte device) using the STS command or the DIP switch (SW2) on the IE-75000-R-EM (emulation board), the three-byte stack is automatically selected. Otherwise, the two-byte stack is selected.

SLWAIT:
OFF: The wait time is set to 31.25 ms. It is a default value.
ON: The wait time is set to 7.81 ms.

Programming note:

The set values become effective after the reset command is executed.

Examples:

(a) Without operands

When operands are not specified, current set values are displayed.

```
brk:0>SET <cr>

  STACK:OFF
  SLWAIT:OFF

brk:0>■
```

(b) Switching emulated device mode

This sample program sets STACK and SLWAIT, and switches the emulated device mode.

```
brk:0>SET STACK ON <cr>                    ①
brk:0>SET SLWAIT OFF <cr>                  ②
brk:0>RES <cr>                             ③
```

① The number of bytes per stack is set to 3.
② The wait time is set to 31.25 ms.
③ The reset command is executed. After the execution the mode is switched.

8.4.33 Manipulate special register command (SPR)

General formats

Format 1 SPR △C[{ △group / △register }]

Format 2 SPR[△D[{ △group / △register }]]

Function overview:

The SPR command changes or displays I/O areas (called special registers) in the target devices.

The special registers which can be changed or displayed vary with what the target device is; they are specified with the STS command.

(a) Group name

In 75X series, the special registers are mapped at addresses 0F80H to 0FFFH in data memory. The SPR command specifies the name of a special register group with 8 high-order bits of each mapping address. The special registers which belong to the group can be changed or displayed as a whole by specifying the group name.

F8X: The special registers mapped at 0F80H to 0F8FH in data memory are changed or displayed.

Valid group names depend on the target devices.

(b) Register name

The special registers are changed or displayed by specifying the names of the special registers. The names depend on the target devices.

(1) Changing special registers

Format 1 SPR Δ C[{ Δ group / Δ register }]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The command changes special registers specified by register or group name.

(a) When register names are entered, only the specified special registers are changed.

(b) When a group name is entered, all the special registers which belong to the group are changed.

(c) When neither a register name nor a group name is entered, all special registers are changed in group units.

(d) To stop changing of special registers in group units, press the <ESC> key. To proceed to the next register without completing the change of the current register, press the return key.

(e) The special registers which can be changed are write only and read/write registers.

(f) Distinguishing the write only register and the read only register

. A write only register is marked Wo.

. A read only register is marked Ro.

Programming:

(a) Subcommand

Enter C.

(b) Operand

group: Eight high-order bits of data memory addresses 0F80H to 0FFFH where special registers are mapped are specified as a group name.

Remark: Valid group names vary with the target devices.

register: The name of a special register is specified.

Default: All special registers are specified to be changed in group units.

Examples:

(a) Specifying a group name

This sample program changes special registers by specifying a group name.

```
brk:0>SPR C F8X <cr>                                    ①

 [F8X]
 SBS        2=1 <cr>                                    ②
 BTM.3     .0=1 <cr>
 BTM       Wo=<cr>                                      ③
 BT        00=Ro <cr>                                   ④
 DSPM      Wo=<cr>
 DIMS      Wo=<cr>
 DIGS      Wo=<cr>
 KSF.3     .0=Ro <cr>
 KSF        2=Ro <cr>

brk:0>■
```

① The special registers at addresses 0F80H to 0F8FH are changed by specifying the group name.

② The read/write special registers are changed.

③ The write only register is marked Wo.

④ The read only register is marked Ro.

(b) Specifying a register name

This sample program changes a special register by specifying a register name.

```
brk:0>SPR C BTM <cr>                                          ①
 BTM          Wo=1 <cr>
 brk:0>■
 brk:0>SPR C BT <cr>
 BT           00=Ro <cr>                                      ②
brk:0>■
```

① The special register is changed by specifying the register name.

② If an attempt to change read only registers is made, Ro is displayed and the registers cannot be changed.

(c) Without operands

When no operand is specified, all special registers are specified to be changed.

```
brk:0>SPR C <cr>
 [F8X]                                                        ①
 SBS                1=0 <cr>
 BTM.3             .1=0 <cr>
   .
   .
   .
 [F9X]                                                        ②
 TPGM.3            .0=<cr>
 TPGM              Wo=0 <cr>
   .
   .
   .
```

① A group name is displayed.

② A group name is displayed.

(2) Displaying special registers

Format 1 SPR[△ D[{ △ group / △ register }]]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SPR D command displays information about the special registers specified by register or group name.

(a) When a register name is specified, information about a special register with the specified name is displayed.

(b) When a group name is specified, information about the special registers belonging to the specified group is displayed.

(c) When neither register nor group name is specified, information about all special registers is displayed.

(d) Pressing the <ESC> key can abort a display started with a group name specified.

(e) A write only special register is marked Wo.

Programming:

(a) Subcommand

Enter D.

(b) Operand

group: Specifies the 8 high-order bits of a start address (0F80H to 0FFFH) of a data memory area where a special register is mapped as a group name.

Remark: Valid group names depend on the target devices.

register: Specifies a special register name.

Default: All special registers are selected for display in units of groups.

Examples:

(a) Specifying a group name

Special registers are specified by group name in this example.

```
brk:0>SPR D F8X <cr>                                              ①

 [F8X]
          SBS=1    BTM.3=.1  BTM=Wo  BT=00  DSPM=Wo  DIMS=Wo      ②
          DIGS=Wo  KSF.3=.0  KSF=2

brk:0>■
```

① Group name is specified.

② Register name and value are displayed.

(b) Specifying a register name

Special registers are specified by register name in this example.

```
brk:0>SPR D BT <cr>                                    ①
 BT          00
brk:0>■
brk:0>SPR D BTM <cr>                                   ②
 BTM         Wo
brk:0>■
```

① Register name is specified.
② A write only special register is displayed.

(c) When no operand is specified

If no operand is specified, all special registers are selected for display.

```
brk:0>SPR D <cr>                                                    ①
 [F8X]
          SBS=1     BTM.3=.1  BTM=Wo  BT=00  DSPM=Wo  DIMS=Wo      ②
          DIGS=Wo   KSF.3=.0  KSF=2
 [F9X]
      TPGM.3=.0  TPGM=Wo  MODL=00  MODH=00  WM=Wo
 [FAX]
  .         .....
 [FBX1]
  .
  .         .....
 [FBX2]
  .
  .
 [FEX]
  .
  .
 [FEX]
  .
  .         .....

brk:0>■
```

① All special registers are specified for display.

② Special register name=value

8.4.34 Redirect input command (STR)

General format

Format 1 STR △ [d:]file[△ parameter list]

brk:n>	o	emu:n>	x	trc:n>	o

Function:

The STR command fetches all commands and data that follow this command from a specified file on the host machine. Actual parameter specified in this command can be substituted with formal parameters in the file.

(a) Entering ^L breaks fetching of commands and data from a file.

(b) When the system is ready for command entry, entering ^L restarts command and data fetching.

(c) Entering ^K terminates execution of the command.

Programming:

Operand

[d:]file:	Specifies the name of a file containing commands and data. A drive number may be omitted.
parameter list:	Specifies actual parameters. Up to four actual parameters can be specified.

Programming note:

Creating an input file

Input files usable with this command include files created with a COM command and command and/or data files created with an editor. When formal parameters are used, input files must have been created with an editor.

Formal parameters are coded as $0, $1, $2, or $3.

When relative addresses are used, $ for formal parameters must be substituted by $$ so that it can be distinguished from $ used in the assembly language.

Examples:

(a) Commands are supplied from file SAMPLE.STR.

```
brk:0>STR b:SAMPLE.STR <cr>                                          ①

 brk:0>LOD SAMPLE
   object load complete
   symbol table loading
   PUBLIC          load complete
   MOD00           load complete
   MOD01           load complete
   MOD02           load complete
   MOD03           load complete
   MOD04           load complete
   MOD05           load complete
 brk:0>MEM D 0X
   0000   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ................
 brk:0>MEM F 0XX 00
 brk:0>MEM D 0XX
   0000   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
   0010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
   0020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
                              :
   00F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

 brk:0>■
```

① SAMPLE.STR in drive B is specified as input file.

Detecting the end of an input file terminates the fetching of commands and data from it. When fetching from a file ends, the keyboard gets ready for entry. The contents of file SAMPLE.STR are listed below.

```
A>TYPE b:SAMPLE.STR <cr>
LOD SAMPLE
MEM D 0X
MEM F 0XX 00
MEM D 0XX
A>■
```

(b) Commands are supplied from a file containing formal parameters.

```
brk:0>STR SAMPLE.STR 0XX 0XX <cr>                                          ①

 brk:0>LOD SAMPLE
   object load complete
   symbol table loading
   PUBLIC          load complete
   MOD00           load complete
   MOD01           load complete
   MOD02           load complete
   MOD03           load complete
   MOD04           load complete
   MOD05           load complete
 brk:0>MEM D 0X                                                            ②
   0000   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F .........
 brk:0>MEM F 0XX 00                                                        ②
 brk:0>MEM D 0XX                                                           ②
   0000   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .........
   0010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .........
   0020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .........
                          :
   00F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .........

 brk:0>■
```

① Parameters are specified.

② Formal parameters have replaced actual parameters.

If parameters are specified, the input file must contain formal parameters. If it does not contain one, actual parameters are ignored.

The contents of file SAMPLE.STR are listed below.

```
A>TYPE b:SAMPLE.STR <cr>
LOD SAMPLE
MEM D 0X
MEM F $0 00
MEM D $1
A>■
```

①
②

① Formal parameter 1
② Formal parameter 2

(c) When a specified file is not found

```
brk:0>STR b:SAMPLE.STR <cr>

  B:SAMPLE.STR file not found

brk:0>■
```

①

① Message indicating a specified file has not been found

8.4.35 Stop real-time emulation command (STP)

General formats

Format 1 STP

Format 2 STP ΔT

Function overview:

The STP command causes the emulated device to stop emulation and the tracer to stop operating.

Programming note:

The STPΔT command cannot be used with a RUNΔB command.

(1) Bringing the emulated device and tracer to a halt

Format 1 STP

brk:n>	x	emu:n>	o	trc:n>	o

Function:

The STP command brings both emulated device and tracer to a halt.

After this command has been executed, the brk:n> mode is entered.

Programming:

Enter the command only.

Example:

Real-time emulation is brought to a halt by an STP command.

```
brk:0>RUN N 100 <cr>                                              ①

  Emulation start at 100

trc:0>STP <cr>                                                    ②

  STP break terminated                                            ③

  XA HL DE BC XA' HL' BC' RBS MBS RBE MBE CY IST1 IST0  SP   PC   ④
  00 00 00 00 00  00  00   0   0   0   0  0    0    0   000 0000

brk:0>■
```

① Starting emulation is specified.
② A break of emulation is specified by an STP command.
③ Message indicating this termination has been caused by an STP command.
④ Register contents

(2) Bringing the tracer to a halt

Format 2 STP ΔT

brk:n>	x	emu:n>	x	trc:n>	o

Function:

The STP T command brings only the tracer to a halt.

After this command has been executed, the emu:n> mode is entered.

Programming:

Enter the main command and operand.

Example:

The tracer is brought to a halt by the STPΔT command.

```
brk:0>RUN N 200 <cr>                    ①
  Emulation start at 0200
trc:0>STP T <cr>                        ②
  STP T tracer stop
emu:0>■
```

① Starting emulation is specified.
② Stopping trace is specified.

8.4.36 Select target device command (STS)

```
General formats
  Format 1   STS Δ C
  Format 2   STS[ Δ D]
```

Function overview:

The STS command selects a device to be debugged with IE-75001-R or displays information about a device that has been selected for debugging.

STSΔC: Selects a target device.

STSΔD: Displays information about a target device.

(a) When a target device is selected, the following items are changed.

① Guard address (data memory, SFR, register, stack memory)

② Register that can be operated on with an REG command

③ I/O port that can be referenced with a TRD command

④ Special register that can be operated on with an SPR command

⑤ Instruction set that can be used with an ASM and DAS commands

⑥ Data memory area that can be operated on with an RAM command

(b) When the IE-75000-R-EM is supplied with power, the specified target devices are the uPD75108, uPD75108A, uPD75108F, uPD75P108, uPD75P108B, uPD75112, uPD75112F, uPD75116, uPD75116F, and uPD75P116.

(1) Selecting a target device

Format 1 STS ΔC

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The STS C command selects a target device.

When this command is entered, a list of devices is displayed. A device in the list can be selected by entering a number displayed at the left of the device name.

Devices are listed over two pages of the screen. After STS C is entered, the pages can be switched using the <cr> key.

Programming:

Enter the command and subcommand C.

Note: If only the period (.) is entered without a device number, a device will not newly be selected; the previous selection remains active.

Example:

Devices uPD75512, uPD75516, and uPD75P516 are selected for debugging.

```
brk:0>STS C <cr>                                                    ①
  (1)   Target CPU uPD75004/006/008/P008
  (2)   Target CPU uPD75028
  (3)   Target CPU uPD75036/P036
  (4)   Target CPU uPD75048/P048
  (5)   Target CPU uPD75064/066/068/P068
  (6)   Target CPU uPD75104/104A/106
  (7)   Target CPU uPD75108/108A/108F/P108/P108B/112/112F/116/116F/P116
  (8)   Target CPU uPD75116H
  (9)   Target CPU uPD75117H/P117H
  (10)  Target CPU uPD75206
  (11)  Target CPU uPD75208/CG208                                   ②
  (12)  Target CPU uPD75212A/216A/CG216A/P216A
  (13)  Target CPU uPD75217
  (14)  Target CPU uPD75218/P218
  (15)  Target CPU uPD75236
  (16)  Target CPU uPD75237/238/P238
  (17)  Target CPU uPD75268
  (18)  Target CPU uPD75304/304B/306/306B/308/308B/P308/312/316/P316
  (19)  Target CPU uPD75312B/316B/P316A/P316B
  (20)  Target CPU uPD75328/P328
  (21)  Target CPU uPD75336/P336
  (22)  Target CPU uPD75402/402A/P402
  Target CPU No.7 (cr:next page/.:end)=<cr>                         ③
```

① Command entered

② Device list displayed

③ Next page displayed on the screen

```
  (23)  Target CPU uPD75512/516/P516
  (24)  Target CPU uPD75517/518/P518

  Target CPU No.7 (cr:next page/.:end)=23 <cr>                      ①
```

① 23 entered to select devices uPD75512, uPD75516, and uPD75P516

Remarks 1. The following devices are under development:
The uPD75P048, uPD75064, uPD75066, uPD75068, uPD75P068, uPD75P117H, uPD75218, uPD75P316B.

2. The uPD75402 is available only as maintenance parts.

(2) Displaying information about a device to be debugged

Format 1 STS[△ D]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The STS D command displays the following information about a target device.

① Name of the device to be debugged
② Program memory area address
③ Data memory area address
④ Data memory bank range
⑤ Register bank range

Programming:

Enter the command and subcommand D.

Example:

Information about a device to be debugged is displayed.

```
brk:0>STS D <cr>

  Target CPU      uPD75108/108A/108F/P108/P108B/112/112F/116/116F/P116
  Program Memory  0-FFFFH
  Data Memory     00H-1FFH, F80H-FFFH
  Memory Bank     0-1, 15
  Register Bank   0-3

brk:0>■
```

① ②

① Display of information about a device to be debugged is specified.

② Display of information about a device to be debugged

8.4.37 Manipulate symbol command (SYM)

General formats

Format 1 SYM △A △symbol △word { △N / △C / △D / △B }

Format 2 SYM △C △symbol △word

Format 3 SYM[△D [△module name¥(*)]]

Format 4 SYM △E[△symbol]

Format 5 SYM[{ △K / △L / △S / △M }]

* When the host machine is an IBM PC, use \ instead of ¥.

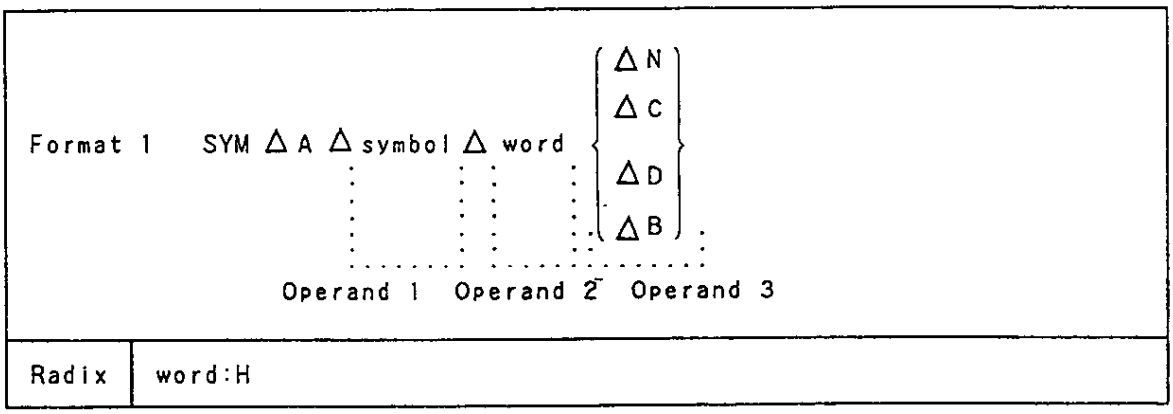
Radix	word:H

Function overview:

The operations of the SYM command depends on a subcommand entered, as listed below.

SYMΔA: Defines an append symbol.
SYMΔC: Changes a value assigned to an append symbol.
SYMΔD: Displays a symbol.
SYMΔE: Deletes an append symbol.
SYMΔK: Deletes all symbols.
SYMΔL: Loads append symbols.
SYMΔS: Saves append symbols.
SYMΔM: Specifies, displays, and changes a current module.

(1) Defining an append symbol



Radix	word:H

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYM A command defines an append symbol, to which module name IESYMBOL is assigned.

The type of a specified symbol is defined in operand 3.

N: Number type
C: Code type
D: Data type
B: Bit type

Programming:

(a) Operand 1

symbol: Specifies a name for a symbol to be added.

Note: This command cannot specify a name that is the same as for a previously defined append symbol.

(b) Operand 2

word: Specifies a value for a symbol to be added.
Valid value range: 0 to 0FFFFH

(c) Operand 3

The symbol types listed below can be specified.

Parameter	Description
N	Defines an append symbol of number type.
C	Defines an append symbol of code type.
D	Defines an append symbol of data type.
B	Defines an append symbol of bit type.

Remark: The SYM command does not check that a specified symbol name has the same string of characters as any reserved word. If there is an append symbol name with the same string of characters as a reserved word, the ASM and DAS do not treat that reserved word as reserved any more. Instead, they accepts such append symbols and types as valid.

Examples:

(a) Defining an append symbol

SYMBOL01 is defined to have value 1000 and be of code type.

```
brk:0>SYM A SYMBOL01 1000 C <cr>                    ①
brk:0>■
```

① Append symbol of code type is defined.

(b) SYMBOL02 is defined to have value 100.3 and be of bit type.

```
brk:0>SYM A SYMBOL02 100.3 B <cr>                   ①
brk:0>■
```

① Entering B defines an append symbol of bit type.

(c) An append symbol with the same string of characters as a reserved word is defined.

```
brk:0>SYM A HL 55 N <cr>                                          ①

brk:0>ASM 1000H <cr>                                              ②

 Addr    Code     Label        Mnem. Operand
 1000    60                    NOP            =MOV HL,#00H <cr>   ③
         ******** Error !! ********
 1000    60                    NOP            =,<cr>

brk:0>SYM E <cr>                                                  ④

brk:0>ASM 1000H <cr>                                              ⑤

 Addr    Code     Label        Mnem. Operand
 1000    60                    NOP            =MOV HL,#00H <cr>   ⑥
 1000    8B 00
 1002    60                    NOP            =,<cr>

brk:0>■
```

① HL (same string of characters as the name of a register pair) is defined as symbol.

② Starting assembling at address 1000H is specified. (See descriptions of the ASM command.)

③ Entering MOV HL.#00H <cr> results in an error being detected.

④ Symbol HL is deleted. (See descriptions of the SYM E command.)

⑤ Starting assembling at address 1000H is specified again.

⑥ Assembling is started.

(2) Changing a value for an append symbol

Format 2 SYM Δ C Δ symbol Δ word					
Radix	word:H				
brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYM C command changes a value that has been assigned to an append symbol by an SYM A command.

Programming:

Operand

symbol: Specifies a symbol name.

Note: It is impossible to specify any name that has not been defined as a symbol name.

word: Specifies a symbol value to be changed.
Valid value range: 0 to 0FFFFH

Examples:

(a) When a value for an append symbol is changed

```
brk:0>SYM A SYMBOL01 1000 C <cr>                    ①
brk:0>SYM C SYMBOL01 2000 <cr>                      ②
brk:0>■
```

① SYMBOL01 with value 1000 is defined.
② Symbol value is changed from 1000 to 2000.

(b) When a specified append symbol has not been defined

```
brk:0>SYM C SYMBOL99 1000 <cr>                      ①
  Symbol not found                                  ②
brk:0>■
```

① Changing a value for SYMBOL99 is specified.
② Message indicating that a symbol has not been defined

(3) Displaying information about a symbol

```
Format 3   SYM[ △ D[ △ module name¥(*) ]]

                    * When the host machine is an IBM PC, use \
                      instead of ¥.
```

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYM D command displays the name, value, and type of append, public, and module local symbols.

Specifying module name¥ in operand causes a public and local symbols for the specified module to be displayed.

If an operand is omitted, information about all defined symbols is displayed. In this case, subcommand D can be omitted.

Programming:

Operand

Specify a module name to be displayed.

module name¥: Specifies a module name. A public and local symbols for a specified module name is displayed.

Default: Information about all defined symbols is displayed.

Examples:

(a) When an append symbol has been defined

```
brk:0>SYM D <cr>                                                    ①

 module:!ESYMBOL                                                    ②
 [1000    :SYMBOL01(C)]  [2000.2 :SYMBOL02(B)]  [3000 :SYMBOL03(C)]
 [1100    :SYMBOL04(C)]  [4000   :SYMBOL05(C)]  [2100 :SYMBOL08(C)]
 module:MODULE1                                                     ③
 [0000    :XSYM0001(C)]  [0100   :XSYM0002(C)]  [0200 :XSYM0007(C)]
 [0400    :XSYM0005(C)]
 module:MODULE2                                                     ④
 [0A00    :LSYM0001(C)]  [0B00   :LSYM0002(C)]
   .
   .
 module:MODULE3                                                     ⑤
   .
   .
   .

 brk:0>■
```

① All symbols are specified to display.
② Append symbol display
③ Local symbol for module MODULE1
④ Local symbol for module MODULE2
⑤ Local symbol for module MODULE3

(b) When an append symbol has not been defined

```
brk:0>SYM D <cr>                                                    ①
 module:!ESYMBOL
 module:MODULE1
 [0000  :XSYM0001(C)]  [0100   :XSYM0002(C)]  [0200   :SXYM0007(C)]
 [0400  :XSYM0005(C)]
 module:MODULE2
 [0A00  :LSYM0001(C)]  [0B00   :LSYM0002(C)]
   .
   .
 module:MODULE3
   .
   .
   .
brk:0>■
```

① Display of information about all symbols is specified

(c) When module MODULE2 is specified

Local symbols for module MODULE2 are only displayed.

```
brk:0>SYM D MODULE2¥ <cr>                                           ①
 module:MODULE2
 [0A00  :LSYM001(C)]   [0B00   :LSYM002(C)]
   .
   .

brk:0>■
```

① Module name is specified.

(4) Deleting append symbols

Format 4 SYM △E[△symbol]

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYM E command deletes a specified append symbol.

If no operand is specified, the command deletes all append symbols defined with an SYM△A command.

Programming:

Operand

symbol: Specifies the name of a symbol to be deleted.

Note: Any symbol that has not been defined with an SYM△A command cannot be specified in the SYM E command.

Default: Deletes all append symbols defined with an SYM△A command.

Examples:

(a) When a symbol to be deleted is specified

```
brk:0>SYM E SYMBOL01 <cr>                                    ①
brk:0>■
```

① SYMBOL01 is specified to be deleted.

(b) When no operand is specified

```
brk:0>SYM E <cr>                                             ①
brk:0>■
```

① All append symbols are deleted.

(5) Deleting all symbols

Format 5 SYM △K

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYM K command deletes all predefined append symbols, public symbols, and module local symbols.

Programming:

Enter the command and subcommand K.

Example:

All symbols are deleted.

```
brk:0>SYM K <cr>                                    ①
brk:0>■
```

① All symbols are specified to be deleted.

(6) Loading append symbols

Format 5 SYM △L

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYM L command loads append symbols from append symbol file IE75000.SYM (automatically assigned name), which is supposed to be in the current drive.

Programming:

Enter the command and subcommand L.

Examples:

(a) Loading append symbols

```
brk:0>SYM ↓ <cr>

brk:0>■
```

①

① Symbols are loaded from append symbol file IE75000.SYM.

(b) When file IE75000.SYM is not in the current drive

```
brk:0>SYM ↓ <cr>

  IE75000.SYM file not found

brk:0>■
```

①

① Message indicating that IE75000.SYM was not found

(7) Saving append symbols

```
Format 5    SYM △ S
```

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYM S command saves append symbols in an append symbol file IE75000.SYM (automatically assigned name), which is supposed to be in the current drive.

Programming:

Enter the command and subcommand S.

Examples:

(a) Saving append symbols

```
brk:0>SYM S <cr>
brk:0>■
```

①

① Append symbols are saved in file IE75000.SYM.

(b) When no append symbol has been defined

```
brk:0>SYM S <cr>
  no symbol of append
brk:0>■
```

①

① Message indicating that no append symbol was defined

(c) When file IE75000.SYM is already in the current drive

```
brk:0>SYM S <cr>
  File already exists. Delete? (Y or N) Y <cr>
brk:0>■
```

①

① When file IE75000.SYM is already in the current drive, the message shown above is displayed. Entering Y <cr> in response to this message deletes the existing IE75000.SYM file and creates another IE75000.SYM file.

Entering other than Y <cr> causes this command to be ignored provided that the existing IE75000.SYM has file attribute DIR and R/W.

(d) When file IE75000.SYM cannot be created

```
brk:0>SYM S <cr>
  File already exists.
brk:0>■
```

①

① If the file attribute is SYS or R/O, a message shown above is displayed, and the command is ignored.

(8) Specifying the current module

Format 5 SYM △ M

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYM M command specifies, displays, or changes the name of the current module.

If a drive has been specified as current drive, symbols in the current module can be described without specifying its module name.

Describing local symbols belonging to another module requires that the name of the module be added in front of the local symbol name.

ALL SYM, each module name and " ¥" (without setting) can be set. The default value is ALL SYM.

The ASM, BRA, BRK, BRS, DAS, TRD, and TRF commands can display symbols. This display is executed by specifying the SYM△M command.

Module specification / Symbol type	Module specification	ALL SYM specification	No module specification (¥ specification)
Current public symbol	o	o	o
Current local symbol	o	o	x
Public symbol other than current	o	o	o
Local symbol other than current	x	o	x

Remarks: When ALL SYM is specified, only one of the local symbols having the same address is displayed.

Programming:

Enter the command and subcommand M.

Example:

(a) When specifying MOD01 as a current module.

The current module is MOD01 at first, and is changed to MOD02.

```
brk:0>SYM M <cr>
        =MOD01¥ <cr>

brk:0>SYM M <cr>
  MOD01¥ =MOD02¥ <cr>

brk:0>■
```

① ②

① MOD01 is specified as current module.
② The current module is changed to MOD02.

(b) When ALL SYM is specified as current module

```
brk:0>SYM M K <cr>                                    ①
```

① "ALL SYM" is specified as the current module. ALL SYM specification is set, and same symbol as the current symbol without SYM specification is displayed.

(c) When current module is not specified

```
brk:0>SYM M <cr>                                      ①
 ALL SYM= ¥ <cr>
```

① No module is specified. Input " ¥" after the equal (=) mark.

8.4.38 Restart system command (SYS)

General format

Format 1 SYS

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The SYS command restarts the IE75000 (control program).

Example:

(a) Restarting the system

```
brk:0>SYS                                                          ①

 IE-75000/1-R Monitor V1.4 [1 Aug 92]
   Copyright (C) 1989, 1992 by NEC Corporation

Self check ok

Target CPU      uPD75104/104A/106
Program Memory  0-FFFFH
Data Memory     00H-13FH, F80H-FFFH
Memory Bank     0-1, 15
Register Bank   0-3
 Power on target system (Y/N) Y <cr>
 Create new set up mode (Y or N) :Y<cr>
 Do you use high speed down load mode ? (Y/N) =N <cr>
 Lod object file name = TEST C S <cr>
brk:0>
```

① The system is restarted.

```
PORT XXH, XXH DATA XX
XX
E-CPU ERROR!!

   RESET

 IE-75000/1-R Monitor V1.4 [1 Aug 92]
   Copyright (C)1989, 1992 by NEC corporation

Self check ok

Target CPU      uPD75104/104A/106
Program Memory  0-FFFFH
Data Memory     00H-13FH, F80H-FFFH
Memory Bank     0-1, 15
Register Bank   0-3
Power on target system (Y/N)Y <cr>
 Create new set up mode (Y or N):Y <cr>
 Do you use high speed down load mode ? (Y/N)=N<cr>
 Lod object file name = TEST C S <cr>
brk:0>SYS <cr>

 IE-75000/1-R Monitor V1.4 [1 Aug 92]
   Copyright (C)1989, 1992 by NEC corporation

Self check ok

Target CPU      uPD75104/104A/106
Program Memory  0-FFFFH
Data Memory     00H-13FH, F80H-FFFH
Memory Bank     0-1, 15
Register Bank   0-3
Power on target system (Y/N)Y <cr>
 Create new set up mode (Y or N):Y <cr>
 Do you use high speed down load mode ? (Y/N)=N <cr>
 Lod object file name = TEST C S <cr>
brk:0>
```

①

① Press the RESET switch on the front panel of the IE-75001-R.

Remark: Pressing the RESET switch starts the IE-75001-R monitor program. Then, prompt brk: appears. Entering SYS <cr> in this mode restarts the IE75000 (control program) and displays the message. Then, prompt brk:n appears.

8.4.39 Display trace data command (TRD)

General format

Format 1 TRD [△ ALL] [△ $C] [△ option[,option][,option]...
△ $F
△ $Q
... [,option] [△ $B]]

Operand 1 Operand 2 Operand 3 Operand 4

Note: When ALL is specified in operand 1, $F cannot be specified in operand 2.

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The TRD command displays trace data.

The display format used depends on how operands are combined.

(a) Quantity of trace data to be displayed

Specifying ALL causes all trace data to be displayed. Specifying a retrieval condition ($C or $Q) causes all data that meet the specified retrieval condition to be displayed continuously.

If ALL is omitted, one page's worth of trace data starting at the current pointer is displayed. A concept of page varies according to a specified retrieval display option.

Table 8-5 Concept of Page

Retrieval display option	Number of trace data lines per page
None	12
$C	1
$Q	12
$F	11

Remark: When a trigger frame (frame for which a condition specified in a BRM command is met) is displayed, one trace data page consists of 11 lines.

(b) Retrieving trace data

Specifying $C, $Q, and $F in operand 2 can cause selected trace data to be displayed.

If operand 2 is omitted, all trace data are displayed.

$C: Selects and displays checkpoint data (trace data collected at a checkpoint specified in a CHK command) and a trigger frame (frame for which a condition specified in a BRM command is met).

$Q: Selects and displays a !frame (frame for which a retrieval condition specified in a TRF command is met).

$F: Selects a !frame or trigger frame, and displays it and two frames that precede and follow it.

Each !frame display line begins with special character "!", while each trigger frame display line begins with letter T.

(c) Additional information about trace data

This command normally displays frame number, program memory address, instruction code, label, and mnemonic.

Specifying options can add information about data memory access status, external sense clips, and I/O port to the normally displayed information. (See Table 8-6.)

Specifying $B causes a value for EXT or an I/O port name specified as an option to be displayed in binary format. If $B is not specified, such values are in hexadecimal format.

Table 8-6 Options

Format	Description
P0, P1, P2-P15	Specifies I/O port trace data to be displayed. This option cannot be used in the event cycle trace mode. (See Section 8.4.42.) Valid I/O port names vary with target devices.
EXT	Specifies external sense clip trace data to be displayed.
DMEM	Specifies trace data in data memory to be displayed. Displays include data memory address, data read from or written to data memory, and read/write state.

Remark: Options can be specified in any order. The number of options that can be specified is limited. (See programming note (a)).

(d) Menu mode

If ALL is not specified, a menu mode is entered.

If selective display is not specified, the total number of frames is displayed.

Specifying $C causes the quantity of checkpoint data to be displayed.

Specifying $Q or $F causes the number of !frames to be displayed.

Table 8-7 lists the relationships between keys pressed in the menu mode and responses to each pressed key.

Table 8-7 Entries and Responses in the Menu Mode

Entry	Retrieval display option	Operation
L	None	Displays the latest page.
	$C	Displays the latest checkpoint data.
	$Q	Displays the latest !frame page.
	$F	Displays the latest !frame page and pages around it.
F	None	Displays the previous page.
	$C	Displays the previous checkpoint data.
	$Q	Displays the previous !frame page.
	$F	Displays the previous !frame page and pages around it.
T	Common	Displays a trigger frame page and pages around it.
"+" or <cr>	None	Displays the page following one that was displayed most recently.
	$C	Displays the checkpoint data following one that was displayed most recently.
	$Q	Displays the page following the !frame page that was displayed most recently.
	$F	Displays the trigger frame or !frame following the page that was displayed most recently and pages around it.
"-"	None	Displays the page preceding one that was displayed most recently.
	$C	Displays the checkpoint data preceding one that was displayed most recently.
	$Q	Displays the page preceding the !frame page that was displayed most recently.

(to be continued)

Entry	Retrieval display option	Operation
"_"	$F	Displays the trigger frame or !frame page preceding the page that was displayed most recently and pages around it.
"."	Common	Terminates the display trace data command.

Programming

(a) Operand 1

Specify a continuous display option.

ALL: Specifies that all trace data meeting a retrieval condition be displayed. After this command has been executed, another command can be entered.

Default: Specifies that one page at the current pointer be displayed. After displaying, the menu mode is entered.

(b) Operand 2

Specify a retrieval display option.

$C: Specifies a mode in which selected checkpoint data or trigger frame is displayed.

$Q: Specifies a mode in which a selected !frame or trigger frame is displayed.

$F: Specifies a mode in which a selected !frame or trigger frame and two frames preceding and following it are displayed.

Default: Specifies that a search be not performed.

(c) Operand 3

Specify an additional display option.

option: Any combination of options listed in Table 7-6 can be specified for display of additional information; however, the number of options that can be specified is limited. (See programming note (a).)

Default: No additional information is displayed.

(d) Operand 4

Specify a binary display option.

$B: With $B specified, specifying I/O port or EXT as an option causes its value to be displayed in binary format.

Default: Display is in hexadecimal format.

Programming note:

(a) Quantity of I/O port trace data selectable for display

The quantity of I/O port trace data that can be displayed on one screen with this command varies with a display specification (DMEM, EXT, or $B) other than an I/O port specification in the operand.

Table 8-8 Quantity of Options That Can Be Specified

DMEM specification	Pn specification	EXT specification	$B specification
—	Pn ... 6	—	—
	Pn ... 5	EXT	
	Pn ... 5	—	$B
	Pn ... 3	EXT	
DMEM	Pn ... 3	—	—
	Pn ... 2	EXT	
	Pn ... 2	—	$B
	Pn ... 1	EXT	

(b) Specifying ALL

If continuous display option ALL is specified, retrieval display option $F cannot be specified.

Examples:

(a) When all trace data on each event cycle are displayed

```
brk:0>TRD ALL <cr>

Frame   PA     PD       Label     Mnemonic
  000   0100   99  10             SEL     MB0
  001   0102   89  00             MOV     XA, #0000H
  003   0104   8B  30             MOV     HL, #0030H
  005   0106   8A                 INCS    HL
  007   0107   AA  5A             MOV     XA, HL

brk:0>■
```

(b) When all trace data on each event cycle are displayed with option DMEM specified

```
brk:0>TRD ALL DMEM <cr>

Frame   PA     PD       MA    MD   MRW  Label   Mnemonic
  000   0100   99  10   ---   --   ---          SEL    MB0
  001   0102   89  00   ---   --   ---          MOV    XA, #0000H
  002                   000   00   MWR
  003   0104   8B  30   ---   --   ---          MOV    HL, #0030H
  004                   002   30   MWR
  005   0106   8A       ---   --   ---          INCS   HL
  006                   002   31   MWR
  007   0107   AA  5A   ---   --   ---          MOV    XA, HL
  008                   000   31   MWR

brk:0>■
```

(c) When all trace data on each machine cycle are displayed

```
brk:0>TRD ALL <cr>

Frame   PA     PD        Label      Mnemonic
  000   0100   99  10               SEL     MB0
  002   0102   89  00               MOV     XA, #0000H
  004   0104   8B  30               MOV     HL, #0030H
  006   0106   8A                   INCS    HL
  007   0107   AA  5A               MOV     XA, HL

brk:0>■
```

(d) When all trace data on each machine cycle are displayed with option DMEM specified

```
brk:0>TRD ALL DMEM <cr>

Frame  PA     PD        MA   MD  MRD  Label   Mnemonic
  000  0100   99  10    ---  --  ---          SEL     MB0
  001                   ---  --  ---
  002  0102   89  00    ---  --  ---          MOV     XA, #0000H
  003                   000  00  MWR
  004  0104   8B  30    ---  --  ---          MOV     HL, #0030H
  005                   002  30  MWR
  006  0106   8A        002  31  MRW          INCS    HL
  007  0107   AA  5A    ---  --  ---
                        000  31  MWR          MOV     XA, HL

brk:0>■
```

(e) When there are checkpoint data

```
brk:0>TRD ALL <cr>

Frame   PA     PD      Label     Mnemonic
  000   0100   99 10              SEL     MB0
  002   0102   89 00              MOV     XA,#0000H
  004   ***CHK REG***  XA=00  HL=00  DE=00  BC=00
  005   ***CHK REG***  XA'=00 HL'=00 DE'=00 BC'=00 RBS=0  MBS=0
  006   ***CHK REG***  RBE=0  MBE=0  CY=0   IST1=0 IST0=0 SP=000 PC=000
  007   0104   8B 30              MOV     HL,#0030H
                .
                .
                .

brk:0>■
```

The example shows that there are checkpoint data in frame numbers 004 to 006. The types of checkpoint data are listed in Table 7-9.

Table 8-9 Checkpoint Data Types

Specified type	Display format
REG	***CHK REG***
special register	***CHK SPR***
data memory	***CHK MEM***

Remark: See Section 8.4.7 for how to set up checkpoints.

(f) When trace data are displayed with an I/O port specified as an option

```
brk:0>TRD ALL P0, P1, P2 <cr>

Frame   PA    PD       P0  P1  P2  Label   Mnemonic
  000   0109  8A       F   F   F           INCS    HL
  001   010A  AA  10    F   F   F           MOV     @HL, XA
  002                  F   F   F           MOV
  003   010C  89  34    F   F   F           MOV     XA, #0034H
  004                  F   F   F

brk:0>■
```

(g) When binary option $B and additional display option EXT and I/O port name are specified

```
brk:0>TRD ALL P0, P1, EXT $B <cr>

Frame   PA    PD       EXT       P0    P1    Label   Mnemonic
  000   0109  8A       11111111  1111  1111          INCS    HL
  001   010A  AA  10    11111111  1111  1111          MOV     @HL, XA
  002                  11111111  1111  1111
  003   010C  89  34    11111111  1111  1111          MOV     XA, #0034H
  004                  11111111  1111  1111

brk:0>■
```

(h) When one page of trace data is displayed with continuous display option ALL omitted

```
brk:0>TRD DMEM <cr>

Frame  PA      PD        MA      MD  MRW   Label  Mnemonic
  018   0102   89 11     ---     --  ---          MOV      XA, #0011H
  019                    000     11  MWR
  01A   ***CHK REG***    XA=11       HL=02  DE=34  BC=00
  01B   ***CHK REG***    XA'=00      HL'=00 DE'=00 BC'=02 RBS=0  MBS=0
  01C   ***CHK REG***    RBE=11      MBE=0  CY=0    IST1=0 IST0=1 SP=0F2 PC=0104
  01D   0104   8A        002     03  MRW          INCS     HL
  01E   0105   AA 4A     ---     --  ---          SKE      XA, HL
  01F                    002     03  MRD
  020   0107   AB 01 02  ---     --  ---          BR       !0102H
  021                    ---     --  ---
  022                    ---     --  ---
  023   0102   89 11     ---     --  ---          MOV      XA, #0011H

Total frame=0D9       (L/F/T/+/cr/-/Frame No./.)?■
```

(i) When one page of trace data is displayed with retrieval display option $C specified

```
brk:0>TRD $C DMEM <cr>

Frame   PA      PD        MA      MD  MRW     Label  Mnemonic
  00F   ***CHK REG***  XA=00       HL=00   DE=00   BC=00

  Total frame=107  CHK.frame=017 (L/F/T/+/cr/-/Frame No./.)?■
```

(j) When one page of trace data is displayed with retrieval display option $Q specified

```
brk:0>TRD $Q DMEM <cr>

Frame  PA     PD          MA   MD  MRW Label     Mnemonic
  !025  1A08  89  34      ---  --  ---           MOV     XA, #0034
  !026                    000  34  MWR
  !02A  1A08  89  34      ---  --  ---           MOV     XA, #0034
  !02F                    000  34  MWR
  !030  1A08  89  34      ---  --  ---           MOV     XA, #0034
  !034                    000  34  MWR
  !035  1A08  89  34      ---  --  ---           MOV     XA, #0034
  !039                    000  34  MWR
  !03A  1A08  89  34      ---  --  ---           MOV     XA, #0034
  !03E                    000  34  MWR
  !03D  1A08  89  34      ---  --  ---           MOV     XA, #0034
  !043                    000  34  MWR

Total frame=0d9           !frame=023 (L/F/T/+/cr/-/Frame No./.)?■
```

(k) When one page of trace data is displayed with retrieval display option $F specified

```
brk:0>TRD $F DMEM <cr>

Frame  PA     PD          MA   MD  MRW Label     Mnemonic
  078                     ---  --  ---
  079   1A04  9D  88      ---  --  ---           SET1    !RQBT
  07A                     FB8   3  MRW
  07B   1A06  9D  82      ---  --  ---           EI
  07C                     FB2   8  MRW
 !07D   1A08  89  34      ---  --  ---           MOV     XA, #0034H
  07E                     000  34  MRW
  07F   1A0A  8B  30      ---  --  ---           MOV     HL, #0030H
  080                     002  30  MWR
  081   1A0C  8A          002  31  MRW           !NCS    HL
  082   1A0D  AA  5A      ---  --  ---           MOV     XA, HL

Total frame=107           !frame=023 (L/F/T/+/cr/-/Frame No./.)?■
```

(1) Operations of trace data pointer

Trace data are displayed on each machine cycle.

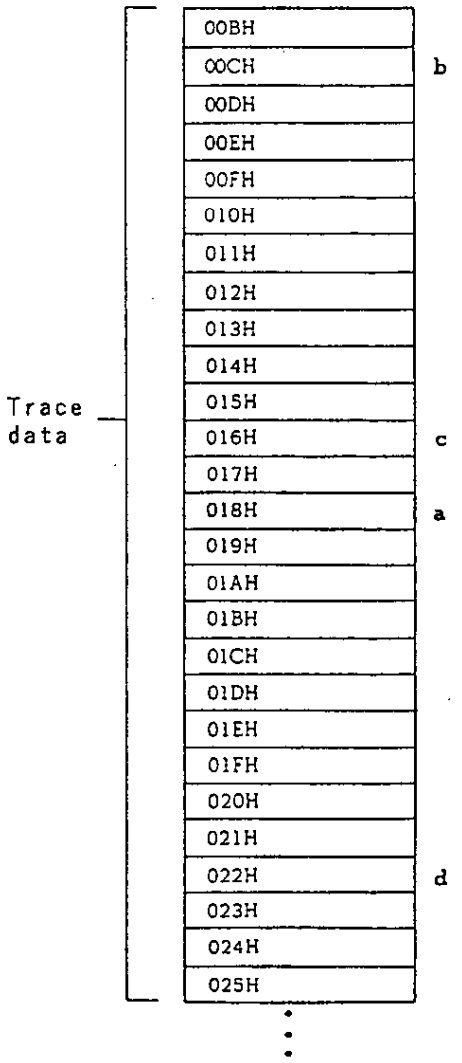
```
brk:0>TRP <cr>                                                              ①

  Total frame number=107
  Display frame number=018

brk:0>TRP C <cr>                                                            ②

brk:0>TRD DMEM <cr>                                                         ③

Frame    PA      PD     MA     MD  MRW     Label  Mnemonic
  00C                    --     --  ---
  00D    1A00    9C B2   --     --  ---            DI
  00E                    FB2    0   MRW
  00F   ***CHK REG***    XA=00      HL=00    DE=00  BC=00
  010   ***CHK REG***    XA'00      HL'=00   DE'=00 BC'=00  RBS=0  MBS=0
  011   ***CHK REG***    RBE=0      MBE=0    CY=0   IST1=0  IST0=0 SP=000 PC=0000
  012    1A02    9D 98   --     --  ---          SET1   IEBT
  013                    FB8    3   MRW
  014    1A04    9D 88   --     --  ---          SET1   IRQBT
  015                    FB8    3   MRW
  016   ***CHK REG***    XA=00      HL=00    DE=00  BC=00
  017   ***CHK REG***    XA'=00     HL'=00   DE'=00 BC'=00 RBS=0  MBS=0

  Total frame=107 (L/F/T/+/cr/-/Frame No./.)?16 <cr>                        ④

Frame    PA      PD     MA         MRW     Label  Mnemonic
  016   ***CHK REG***    XA=00      HL=00    DE=00  BC=00  RBS=0  MBS=0
  017   ***CHK REG***    XA'=00     HL'=00   DE'=00 BC'=00 IST0=0 SP=000 PC=000
  018   ***CHK REG***    RBE=0      MBE=0    CY=0   IST1=0
  019    1A06    9D B2   --     --  ---          EI
  01A                    FB2    8   MRW
  01B    1A08    89 34   --     --  ---          MOV XA,   #0034H
  01C                    000    34  MWR
  01D   >                0EA    12  MWR
  01E                    0E8    0A  MWR
  01F                    0E6    1A  MWR
  020    2000    89 55   --     --  ---          MOV XA,   #0055H
  021                    000    55  MWR

  Total frame=107 (L/F/T/+/cr/-/Frame No./.)?■                              ⑤
```

① Displaying the current pointer is specified.

② Shifting the pointer is specified.
③ Displaying the trace data is specified.
④ Specification in the menu mode
⑤ Command-acceptable state in the menu mode

Figure 8-4 illustrates the operations of the pointer described in the example above.

Fig. 8-4 Trace Pointer Chart



The current pointer is located at a (018H) by TRP command ①

It is shifted back by one page to b by TRP command ②. One page's worth of trace data starting at b (up to 017H) are displayed by TRD DMEM <cr> command ③. (The pointer is shifted to a.) After the trace data are displayed, the menu mode is entered because continuous display option ALL has not been specified.

Entering a frame number (for example, 16 <cr> as shown at ④) directly in the menu number moves back the pointer to c (016H), causing one page starting at c (up to 021H) to be displayed. (After this page of trace data is displayed, the pointer rests at d.)

Sign ">" displayed at the right of frame number 01DH means that an interrupt occurred.

(m) Displaying skipped instructions

Instructions skipped by a skip instruction are enclosed in square brackets [and]. Skipped instructions are traced but not executed.

```
Frame   PA     PD       Label     Mnemonic
 00E    1002   CF                 DECS      B
 00F    1003   9A 07              SKE       B,#0000H
 011   [1005]  ***** SKIP *****
 013    1008   A3 30              MOV       A,0030H
```

8.4.40 Set trace data retrieval condition command (TRF)

General format

Format 1 TRF [△PA=[addrX / partition1]][△PD=data1][△MA=[addrX / partition2]]
[△MD= data2] [△MRW=status] [△Pn=data3]
[△EXT=data4]

Radix	addr1-2:H partition1-2:H data1-4:H

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The TRF command sets up trace data retrieval conditions.

Retrieval conditions can be set up either on one line of command entry or interactively.

In interactive mode, retrieval conditions are specified one by one after the current retrieval conditions are displayed.

Entering TRF<cr> displays the currently active retrieval conditions, then causes the interactive set mode to be entered.

Programming:

Retrieval conditions set up with this command are used in retrieval requested by a TRD command. See Section 8.4.39 for details.

Operand

PA=addrX

partition1:

Specifies program memory retrieval addresses. An address range can be specified by mask and partition.

Valid address range: 0 to 0FFFFH
Valid mask range: 0 to 0XXXXH
Default: A new retrieval condition is not set up.
Initial value: 0XXXXH

PD=data1: Specifies data in program memory to be retrieved.

Valid data value range: 0 to 0FFH
Valid mask range: 0 to 0XXH
Default: A new retrieval condition is not set up.
Initial value: 0XXH

MA=addrX

partition2:

Specifies data memory access retrieval addresses. A valid address range varies with a target device.

Valid address range: 0 to 0FFFH
Valid mask range: 0 to 0XXXH
Default: A new retrieval condition is not set up.
Initial value: 0XXXH

MD=data2: Specifies data at accessed data memory locations to be retrieved.

Valid data value range: 0 to 0FFH
Valid mask value: 0 to 0XXH
Default: A new retrieval condition is not set up.
Initial value: 0XXH

MRW=status: Specifies the data memory access state for which a retrieval is to be made.

Specification	Access state
MRD	Memory read
MWR	Memory write
MRW	Memory read-modify-write
NC	All read and write types

Default: A new retrieval condition is not set up.
Initial value: NC

Pn=data3: Specifies I/O port retrieval data. Valid I/O port names vary with target devices.

Valid data value range: 0 to 0FH
Valid mask range: 0 to 0XH
Default: A new retrieval condition is not set up.
Initial value: 0XH for all of P0 to P15

EXT=data4: Specifies external sense clip retrieval data.
Valid data value range: 0 to 0FFH
Valid mask range: 0 to 0XXH
Default: A new retrieval condition is not set up.
Initial value: 0XXH

Default: The currently active retrieval conditions are displayed, and the interactive set mode is entered.

Programming note:

Specifying retrieval for I/O port data is invalid in event-by-event cycle mode. Specifying program fetch (PA, PD) and data memory access (MA, MD, MRW) as retrieval conditions at one time in a TRX command makes retrieval invalid.

Examples:

(a) Setting up retrieval conditions

This example specifies retrieval of trace data for program memory addresses 0H to 1FFFH.

```
brk:0>TRF PA=0,1FFF <cr>

brk:0>■
```

(b) Omitting operands

Omitting operands causes the menu mode to be entered, thus enabling interactive setting up of retrieval conditions.

```
brk:0>TRF <cr>                                          ①

  PA    0, 1FFF = 0XXX <cr>                             ②
  PD    0XXH = <cr>                                     ③
  MA    0XXXH = <cr>
  MD    0XXH = <cr>
  MRW   NC = <cr>
  P0    0X = <cr>
  P1    0X = <cr>
  P2    0X = <cr>
  P3    0X = <cr>
  P4    0X = <cr>
  P5    0X = <cr>
  P6    0X = <cr>
  P7    0X = <cr>
  P8    0X = <cr>                                       } (*)
  P9    0X = <cr>
  P10   0X = <cr>
  P11   0X = <cr>
  P12   0X = <cr>
  P13   0X = <cr>
  P14   0X = <cr>
  P15   0X = <cr>
  EXT   0XX = <cr>

brk:0>■
```

① Menu mode is specified.

② 0XXXH is newly specified.

③ Current data remain unchanged.

* A display depends on a target device.

8.4.41 Trigger tracer command (TRG)

General format

Format 1 TRG

brk:n>	x	emu:n>	o	trc:n>	x

Function:

The TRG command restarts the tracer.

The tracer can be restarted, only when the emulated device is running but the tracer remains at rest (emulation mode).

Executing this command causes the trace mode to be entered.

Programming:

Enter the command only.

Programming note:

This command cannot be used in the break mode or the trace mode.

Example:

The tracer is restarted.

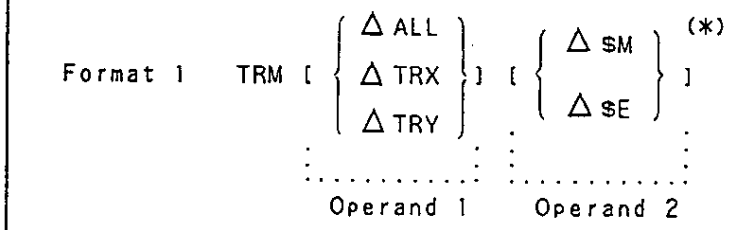
```
emu:0>TRG <cr>                                          ①
 Tracer start
trc:0>■
```

① The tracer is restarted.

A prompt serves to indicate that the tracer has been restarted.

8.4.42 Select trace mode command (TRM)

General format



brk:n>	o	emu:n>	o	trc:n>	x

* If TRX or TRY is specified in operand 1, $M cannot be specified in operand 2.

Function:

The TRM command selects a trace condition mode or a trace cycle mode for real-time trace.

(a) Specifying ALL in operand 1 causes an unconditional trace through a program being emulated on each specified cycle (Δ$M or Δ$E).

(b) Specifying TRX or TRY in operand 1 causes a trace on each event under the conditions specified in each command. For details, see Sections 8.4.44 and 8.4.45.

(c) Entering TRM<cr> indicates what is the current trace mode.

Programming:

(a) Operand 1

Specify a trace condition mode.

ALL: Specifies an unconditional trace, in which the tracer is started real time during emulation.

TRX: Specifies a qualified trace, in which the tracer is started when an event condition is met at a trace point. See descriptions of the TRX command for details.

TRY: Specifies a sectional trace, in which the tracer is started when an event condition is met at the trace start point and ended when an event condition is met at the trace end point. See Section 8.4.45 for details.

Initial value: ALL

(b) Operand 2

Specify a trace cycle mode.

$M: Specifies a trace mode for an individual machine cycle. In this mode, the states of program address, data memory bus, read/write signal, port, and external sense clip signal are traced on each machine cycle.

$E: Specifies a trace mode for an individual event cycle. In this mode, only the requested types of information are traced on every occurrence of events, except for ports, for which no trace is performed.

Initial value: $M

Programming note:

Neither qualified trace (TRX) nor sectional trace (TRY) can be specified as a trace condition for the machine cycle trace mode ($M).

Example:

An unconditional trace is performed on machine cycles.

```
brk:0>TRM <cr>
 TRK $E                                                    ①
brk:0>TRM ALL $M <cr>                                      ②
brk:0>TRM <cr>
 ALL $M                                                    ③
```

① Current trace mode is displayed.
② New trace mode is specified.
③ Current trace mode is displayed.

8.4.43 Manipulate trace pointer command (TRP)

General format

Format 1 TRP [{ Δword / ΔF / ΔL / ΔT }]

Radix	word:H

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The TRP command moves the trace pointer to any specified trace memory location.

(a) Shifting and displaying the trace pointer

Entering TRP F/L<cr> causes the trace pointer to move to the beginning or end of trace memory.

Entering TRP T<cr> causes the trace pointer to move to a frame (trigger frame for which a condition specified in a BRM command is met). If there is no such frame, this command has no meaning.

If the number of words through which the trace pointer to move is specified in the operand, the pointer can move by ±7FFH relative to its current location.

Entering TRP<cr> displays the number of trace frames in trace memory. It also indicates where the trace pointer is now.

(b) Relationships between TRP and TRD commands

A TRD command with ALL omitted begins trace data display at the pointer specified in the TRP command.

If the TRP command has T in the operand, the corresponding TRD command displays a line of the trigger frame trace data and 5 lines of trace data before and after it.

If a TRD command has a retrieval display option $C, $Q, or $F, there might be a slight difference between a memory location pointed to by the current trace pointer and a location where the display actually begins because of a time lag caused in a search. See Section 8.4.39 for details.

(c) Specifying absolute address

Specification with absolute address is available only when trace data exists.

Programming:

Operand

word: Specifies how much the trace pointer to move away from the current location.
Valid pointer shift range: ±1H to ±7FFH

F: Causes the trace pointer to move to the beginning of trace memory.

L: Causes the trace pointer to move to the end of trace memory.

T: Causes the trace pointer to move to the trigger frame. If there is no trigger frame, this command is meaningless.

Default: The quantity of trace data and the trace pointer are indicated in hexadecimal format.

Examples:

(a) Moving the trace pointer

```
brk:0>TRP 20 <cr>                    ①
brk:0>TRP L <cr>                     ②
brk:0>■
```

① The trace pointer is advanced by 20H.
② The trace pointer is moved to the end of trace memory.

(b) When no operand is specified

The quantity of the current trace data and the trace pointer are displayed.

```
brk:0>TRP <cr>
  Total frame number  =200                    ①
  Display frame pointer =012
brk:0>■                                       ②
```

① The quantity of traced frames is displayed.
② The trace pointer value is displayed.

(c) Absolute address specification

```
brk:0>TRP 050H<cr>
```

① Display pointer 050H is specified. This command is valid only when trace data exists.

8.4.44 Set qualified-trace condition command (TRX)

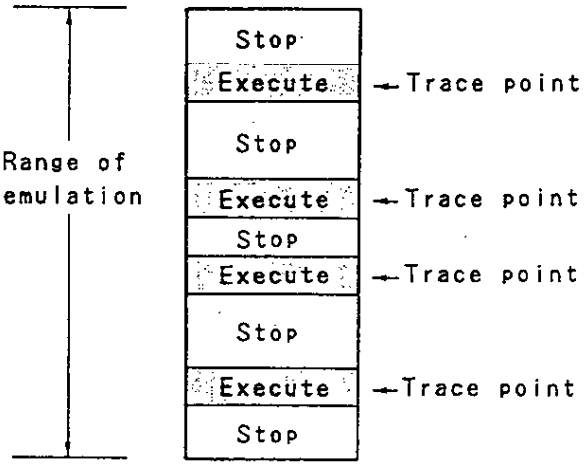
```
General format
  Format 1   TRX{ △ BR?}[ △ BR?}[ △ BR?}[ △ BR?}[ △ BR?}[ △ BR?}
```

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The TRX command specifies that event conditions in event condition registers be used as event conditions for qualified trace.

Fig. 8-5 Tracer Status Diagram



Qualified trace is intended to trace only the necessary information. The TRX command specifies event conditions in event condition registers as trace point conditions.

As Figure 8-5 shows, trace data are written to the tracer only when a trace point condition is met.

Entering TRX<cr> causes the currently active conditions to be displayed.

If BRA3 or BRA4 is specified in an OUT command, the TRX command cannot specify BRA3 or BRA4.

Programming:

Operand

BR?: Up to 6 of the following event condition registers can be specified.

BRA1, BRA2, BRA3, BRA4, BRS1, BRS2, OFF

Remark: Specifying OFF cannot trigger the tracer.

Initial value: OFF

Default: The currently active conditions are displayed.

Programming note:

To perform qualified trace under the conditions specified by the TRX command, TRX (qualified trace) must previously specified by a TRM (select trace mode) command. When a skip operation is performed, the message which indicates the skip operation, *****SKIP*****, is not displayed in the qualified trace information. See Section 8.4.42 for details.

Examples:

(a) Specifying qualified-trace conditions

When data memory address 0100H is accessed, trace data are written to the tracer.

```
brk:0>BRA 1 MA=0100 <cr>          ①
brk:0>TRX BRA1 <cr>               ②
brk:0>TRM TRX <cr>                ③
brk:0>RUN N <cr>                  ④
```

① Data memory address is set in BRA1.
② BRA1 is specified as qualified-trace conditions.
③ Qualified trace is specified.
④ Executing emulation is specified.

(b) When no operand is specified

The currently active settings are displayed.

```
brk:0>TRX <cr>
  BRA1                                                    ①

brk:0>■
```

① BRA1 is set as qualified-trace conditions.

8.4.45 Set sectional-trace condition command (TRY)

General format

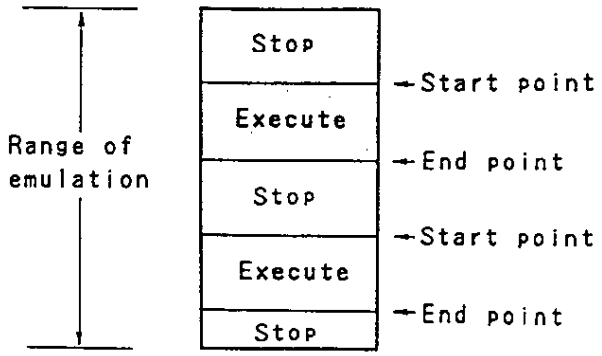
Format 1 TRY{ △E / △D }[△BR?][△BR?][△BR?][△BR?][△BR?][△BR?]

brk:n>	o	emu:n>	o	trc:n>	x

Function:

The TRY command specifies that event conditions in event condition registers be used as sectional-trace event conditions.

Fig. 8-6 Tracer Status Diagram



Sectional trace is intended to trace only a specified range. The TRY command specifies event conditions in event condition registers as start and end point conditions.

As Figure 8-6 shows, the tracer is triggered when a start point condition is met and deactivated when an end point condition is met. When the tracer is active, trace data are written to the tracer.

Entering TRY<cr> causes the currently active conditions to be displayed.

If BRA3 or BRA4 is specified in an OUT command, the TRY command cannot specify BRA3 or BRA4.

Programming:

(a) Subcommand

E: Specifies a condition for a start point.
The tracer is triggered when any condition in the event condition registers specified in the operand is met.

D: Specifies a condition for an end point.
The tracer is deactivated when any condition in the event condition registers specified in the operand is met.

(b) Operand

BR?: The following event condition registers can be specified, but registers for data memory access related trace (BRA1 to BRA4) cannot be used together with those for program fetch related trace (BRS1 and BRS2).

BRA1, BRA2, BRA3, BRA4, BRS1, BRS2, OFF

Remark: Specifying OFF invalidates start and end point conditions.

Initial value: Start point condition --> OFF
End point condition --> OFF

Programming note:

(a) To activate and deactivate the tracer under the conditions specified by the TRY command, TRY (sectional trace) must previously specified by a TRM (select trace mode) command. See Section 8.4.42 for details.

(b) If any of BRA1 to BRA4 is specified as start point condition, an end point condition must be specified by any of BRA1 to BRA4.

(c) If BRS1 or BRS2 is specified as start point condition, an end point condition must be specified by BRS1 or BRS2.

(d) If BRS1 or BRS2 is specified as an end point, only the last trace information may not be displayed under the following condition:

A two- or three-byte instruction is specified at the end point.

If any specification other than described above is given, no valid condition can be set up to deactivate the tracer.

Examples:

(a) Specifying conditions to activate and deactivate the tracer

Program memory event registers BRS1 and BRS2 are specified for the start and end point of sectional trace.

The tracer starts operating at program address 2000H and stops at 3000H.

```
brk:0>BRS 1 P 2000H <cr>                    ①
brk:0>BRS 2 3000H <cr>                      ②
brk:0>TRY E BRS1 <cr>                       ③
brk:0>TRY D BRS2 <cr>                       ④
brk:0>TRM TRY <cr>                          ⑤
brk:0>BRM OFF <cr>                          ⑥
brk:0>RUN N <cr>                            ⑦
```

① Start address is set in BRS1.
② End address is set in BRS2.
③ BRS1 is specified as trace start point.
④ BRS2 is specified as trace end point.
⑤ Sectional trace is specified.
⑥ "No break condition" is set.
⑦ Executing emulation is specified.

(b) When neither a subcommand nor an operand is specified

The currently active settings are displayed.

```
brk:0>TRY <cr>

  Enable condition  :BRS1                   ①
  Disable condition :BRS2                   ②


brk:0>■
```

① BRS1 is set as trace start point.
② BRS2 is set as trace end point.

(c) Changing start and end point conditions

Different types of event condition registers are used for start and end point conditions.

```
brk:0>TRY <cr>                                          ①

 Enable condition  :OFF
 Disable condition :OFF

brk:0>TRY E BRA1 <cr>                                   ②

brk:0>TRY D BRS1 <cr>                                   ③

 TRY condition unmatch                                  ④

brk:0>■
```

① Displaying the current settings is specified.

② Start point condition is in an event condition register for data memory access related trace.

③ End point condition is in an event condition register for program fetch related trace.

④ Warning message indicating that an event condition register of different type is specified.
The setting is accepted, but it does not function.

8.4.46 Verify object command (VRY)

```
General format
  Format 1    VRY  △ [d:] file
```

brk:n>	o	emu:n>	x	trc:n>	x

Function:

The VRY command compares a specified object file with memory contents. If a mismatch occurs, its address and results are displayed.

Programming:

Operand

[d:]file: Specifies the name of a file containing object codes.
A drive number may be omitted.

Examples:

(a) When a mismatch does not occur

```
brk:0>VRY SAMPLE.HEX <cr>                    ①

  object verify complete                     ②

brk:0>■                                      ③
```

① File SAMPLE.HEX is compared with memory contents.
② Message indicating a normal end
③ Prompt indicating that a command is acceptable

(b) When a mismatch occurs

```
brk:0>VRY SAMPLE.HEX <cr>                    ①
  object verify                              ②
  Address File Memory
   0123    00    01
   1234    FF    FE
brk:0>■
```

① File SAMPLE.HEX is compared with memory contents.

② Message indicating a mismatch has occurred

(c) When a specified file is not found

```
brk:0>VRY SAMPLE.HEX <cr>
 object SAMPLE.HEX not found                 ①
brk:0>■
```

① Message indicating that file SAMPLE.HEX was not found

CHAPTER 9 OTHER FUNCTIONS

9.1 Overview of the RS-232-C Interface Function in the IE-75001-R

This section explains the RS-232-C interface provided for channels 1 and 2 in the IE-75001-R.

This section is intended for those who want to know the functions and specifications of channels 1 and 2 in detail. See Chapter 4 for connecting the IE-75001-R to the host machine or PROM programmer.

9.1.1 Signal lines for the RS-232-C interface

In the following descriptions, a device operating in the terminal mode is called a terminal, and a device in the modem mode is called a modem.

Table 9-1 lists the signal lines used in the IE-75001-R.

Table 9-1 RS-232-C Interface Signal Lines

No.	Type	Signal name		Function	Direction		Pin No.
					Modem	Terminal	
1	Ground	Frame Ground	FG	Protective ground			1
		Signal Ground	SG	Signal ground			7
2	Data	Transmitted Data	TxD	Line for sending data from terminal to modem	←		2
		Received Data	RxD	Line for sending data from modem to terminal		→	3
3	Static hand-shaking	Data Set Ready	DSR	Modem active-state signal line		→	6
		Data Terminal Ready	DTR	Terminal active-state signal line	←		20
4	Dynamic hand-shaking	Request To Send	RTS	Signal line for enabling data transmission from terminal to modem	←		(*) 4
		Clear To Send	CTS	Signal line for enabling transmission from modem to terminal		→	5

For details, see Section 9.1.3 Setting RTS.

Table 9-2 Pin number and RTS Signals

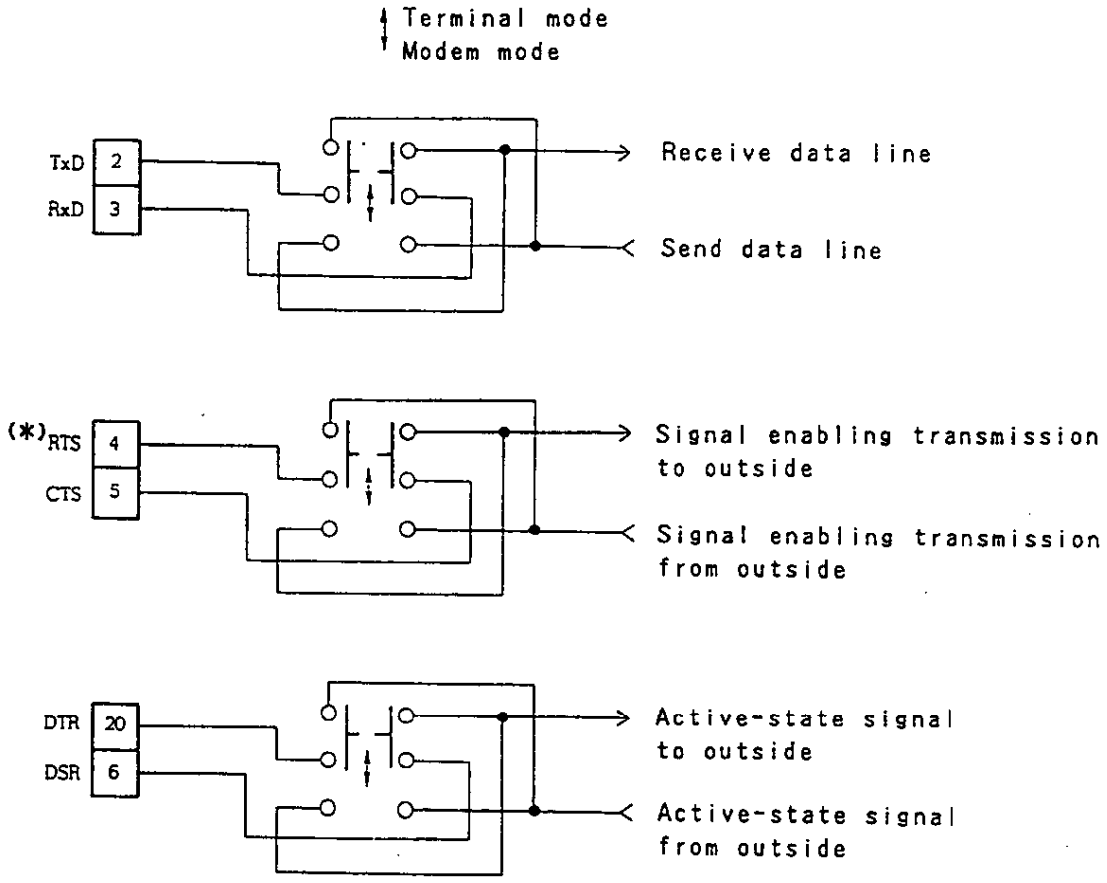
Pin No.	Signal name	Device to be connected
4	RTS N	Host machine, PROM programmer

9.1.2 Terminal mode and modem mode

The terminal mode can be switched to the modem mode or vice versa by changing the directions of the data, static handshaking, and dynamic handshaking signals.

The following circuit diagrams show how the modes are switched with slide switches.

Fig. 9-1 Circuit Diagrams for Switching between the Terminal and Modem Modes



* See Section 9.1.3 Setting RTS.

Caution: For the RS-232-C interface, be sure to connect a device operating in the terminal mode and a device in the modem mode.

Be careful not to connect devices both operating in the terminal mode or those both operating in the modem mode.

If devices in the same mode are connected, a collision of signals output from each device occurs, which may damage the interface driver in one of the devices.

9.1.3 Setting RTS

Usually, pin 4 is assigned to RTS in the RS-232-C interface. In some devices, however, signals having the same function as RTS are assigned to pins other than pin 4.

For hardware handshaking with such devices, the IE-75001-R allows the following RTS setting:

Table 9-3 RTS Setting

Selected RTS	RTS and FG setting				Device to be connected
	1	2	3	4 (*2)	
RTS N (*1)	ON	OFF	OFF	OFF	Host machine, PROM programmer

*1 Normally, this RTS setting is used.

*2 Position 4 of the DIP switch is set to ON to connect the signal ground (SG) line and frame ground (FG) line to a common point, or set to OFF to leave these signal lines open. Normally, position 4 is set to OFF.

9.1.4 Software handshaking and hardware handshaking

(1) Software handshaking (flow control)

(a) Connection for software handshaking

Connect signal lines as follows:

Table 9-4 Connection for Software Handshaking

Pin No.	Signal line type	Modem	Direction	Terminal
		Signal name		Signal name
1	Ground	FG	-	FG
7		SG	-	SG
2	Data	TxD	←	TxD
3		RxD	→	RxD

(b) Method for software handshaking

There is a 256-byte buffer to store received data. When received data have occupied half of the buffer capacity, Ctrl-S is sent to the sending destination to request it to stop data transfer. When the data in the buffer have decreased to a third of the buffer capacity, Ctrl-Q is sent to the sending destination to request it to resume data transfer.

If received data that cause the buffer to overflow are transferred between the issuance of Ctrl-S and stop of data transfer, data missing may occur.

(2) Hardware handshaking (CHAR: character-by-character handshaking)

(a) Connection for hardware handshaking

Connect the signal lines as follows. Unless the signal lines are connected properly, normal data transfer cannot be expected.

Table 9-5 Connection for Hardware Handshaking

Pin No.	Signal line type	Modem	Direction	Terminal
		Signal name		Signal name
1	Ground	FG	-	FG
7		SG	-	SG
2	Data	TxD	←	TxD
3		RxD	→	RxD
6	Static handshaking	DSR	→	DSR
20		DTR	←	DTR
4	Dynamic handshaking	RTS	←	RTS
5		CTS	→	CTS

(b) Method for hardware handshaking

Hardware handshaking is different from software handshaking in that hardware handshaking does not use a buffer. Data sending and reception are completely controlled with the signals listed in (a) above. This means that normal hardware handshaking is accomplished only when the signal lines are connected properly.

(i) Data sending

If RTS and DTR are both active, data are transmitted over RxD.

(ii) Data reception

To receive data, DSR must be left active. If the device becomes ready for data reception, CTS is made active. If the device is not ready, CTS is made inactive. With these signals, data transfer is controlled on a byte basis.

(3) Handshaking in channel 1 (using both hardware and software handshaking)

In channel 1, handshaking is performed by both hardware and software.

In principle, data are sent and received through software handshaking, and hardware handshaking is used for supplementary control to prevent data missing.

(4) Handshaking in channel 2

In channel 2, either hardware handshaking or software handshaking is selected with a command(Note).

Before selected handshaking is performed, observe the following:

Note: See the description of the set channel 2 mode command (MOD) in Section 8.4.23.

(a) When hardware handshaking is selected

If handshaking signals DSR, DTR, RTS, and CTS are not connected, normal handshaking is not accomplished. When hardware handshaking is selected, be sure to connect these handshaking signals properly.

(b) When software handshaking is selected

If data have been received in excess of the capacity of the buffer that stores serial data, data missing may occur. (See (b) of (1) in this section.)

9.1.5 Functions of channels 1 and 2

Table 9-6 lists the functions of channels 1 and 2.

Table 9-6 Functions of Channels 1 and 2

Item		Channel 1	Channel 2
Mode switching		Terminal/modem mode (switch-selected)	←
Baud rate		300, 600, 1200, 2400, 4800, 9600, or 19200 bps (switch-selected)	← (selected by software) (*)
Handshaking scheme		Hardware (on a character basis) and software (flow control)	Hardware (on a character basis) or software (flow control) (selected by software) (*)
Character specifications	Character length	8 bits The most significant bit (MSB) is 0 when output and ignored when input.	7 or 8 bits (selected by software) (*) When 8-bit character is selected, the most significant bit (MSB) is 0 when output and ignored when input.
	Parity bit	None	Even parity/odd parity/no parity bit (selected by software) (*)
	Stop bit length	2 bits	1 or 2 bits (selected by software) (*)

* For selection by software, see the description of the set channel 2 mode command (MOD) in Section 8.4.23.

9.2 Overview of IE-75001-R Parallel Interface Functions

This section explains the IE-75001-R parallel interface (channels 3 and 4).

The IE-75001-R has 8-bit parallel interfaces that use input data signals and interface control signals on the TTL level. The interface circuits conform to the Centronics standard.

This section is intended for those who want to know the functions of channels 3 and 4 in detail.

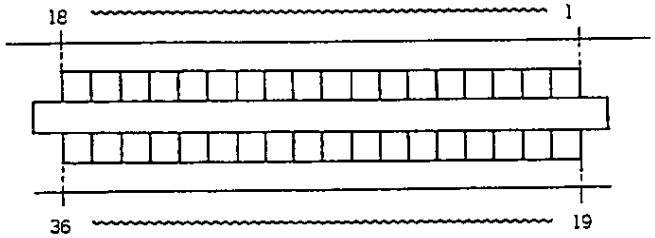
9.2.1 Signal lines for the parallel interface

Table 9-7 lists the parallel interface input signals and pin functions.

Table 9-7 Parallel Interface Signals

Pin No:	Signal name	Direction		Function
		Channel 3	Channel 4	
1	$\overline{\text{STB}}$	Output	Input	Strobe pulse signal for reading data
2	DATA 0	Output	Input	Parallel data 0
3	DATA 1	Output	Input	Parallel data 1
4	DATA 2	Output	Input	Parallel data 2
5	DATA 3	Output	Input	Parallel data 3
6	DATA 4	Output	Input	Parallel data 4
7	DATA 5	Output	Input	Parallel data 5
8	DATA 6	Output	Input	Parallel data 6
9	DATA 7	Output	Input	Parallel data 7
10	$\overline{\text{ACK}}$	Input	Output	Output upon completion of data input
11	BUSY	Input	Output	Signal disabling data reception
19 to 30 33	GND	—	—	Signal ground
12	PE	—	Input	Not used (pulled to +5 V with a 3.3-kΩ resistor)
32	$\overline{\text{ERROR}}$	—	Input	Not used (pulled to +5 V with a 3.3-kΩ resistor)

Fig. 9-2 Parallel Interface Pin Arrangement



9.2.2 Functions of channels 3 and 4 (high-speed download mode)

When the high-speed download mode (channels 3 and 4) is selected at start of the IE-75001-R, the following files can be downloaded from the host machine through the parallel interface at high speed:

Object file
Symbol file
Debugging environment file

Whenever a file is downloaded with a command other than the load command, data are sent to channel 3, bypassing channel 4. For example, when the MS-DOS PRINT command is used to output listing, it is output to the printer connected to channel 3, and it is not necessary to connect a printer to the PC-9800.

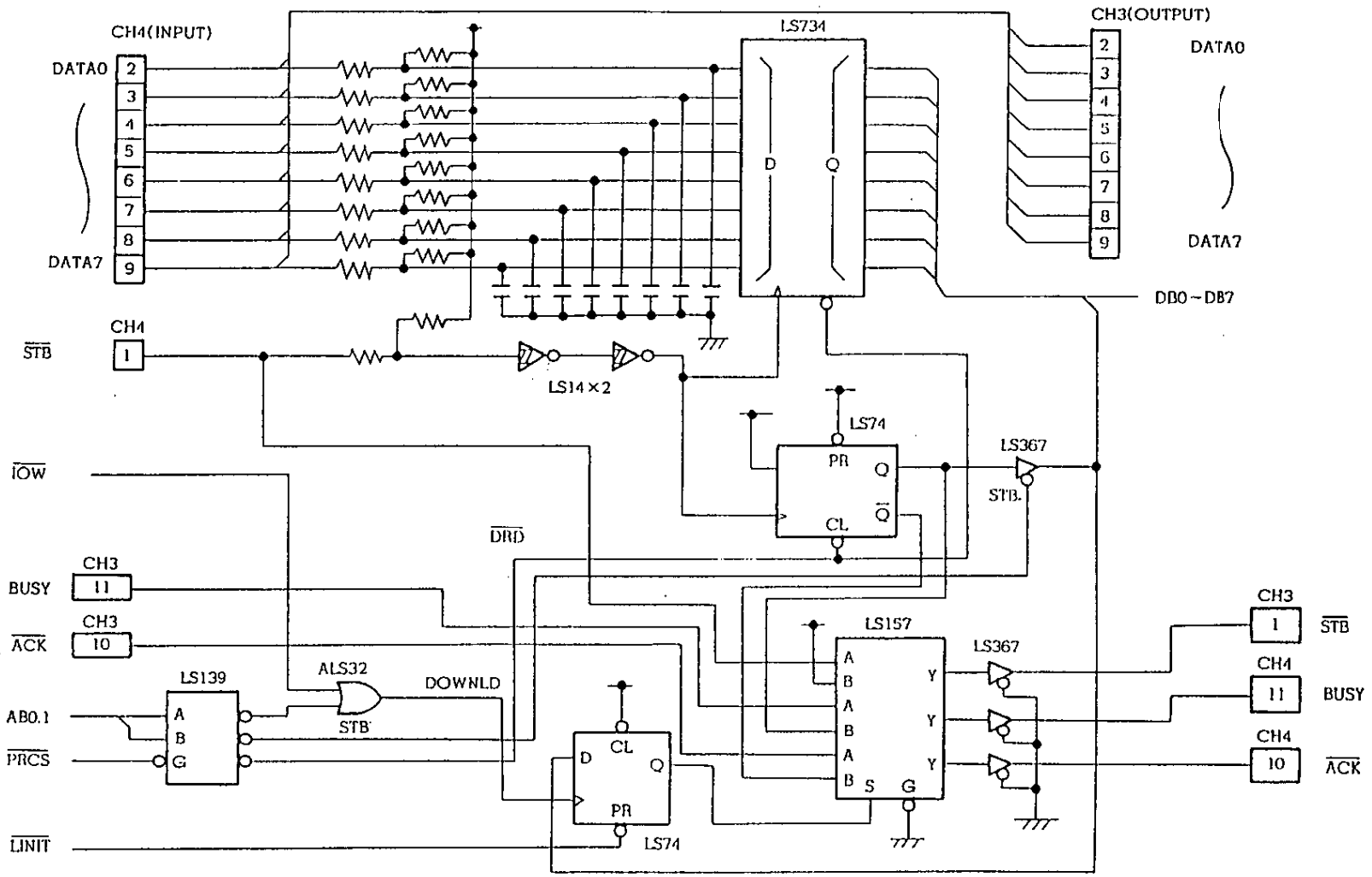
If the parallel interface is being used for list output or other processing with another command when a load command is issued, the following message appears, and downloading starts using channel 1 which is a serial interface:

Select Serial Interface

9.2.3 Parallel interface circuit

Figure 9-3 shows the parallel interface circuit.

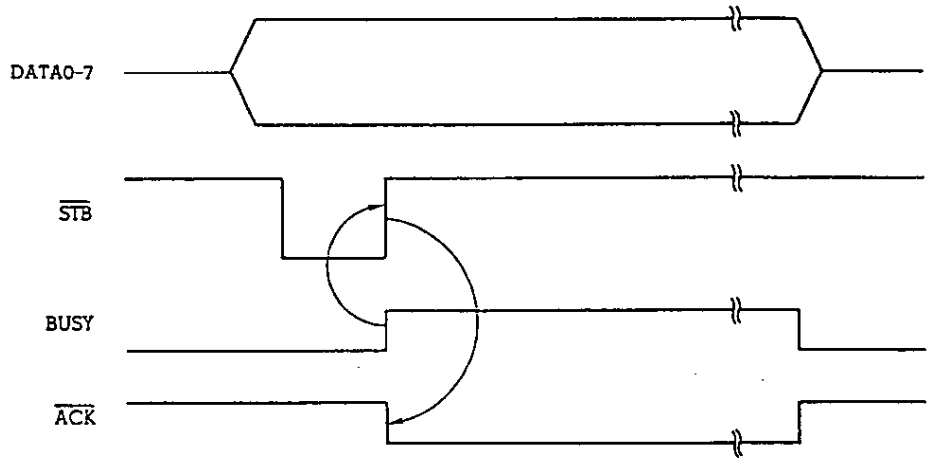
Fig. 9-3 Parallel Interface Circuit



9.2.4 Timing of high-speed downloading mode

Figure 9-4 shows the timing of the high-speed downloading mode.

Fig. 9-4 Timing of High-Speed Downloading Mode



APPENDIX A COMMAND LIST

Command type	Command body	Sub-command	Operand	Operation mode		
				brk:	emu:	trc:
Assemble	ASM	None	[addr] addr Assembly start address	o	x	x
Set bank	BNK	None	[n] n: 0≦n≦0FH n Memory bank selection register content	o	o	x
Set data memory event condition	BRA	{1, 2, 3, 4}	[MA={addr (*1), partition}] (*2) [V=data1][C=status][E=data2] addr Data memory address partition Data memory address range data1 Data value status Access condition data2 Data value input from external sense clip *1 BRA1, BRA2 Address or mask specification and partition specification are allowed. BRA3, BRA4 Only address or mask specification is allowed. *2 {R4, W4, RW4, R8, W8, RW8, SP}	o	o	x

o: Available
x: Not available

(Cont'd)

Command type	Command body	Sub-command		Operand	Operation mode		
					brk:	emu:	trc:
Display event information	BRK	None		None Caution: In the trace mode, this command cannot be used.	o	o	x
Set break mode	BRM	None		[BR?] [BR?] [BR?] [BR?] [BR?] [BR?] BR? Specifies event condition register name.	o	o	x
Set program memory address event condition	BRS	1	S	[addrx [, data]] [addrx [, data]]... [addrx [, data]] 4 at max. addrx Program memory address/address range data Data value input from external sense clip	o	o	x
			P	[addrx [, data]] [addrx [, data]]... [addrx [, data]] 7 at max. addrx Program memory address/address range data Data value input from external sense clip	o	o	x
		2		[addrx [, data]] addrx Program memory address/address range data Data value input from external sense clip	o	o	x

(Cont'd)

Command type	Command body	Sub-command	Operand	Operation mode		
				brk:	emu:	trc:
Set checkpoint	CHK	None	[{[BR?][BR?][BR?][BR?][BR?][BR?] / [BR?][BR?][BR?][BR?][BR?][BR?] / [BR?][BR?][BR?][BR?][BR?][BR?]} {[REG] / [SPR][SPR][SPR][SPR][SPR] / [addr][addr][addr][addr][addr]}] Operand 1 Operand 2 BR?: Specifies event condition register name. REG: Specifies general register to be checked and indicated. SPR: Specifies special register to be checked and indicated. addr: Specifies address of data memory location to be checked and indicated.	o	o	x
Select clock	CLK	None	[{U / I}] U Clock in user system I Clock in IE-75001-R	o	x	x
Create command file	COM	None	[{[d:]file / LST: / CON:}] [d:]file Opens file as command file. LST: Opens list device as command file. CON: Closes command file.	o	o	o
Display coverage measurement	CVD	D	[partition] partition Display range	o	x	x
Delete coverage measurement	CVD	K	None	o	x	x

(Cont'd)

Command type	Command body	Sub-command	Operand	Operation mode		
				brk:	emu:	trc:
Add coverage measurement range	CVM	A	[partition] partition Range to be added	o	x	x
Display coverage measurement range	CVM	D	None	o	x	x
Cancel specified coverage measurement range	CVM	K	[partition] partition Range to be canceled	o	x	x
Disassemble	DAS	None	[{addr / partition}] addr Disassembly start address partition Disassembly address range	o	x	x
Display directory	DIR	None	[[d:]file] [d:]file Directory display file name	o	o	o
Set event detection point	DLY	None	[{F / M / L}] F After specified condition is met, continues tracer operation until trace full state is entered. M After specified condition is met, continues tracer operation until data occupies a half of all frames of tracer. L Immediately after specified condition is met, stops tracer. TRIGGER POINT F M L TRACE MEMORY ------*->	o	o	x

(Cont'd)

Command type	Command body	Sub-command	Operand	Operation mode		
				brk:	emu:	trc:
DOS	DOS	None	None Valid only when system software is used. Entering EXIT<cr> returns to system software.	o	o	o
IE-75000-R control program	EXT	None	None	o	x	x
Display command history	HIS	None	None	o	o	o
Help	HLP	None	[command] command Command name	o	o	o
Load object/ symbol/debugging environment	LOD	None	[[d:]file[module(*)¥...][{C/D/S}] [d:]file File name module Symbol file module name C Specifies object. D Specifies debugging environment. S Specifies symbol file. * When IBM PC series host machine is used, \ must be coded instead of ¥.	o	x	x
Redirect output	LST	None	[{[d:]file/LST:/CON:}] [d:]file Opens file as output device. LST: Opens list device as output device. CON: Closes output device opened.	o	o	o

(Cont'd)

Command type		Command body	Sub-command	Operand	Operation mode		
					brk:	emu:	trc:
Math		MAT	None	expression	o	o	o
Manipulate program memory	Change memory contents	MEM	C	[addr] addr Start address for changing memory contents	o	x	x
	Display memory contents	MEM	D	[addr / partition] addr Start address for displaying memory contents partition Start/end address for displaying memory contents	o	x	x
	Test memory	MEM	E	[partition] partition Address range to be tested	o	x	x
	Initialize memory	MEM	F	partition data-string partition Memory range to be initialized data-string Initialization data	o	x	x
	Search memory	MEM	G	partition data-string partition Memory range to be searched data-string search data	o	x	x
	Copy memory contents	MEM	M	partition addr partition Range of source of copy addr Copy destination address	o	x	x

(Cont'd)

Command type		Command body	Sub-command	Operand	Operation mode brk:	emu:	trc:
Manipulate program memory	Exchange memory contents	MEM	X	partition addr partition addr: Memory range to be exchanged addr: Destination address of exchange	o	x	x
	Compare memory contents	MEM	V	partition addr partition addr: Memory range to be compared addr: Address of data with which memory contents are to be compared	o	x	x
Set channel 2 mode		MOD	None	[MODE={CHAR / FLOW}] [BAUD={19200 / 9600 / 2400 / 1200 / 600 / 300}] [LONG={7 / 8}] [PAR={OFF / EVEN / ODD}] [STOP={1 / 2}] Handshaking mode / Baud rate / Character length / Parity bit / Stop bit	o	x	x
External sense clip mode		OUT	None	[{OFF / ON $T / ON BRA?}] OFF: Performs external signal tracing (8 signals) and external signal level detection. ON $T: Performs external signal tracing (7 signals) and external event output (bit 0). ON BRA?: Outputs D-mem data (8 bits) indicated by BRA3 and BRA4, and performs tracing as well.	o	o	x

(Cont'd)

Command type		Command body	Sub-command	Operand	Operation mode		
					brk:	emu:	trc:
Set pass count		PAS	None	[count] count Pass count	o	o	x
Set terminal mode		PGM	None	[C] C Change of control character	o	x	x
Manipulate data memory	Change data memory contents	RAM	C	[{ addr[{ $B / $N }] / addrb }] addr Start address for changing data addrb Address of data to be changed $B In bytes $N In nibbles	o	x	x
	Display data memory contents	RAM	D	[{ { addr / partition } [{ $B / $N }] / addrb }] addr Display start address partition Data memory range to be displayed addrb Display address $B In bytes $N In nibbles	o	x	x
	Test data memory contents	RAM	E	[partition] partition Data memory range to be tested	o	x	x

(Cont'd)

Command type		Command body	Sub-command	Operand	Operation mode		
					brk:	emu:	trc:
Manipulate data memory	Initialize data memory contents	RAM	F	partition data-string[{ #B / $N }] partition: Data memory range to be initialized data-string: Initialization data	o	x	x
	Search data memory	RAM	G	partition data-string[{ $B / $N }] partition: Data memory range to be searched data-string: Search data	o	x	x
	Copy data memory contents	RAM	M	partition addr[{ $B / $N }] partition: Range of source of copy addr: Destination address of copy	o	x	x
	Exchange data memory contents	RAM	X	partition addr[{ $B / $N }] partition: Range for memory exchange addr: Destination address for exchange	o	x	x

(Cont'd)

Command type		Command body	Sub-command	Operand	Operation mode		
					brk:	emu:	trc:
Manipulate data memory	Compare data memory contents	RAM	V	partition addr[{$B / $N}] partition Data memory range to be compared addr Address of data with which data memory is to be compared	o	x	x
	Load data memory	RAM	L	None	o	x	x
	Save data memory	RAM	S	[partition][,partition]...[,partition] 5 at max. partition Data memory range to be saved	o	x	x
Manipulate register	Change register	REG	C	[register] register Register name	o	x	x
	Display register	REG	D	[{ALL / register}] ALL All registers in all banks register Register name	o	x	x
Reset		RES	None	[H] H Resets IE-75001-R.	o	o	o
Run emulation	Real-time execution without break condition	RUN	N	[addr] addr Execution start address	o	x	x

(Cont'd)

Command type		Command body	Sub-command	Operand	Operation mode		
					brk:	emu:	trc:
Run emulation	Real-time execution under break conditions	RUN	B	[addr] addr Execution start address	o	x	x
	Trace	RUN	T	[addr][(*1) { ,expression / , word }][TRD][REG] addr Execution start address word Number of steps to be traced TRD Displays trace data REG Displays registers *1 { register { = / > / < / => / >= / =< / <= / >< / <> } { master data(*2) / bit data / 4-bit data / 8-bit data / 16-bit data } } *2 When mask data are specified, conditions other than =, ><, and <> cannot be used.	o	x	x

Command type		Command body	Sub-command	Operand	Operation mode brk:	emu:	trc:
Save		SAV	None	[d:]file[partition][{C / D}] [d:]file — File name partition — Program memory area C — Object only D — Debugging environment only * Up to five	o	x	x
Switch emulated device mode		SET	None	[{STACK / SLWAIT}{OFF / ON}] STACK — Specifies the number of bytes to be stacked when target device interrupt occurs. OFF: 2 bytes ON: 3 bytes SLWAIT — Specifies wait time to be allotted after target device is reset. OFF: 31.25 ms ON: 7.81 ms	o	x	x
Manipulate special register	Change special register	SPR	C	[{group / register}] group — Group name register — Register name	o	x	x
	Display special register	SPR	D	[{group / register}] group — Group name register — Register name	o	x	x

(Cont'd)

Command type		Command body	Sub-command	Operand	Operation mode		
					brk:	emu:	trc:
Redirect input		STR	None	[d:]file[parameter list] [d:]file — File name parameter list — List of actual parameters	o	x	o
Stop real-time emulation		STP	None		x	o	o
				[T] T Stops tracer.	x	x	o
Select target device	Display device to be debugged	STS	C	None	o	x	x
	Display device to be debugged	STS	D	None	o	x	x
Manipulate symbol	Define append symbol	SYM	A	symbol word {N, C, D, B} symbol word — Symbol name / Symbol value N — Defines append symbol with number attribute. C — Define append symbol with code attribute. D — Define append symbol with date attribute. B — Define append symbol with bit attribute.	o	x	x

(Cont'd)

Command type		Command body	Sub-command	Operand	Operation mode		
					brk:	emu:	trc:
Manipulate symbol	Change append symbol value	SYM	C	symbol word symbol Symbol name word Symbol value	o	x	x
	Display symbol (append/ public/ module)	SYM	D	[module name¥]* module name¥ Module name * When IBM PC series host machine is used, \ is coded instead of ¥.	o	x	x
	Delete append symbol	SYM	E	symbol symbol Symbol name	o	x	x
	Delete all symbols	SYM	K	None	o	x	x
	Load append symbol	SYM	L	None	o	x	x
	Save append symbol	SYM	S	None	o	x	x
	Specify current module name	SYM	M	None	o	x	x

(Cont'd)

Command type	Command body	Sub-command	Operand	Operation mode		
				brk:	emu:	trc:
Restart system	SYS	None	None	o	x	x
Display trace data	TRD	None	[ALL][{$C / $F / $Q}][(*)][option[,option][,option]...[,option][Δ$B]] ALL Displays all trace data, then terminates command. $C Retrieves and displays checkpoints and trigger frame. $F Retrieves frame under retrieval condition specified with TRF and trigger frame, then displays retrieved data and its preceding and following lines, 5 lines in total. $Q Retrieves and displays frame under retrieval condition specified with TRF and trigger frame. * P0, P1, ...P15 Displays I/O port trace data. EXT Displays external sense clip data. DMEM Displays D-mem trace data (MA, MD, MRW).	o	o	x

(Cont'd)

Command type	Command body	Sub-command	Operand	Operation mode		
				brk:	emu:	trc:
Set trace data retrieval condition	TRF	None	[PA= { addrX / partition1 }] [PD=data1] [MA= { addr2 / partition2 }] [MD= data2] [MRW=status] [Pn=data3] [EXT=data4] addrX — P-mem address retrieval data partition1 — P-mem address retrieval data range data1 — P-mem data retrieval data addrX — D-mem address retrieval data partition2 — D-mem address retrieval data range data2 — D-mem data status — D-mem access status retrieval data MRD: Memory read MWR: Memory write MRW: Memory read-modify-write data3 — I/O port retrieval data data4 — External sense clip retrieval data	o	o	x
Trigger tracer	TRG	None	None	x	o	x
Select trace mode	TRM	None	[{ ALL / TRX / TRY }] [{ $M / $E }] ALL — Specifies unconditional trace. TRX — Specifies qualified trace. TRY — Specifies sectional trace. $M — Specifies machine cycle trace. $E — Specifies event cycle trace.	o	o	x

(Cont'd)

Command type	Command body	Sub-command	Operand	Operation mode		
				brk:	emu:	trc:
Manipulate trace-pointer	TRP	None	[{ word / F / L / T }] word Pointer displacement F Pointer points to top. L Pointer points to end. T Pointer points to trigger frame.	o	o	x
Set qualified-trace condition	TRX	None	[BR?][BR?][BR?][BR?][BR?][BR?] BR? Specifies event condition register name.	o	o	x
Set sectional trace condition	TRY	{ D / E }	[BR?][BR?][BR?][BR?][BR?][BR?] BR? Specifies event condition register name.	o	o	x
Verify object	VRY	None	[d:]file [d:]file File name	o	x	x

APPENDIX B LIST OF ERROR MESSAGES

No.		Error message and explanation
1	Message	Unrecognized command
	Explanation	The command keyword is invalid.
2	Message	Command format error
	Explanation	The command keyword is valid, but there is an invalid operand.
3	Message	Command/Data too long
	Explanation	A command or data of more than 127 characters were entered.
4	Message	Input data error
	Explanation	Invalid data were entered.
5	Message	System mode command
	Explanation	A command available in the system mode was entered in the stand-alone mode.
6	Message	Check sum error
	Explanation	A checksum error was detected while an object was being loaded or saved.
7	Message	Bad character
	Explanation	An invalid character was detected while an object was being loaded or saved.
8	Message	Aborted
	Explanation	The break key was pressed while an object was loaded or saved.
9	Message	Warning double define
	Explanation	The same module name was specified more than once with the LOD command.
10	Message	Bad file entry
	Explanation	An invalid file name is coded.

(Cont'd)

No.		Error message and explanation
11	Message	File overflow
	Explanation	An attempt to execute the LOD command was made, but the number of symbol files to be entered exceeded the maximum.
12	Message	Illegal record
	Explanation	An attempt to execute the LOD command was made, but the record format of the symbol table file was invalid.
13	Message	Load failed
	Explanation	An attempt to execute the LOD command was made, but an error was detected in the symbol or object code.
14	Message	Module overflow
	Explanation	An attempt to execute the LOD command was made, but the number of modules to be entered exceeded the maximum.
15	Message	Not loaded symbol
	Explanation	No symbol is loaded.
16	Message	Not found module record
	Explanation	The module name specified with the LOD command is not present in the symbol table file.
17	Message	File not found
	Explanation	The specified file name is not present.
18	Message	No symbol of append
	Explanation	There is no append symbol in the SYM_D or SYM_S command.
19	Message	Can not execute HLP command!
	Explanation	Neither the help file (IE75000.HLP) nor help overlay file (IE75000.OV2) is present on the current directory.

(Cont'd)

No.		Error message and explanation
20	Message	No. HLP file on the default drive
	Explanation	An attempt to execute the HLP command was made, but neither the help file (IE75000.HLP) nor help overlay file (IE75000.OV2) was found on the current directory.
21	Message	Keyword Error
	Explanation	An attempt to execute the HLP command was made, but the command keyword is invalid.
22	Message	Can not command abbreviation!
	Explanation	There is no overlay file for abbreviation (IE75000.OV1) on the current directory.
23	Message	File already exists.
	Explanation	An attempt was made to create a file with the same name as a file having the SYS or R/O attribute.
24	Message	Reserved file name
	Explanation	The specified file is reserved for the IE-75000-R control program.
25	Message	File name is used by other process
	Explanation	The specified file is already open.
26	Message	Can not close file-name
	Explanation	The indicated file could not be closed normally.
27	Message	Disk write error file-name
	Explanation	An error was detected during write for the indicated file.
28	Message	Disk read error file-name
	Explanation	An error was detected during read for the indicated file.

(Cont'd)

No.		Error message and explanation
29	Message	Can not open file-name
	Explanation	The specified file could not be opened.
30	Message	File make error file-name
	Explanation	The indicated file could not be created.
31	Message	Can not close file-name. Cancel xxx command
	Explanation	The indicated file could not be closed normally during execution of the xxx command (xxx: STR, LST, or COM).
32	Message	Disk write error file-name. Cancel xxx command
	Explanation	An error was detected in writing the indicated file during execution of the xxx command (xxx: LST or COM).
33	Message	Disk read error file-name. Cancel STR command
	Explanation	An error was detected in reading the indicated file during execution of the STR command.
34	Message	List device is used by other process
	Explanation	The list device is being used by another process (when the list device is specified in the COM and LST commands at a time).
35	Message	Append symbol file not found
	Explanation	The append symbol file (IE75000.SYM) is not present on the current directory when the SYM_L command is executed.
36	Message	Illegal append symbol file
	Explanation	An attempt to execute the SYM_L command was made, but the format of the append symbol file was invalid.

(Cont'd)

No.		Error message and explanation
37	Message	Communication error
	Explanation	The IE-75001-R could not communicate with the host machine successfully.
38	Message	Not found memories
	Explanation	Although external memory is specified, there is no available memory.
39	Message	Non map area access!
	Explanation	An attempt was made to access memory not mapped during execution of the ASM command.
40	Message	Assemble area over!
	Explanation	An access to a location out of the allowable memory range was attempted during execution of the ASM command.
41	Message	Disassemble area over!
	Explanation	An access to a location out of the allowable memory range was attempted during execution of the DAS command.
42	Message	Caution!
	Explanation	A generic object is generated. Or, special care is required.
43	Message	Error!
	Explanation	Object code cannot be generated. Or, there is an error.
44	Message	Warning!
	Explanation	Object can be generated, but its operation is unpredictable.
45	Message	Unexecutable command
	Explanation	An invalid command was entered during emulation. Or, a command available only during emulation was entered during break.

(Cont'd)

No.		Error message and explanation
46	Message	Non trace data
	Explanation	An attempt was made to execute the TRD command, but there was no trace data.
47	Message	Not found
	Explanation	The frame specified in the TRD command is not present.
48	Message	Trigger frame not found
	Explanation	TRD input was made in response to prompt, or a trigger frame was specified with TRP, but the trigger frame was not present.
49	Message	PORT xxH, xxH DATA xx xx E-CPU ERROR !!
	Explanation	The emulation device does not operate normally. (*) o When the user clock is used Confirm whether the user clock on the emulation board is set correctly. o When the internal clock is used Excess current or voltage may be applied to the emulation device through the probe. Delete the cause of the overload.
50	Message	Object load aborted
	Explanation	The Centronics is not connected when high speed down load is selected.
51	Message	Abnormal record aborted
	Explanation	The numbers, 0 through 9, are used for the head of the module name. Change the module name.

* If this message is displayed after the CLK U command is executed, the user or external clock supplied to the emulation device does not have enough amplitude and gain. Check the waveform of the clock. If the error occurs in other conditions, please contact an NEC engineer.

(Cont'd)

No.		Error message and explanation
52	Message	No connect Abort (Y or N) :
	Explanation	The IE-75000-R or IE-75001-R cannot be connected to the host machine. Check the setting of the RS-232-C: Check whether the RS232.DRV is incorporated to the device driver (in the case of PC-9800) Check whether the emulation board, break board are correctly inserted into the housing. Check whether the J1 and J2 cables are connected to the break board.

APPENDIX C NOTES ON CORRECT USE

C. 1 General

No.	Notes on correct use
1	Interrupt does not occur at location specified by CHK command
2	CHK data is not traced during DLY
3	Instruction that does not cause break but slips exists
4	STOP or HALT mode is not set when 1-step command is executed
5	One or two instructions are slipped if break condition set by BRA command is satisfied.
6	Instruction that is skipped is not correctly traced and displayed when qualify trace is executed
7	If break us specified for instruction to be skipped, break occurs even if that instruction is skipped and not executed
8	Interrupt does not occur during 1-step execution
9	Last line displayed is different from actual result of executing section trace
10	Instruction that does not operate in accordance with device specifications exists
11	Special register and general-purpose register whose trace result is different from actual execution result exist
12	User clock of subsystem cannot be used when uPD75028, 036*, P036, 048, P048*, 064*, 066*, 068*, or P068* is used

* Under development
Remarks: Numbers in this table correspond to the number of the detailed description on the following pages.

C.2 Detail of Noted on Correct Use

1. Interrupt does not occur at location specified by CHK command

   If an interrupt occurs at the same location as the condition set by the CHK command, the interrupt is accepted when the next instruction is executed. If the condition set by the CHK command is in succession, the interrupt is accepted after the last condition has been satisfied.

2. CHK data is not traced during DLY

   Trace data is not accepted even if the condition set by the CHK command is satisfied during DYK period.

3. Instruction that does not cause break but slips exists

   If a break condition is set for the following instructions, break does not occur immediately, but occurs when the next instruction is executed (i.e., the instruction is slipped.)

   If the instruction shown below is used successively, break does not occur immediately, but occurs after the instruction next to the one executed last is executed.

   ```
   SET1 FBX.X (EI FBX)  }
   CLR1 FBX.X (DI FBX)  }
   SEL  MBX             }
   ```

   Same applies when FBX.X, MBX area is accessed by MOV

4. Stop or HALT mode is not set when 1-step command is executed

   The Stop and HALT instruction is released immediately after they have been executed when a 1-step command is executed; therefore, the STOP or HALT status is not retained.

5. One or two instructions are slipped when break condition set by BRA command is satisfied

   (1) If memory write condition is set by BRA command as event condition
   Break is slipped as follows:

   ```
   1   If a 1-byte instruction follows the instruction for
       which the break condition is set

   -> Two instructions following the instruction for which
      the break condition is set are executed and then break
      occurs
   ```

Example:

```
MOV @HL, A          <- Break condition is satisfied here
MOV XA, BC
MOV DE, XA          <- Break occurs after this instruction
                       is executed
```

2 If a 2-byte or 3-byte instruction follows the instruction for which the break condition is set

-> One instruction following the instruction for which the break condition is set is executed and then break occurs

Example:

```
MOV @HL, A          <- Break condition is satisfied here
MOV DE, XA          <- Break occurs after this instruction
                       is executed.
```

(2) If memory read condition is set by BRA command as event condition

Break occurs after the next instruction has been executed (the number of bytes of the next instruction is irrelevant).

6. Instruction that is skipped is not correctly traced and displayed by means of qualify trace

Event if a skip operation has actually been performed, display to that effect is not made by means of qualify trace.

7. If break is set for instruction to be skipped, break occurs even when that instruction is skipped and not executed.

The instruction set by the break condition is actually skipped, and break occurs even if the instruction is not executed.

8. Interrupt does not occur during 1-step execution

If an interrupt request is generated during 1-step execution, the request flag (IRQxxx) is set, but the interrupt is not executed (the request flag is retained as is).

9. Last line displayed as result of section trace is different from actual result of execution

If BRS is specified as the disable condition of section trace, and if the instruction set as the disable condition is a 2-byte or 3-byte instruction, the result if the trace is not correctly displayed.

10. Instruction that does not operate in accordance with device specifications exists

The following instructions perform operations different from the specifications:

XCH A, @HL XCH A, @HL+ XCH A, @HL- XCH A, @rpal XCH XA, @HL XCH A, mem XCH XA, @HL	If address of following special registers is specified for these instructions, undefined operation is performed (device also performed undefined operation): F80H (SP), F82H (RBS), F83H (MSB), F84H (SBS), FB0H (PSW)

11. Special registers and general-purpose registers whose trace result is different from actual execution result (execution is normal)

The read data of the following special registers are different from the actual trace data, and undefined values are traced:

. BS (F82H)
. PSW (FB0H)

o If the value of a general-purpose register is read after data has been restored by the POP instruction and before the data is updated by the MOV instruction, etc., the actual value of the general purpose register is different from the value of the trace data, and an undefined value is traced (the actual values match, however).

12. User clock of subsystem cannot be used when uPD75028, 0386, P036, 048, P048, 064, 066, 068, or P068 is used

Use the internal subsystem clock.

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-889-1689

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑