To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

RENESAS

USER'S MANUAL

IE-78330-R

IN-CIRCUIT EMULATOR

SOFTWARE

Document No. EEU-1298B
(O. D. No. EEU-714C)
Date Published July 1991P
Printed in Japan

USER'S MANUAL

NEC

IE-78330-R

IN-CIRCUIT EMULATOR

SOFTWARE

The IE-78330-R conforms to the standards of VCCI which restricts radio disturbance in commercial and industrial areas.

Since the IE-78330-R used in residential districts or their neighboring districts may cause radio and TV disturbance, follow this manual.

MAJOR REVISIONS

Page	Description
pp.12-1 to 12-11	Addition of "CHAPTER 12 CAUTIONS"

PREFACE

Readers: This manual is intended for engineers who debug a system having a 78K/III series 16-/8-bit single-chip microcomputer by using the IE-78330-R.

The reader of this manual is assumed to be familiar with the functions and use of the uPD78330, uPD78334, and uPD78P334, and have a knowledge of the debugger.

Organization:

The IE-78330-R Manual consists of the following two parts: Software (this manual) and Hardware.

Software	Hardware
Function overview	Basic specifications
Description of commands	Configuring the system
	External interface functions

> Several cautions must be observed to use the IE-78330-R.
>
> See the summary in Chapter 12.
>
> For the most recent information about this product, please contact an NEC representative.

Purpose: After reading through this software manual, the reader will be able to understand a sequence of operations from starting of the IE-78330-R to the execution of commands in the development of a target device including debugging.

After reading through the hardware manual, the reader will be able to understand the basic specifications of the IE-78330-R and how to connect the IE-78330-R to external devices.

Guidance:

Software

- To obtain information on the overview of the IE-78330-R:
  See Chapter 1.

- To understand the basic operating procedure and functions:
  See Chapters 2 and 3.

- To understand the details of command input:
  See Chapters 5 through 7.

- To understand the types, functions, and input formats of commands:
  See Chapter 8.

- To understand the cautions on the use of the IE-78330-R:
  See Chapter 12.

Hardware

- To understand the basic specification:
  See Chapters 1 and 2.

- To set a user clock:
  See Chapter 3.

- To connect an external device to the IE-78330-R:
  See Section 1.3 and Chapters 4 through 6.

- To understand the differences between the target device and IE-78330-R target interface circuit:
  See Chapter 7.

- To understand the details of the functions of an IE-78330-R serial or parallel interface:
  See Chapters 8 or 9.

Terminology:

The following table shows the meaning of misunderstandable words 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 uPD78334 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.

Conventions:

- o Note: Explanation of an indicated part of text
- o Caution: Information requesting user's special attention
- o Remark: Supplementary information
- o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

See the explanation at the beginning of each chapter for other conventions.

Remark: A part of the start message varies with the type of host machine. The symbol indicating the end of a module name also varies with the type of host machine (¥ is used for PC-9800 series personal computers and a backslash (\) is used for IBM PC series).

SUMMARY OF CONTENTS

CONTENTS

CHAPTER 1 OVERVIEW

This chapter outlines of the IE-78330-R.

The IE-78330-R in-circuit emulator is a development tool to effectively debug hardware and software of the application system to which any of the uPD78330, uPD78334, and uPD78P334 has been applied.

1.1 Features

The IE-78330-R has:

- o The capability to emulate the uPD78330, uPD78334, or uPD78P334
  - . The emulator is available for every package if it is used with an optional target probe.
- o The capability to display and trace data in real time
  - . Many break and trace functions
  - . Capability to display the contents of a real-time trace without stopping the execution of the target device
  - . Capability to search the real-time tracer for the contents
  - . Capability to inputg an 8-bit trace with external sense clips
- o A function for analyzing a program
  - . Capability of displaying the executed contents of a memory area (coverage)
  - . Capability of tracing the value of internal RAM in a constant cycle (internal RAM sample)
- o The capability of symbolic debugging
- o The capability of on-line assembling and disassembling
- o An installed emulation memory (64K bytes)

- o The capability to download object and symbol files at high speed with the Centronics interface (Downloading with this interface is about ten times as fast as that with the RS-232-C interface.)

- o The capability of being used as another 78K series emulator by replacing the boards of the IE-78330-R with optional emulation and break boards.

1.2 Target Device

The following devices can be emulated by the IE-78330-R:

Table 1-1 Target Devices of IE-78330-R

Target device	Internal ROM	Internal RAM
uPD78330	ROMless	1024 bytes
uPD78334	32768 bytes (Mask ROM)	
uPD78P334	32768 bytes (PROM)	

1.3 Configuration

The IE-78330-R consists of the following units of hardware (cabinet and boards).

- Cabinet (new cabinet)
- Control/trace board
- Break board (IE-78330-R-BK)
- Emulation board (IE-78330-R-EM)

If the break and emulation boards of the IE-78330-R are replaced with optional boards, the system can be upgraded to another 78K series in-circuit emulator.

Remark: System upgrade refers to changing the system by replacing the emulation and break boards of the in-circuit emulator with optional boards so that another target device can be emulated.

1.4 System Upgrade from Another Function to IE-78330-R

If either the 75X or 78K series in-circuit emulators are currently used, the system can be upgraded to the emulator that has functions equivalent to that of the IE-78330-R. To upgrade the system, replace the emulation and break boards with the boards for the IE-78330-R.

Table 1-1 lists the models that can be upgraded and the boards required for the system upgrade.

The in-circuit emulators in hand are divided into two types: in-circuit emulators using a new cabinet and those using an old cabinet. Only in-circuit emulators using a new cabinet can be upgraded.

The control program for the IE-78330-R in-circuit emulator is required for the system upgrade from another function to the IE-78330-R by changing the boards.

Table 1-2 System Upgrade from Another Function to IE-78330-R

<table>
<tr><th rowspan="2">In-circuit emulator in hand</th><th rowspan="2">Cabinet</th><th colspan="2">Necessary optional boards</th></tr>
<tr><th>IE-78330-R-EM</th><th>IE-78330-R-BK</th></tr>
<tr><td>IE-78112-R</td><td rowspan="5">Old</td><td rowspan="5">x</td><td rowspan="5">x</td></tr>
<tr><td>IE-78210-R(*1)</td></tr>
<tr><td>IE-78220-R(*1)</td></tr>
<tr><td>IE-78310-R(*1)</td></tr>
<tr><td>IE-78310A-R</td></tr>
<tr><td>IE-75000-R</td><td rowspan="6">New</td><td rowspan="6">o</td><td rowspan="5">o</td></tr>
<tr><td>IE-78130-R</td></tr>
<tr><td>IE-78230-R</td></tr>
<tr><td>IE-78240-R</td></tr>
<tr><td>IE-78320-R</td></tr>
<tr><td>IE-78327-R(*2)</td><td>-</td></tr>
</table>

*1 Maintenance product

*2 Under development

Remarks 1. A new cabinet is used for the IE-78330-R.

2. o: Required

-: Unrequired

x: The system cannot be upgraded.

The use of the IE-78330-R is the same as that of the in-circuit emulator upgraded with optional boards such as IE-78330-R-EM.

1.5 Models that can be Used as a Host Machine and the OSs

This section explains host machines to be connected to the IE-78330-R for use and the OSs available for each host machine.

1.5.1 PC-9800 series

(1) Models

The following models can be used as a host machine in the PC-9800 series.

Table 1-3 Models Usable as a Host Machine in the PC-9800 Series

CPU	8086/V30	80286		80386	
Mode	Normal	Normal	High resolution	Normal	High resolution
Support models	No mark				
	E				
	F1/2/3				
	M2/3				
	VF2				
	VM0/2/4 / 21/11				
	U2				
	UV2/21/11				
	CV21				
			XA model 1/2/3/11/ 21/31		
	XL model 1/2/4	XL model 1/2/4	XL model 1/2/4		
	VX0/2/4/ 01/21/41	VX0/2/4/ 01/21/41			
	UX21/41	UX21/41			
	RX2/4/21/51	RX2/4/21/51			
	EX2/4	EX2/4			
	XL[2]			XL[2]	XL[2]
	RL2/5/21/51			RL2/5/21/51	RL2/5/21/51
	RA2/5/21/51			RA2/5/21/51	
	ES2/5			ES2/5	
	RS21/51			RS21/51	
	T model			T model	
	W2/W5/S5/F5			W2/W5/S5/F5	
	LV21/22				
	LX2/4/5	LX2/4/5			
	LS2/5			LS2/5	
	n				
	ns/-20			ns/-20	

Caution: 640K bytes or more are required for internal memory.

(2) OSs

MS-DOS Ver. 2.11/Ver. 3.10/Ver. 3.30/Ver.3.30A

(3) Control program distribution media

5-inch floppy disk (2HD)
3.5-inch floppy disk (2HD)

Caution: Distribution of a control program on an 8-inch floppy disk (2D) has been discontinued. Understand that the control program is distributed on a 5-inch floppy disk to users using an 8-inch floppy disk when the control program is upgraded.

1.5.2 IBM PC series

(1) Models

IBM PC/AT and IBM PC/XT

(2) OS

PC DOS Ver. 3.10

(3) Control program distribution medium

5-inch floppy disk (2D)

CHAPTER 2 STARTING THE IE-78330-R

This chapter explains the procedure to start the IE-78330-R.

To start the IE-78330-R, hardware used must have been set according to the instructions in "Hardware User's Manual" (EEU-713).

In this manual, the characters to be entered are written in uppercase. However, the system does not distinguish uppercase characters from lowercase characters.

If an invalid entry is made, the system outputs the same message again and waits for a correct entry.

Conventions

o ____: Indicates that the underlined item needs to be entered from the keyboard.

o <cr>: Indicates that the return key (CR (0DH)) needs to be pressed.

o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

2.1 Setting the Host Machine

(1) Checking the device driver

Start MS-DOS for a PC-9800 series personal computer or PC DOS for an IBM PC series personal computer and check that the RS-232-C and printer device drivers are installed.

If these drivers are not installed, add one line in the following format to CONFIG.SYS and install them.

DEVICE=path-name¥device-driver-name

Example: If MS-DOS is used, add the following two lines to CONFIG.SYS. However, RSDRV.SYS and PRINT.SYS is assumed to be in the root directory of the drive to be started.

```
DEVICE=RSDRV.SYS
DEVICE=PRINT.SYS
```

Caution: MS-DOS or PC DOS must be started again by resetting the host machine after the device drivers are installed.

Table 2-1 Host Machines and Corresponding Device Drivers

Host machine / Device driver	PC-9800 series	IBM PC series
RS-232-C device driver	RSDRV.SYS	Driver corresponding to the serial interface to be used
Printer device driver	PRINT.SYS	PRINT.SYS

When the high-speed download mode is not used, the printer device driver is not required. However, this means that the IE-78330-R control program does not require the printer device driver, but that some other programs may require it. If so, the printer device driver must be installed.

In some older versions of MS-DOS or PC DOS, the device driver need not be installed.

Check the version of your software according to the manual.

(2) Initializing the RS-232-C

When a PC-9800 series personal computer is used as the host machine, the SPEED command must be executed to initialize the RS-232-C.

When an IBM PC series personal computer is used as the host machine, the MODE command must be executed to initialize COM1.

Table 2-2 shows the settings. For details, refer to the MS-DOS or PC DOS manual.

Table 2-2 Initializing the RS-232C-0

Setting	Value
Baud rate	9600 bps
Character length	8 bits
Parity	None
Stop bits	2
X parameter	On or off

2.2 Starting the Control Program

The control program consists of the following files.

These files must reside in the current directory of the current drive.

. IE78330.EXE
. IE78330.HLP

To start the control program, enter IE78330 <cr> on a command line for MS-DOS or PC DOS.

The IE-78330-R must be turned on before the control program is started.

When the control program is started correctly, the following start message is displayed.

. Screen display example when the operational environment is set

```
A>IE78330 <cr>
  IE-78330 Controller (PC-9801 SERIES) Vx.x    [Dd Mmm Yy]
  Copyright (C) 1989 by NEC corporation
```

Remark: The above message is output when a PC-9800 series personal computer is used as the host machine. When an IBM PC series personal computer is used as the host machine, PC-9801 SERIES is replaced with IBM PC SERIES.

If the message "No Connect" appears on the screen, the IE-78330-R cannot normally be started yet.

When the next message "Abort (Y or N)" is displayed, enter Y <cr>.

. When the start message is not displayed

```
A>IE78330 <cr>
  IE-78330 Controller (PC-9801 SERIES) Vx.x    [Dd Mmm Yy]
  Copyright (C) 1989 by NEC corporation

  No Connect
  Abort (Y or N) : Y <cr>
```

If the IE-78330-R could not be started normally, one of the following causes may be considered:

① The power to the IE-78330-R is not turned on yet.
② The IE-78330-R is abnormally connected to the host machine.
③ The device driver is not installed yet.
④ The RS-232-C is not initialized yet. (The SPPED command is not executed yet.)

Check the above items. If one of them is detected, correct it and start the control program again.

If the IE-78330-R cannot be started by any means, contact an NEC special agent or sales agent from which the product was purchased.

2.3 Operations after Starting the IE-78330-R

This section explains the operations after the IE-78330-R was normally started.

(1) Request and check of turning on the power for the target system

When the control program is correctly started, a message is issued requesting a user to turn on the power for the target system.

After turning on the power for the target system, enter Y <cr>.

Enter Y <cr> even if the target system is not connected.

. Message requesting a user to turn on the power for the target system

Power on target system (Y/N) Y <cr>

Caution: If N <cr> is entered in response to the above message, the system displays the same message, and the user cannot proceed to the next step.

(2) Setting the operational environment

Once the environment conditions in Section 2.4 are set, these setting values can be set automatically the next time and later.

The system asks a user whether the user needs to change the previous set values. When the values need not be changed, enter N <cr>.

When one or more of the set values are to be changed, enter Y <cr> and set the values again.

. Message requesting a user to set the operational environment

Create new set up mode (Y or N) <u>Y <cr></u>

2.4 Setting the Operational Environment

(1) Setting the ROM size

The setting of the ROM size varies with the level of pin $\overline{EA}$ on the target system.

When the target system is not connected, it is assumed that the level of pin $\overline{EA}$ is high.

(a) When the level of pin $\overline{EA}$ is high

When the system displays the following message, enter the set value according to the ROM size of the target device.

. Message requesting a user to set the ROM size

Internal ROM size
(8K,16K,24K,32K,40K,48K,56K) =

Table 2-3 ROM Size Values

Target device	Set value
uPD78330	0K
uPD78334	32K
uPD78P334	32K

Enter 32K <cr> for a uPD78334 or uPD78P334.

Internal ROM size
(8K,16K,24K,32K,40K,48K,56K) = 32K <cr>

(b) When the level of pin EA is low (uPD78330)

The system displays the following message, but the ROM size is automatically set to 0.

. Message requesting a user to set the ROM size

Internal ROM size = 0K

(2) Setting the RAM size

When the system displays the following message, enter the set value (1024 bytes) according to the RAM size of the target device.

. Message requesting a user to set the RAM size

Internal RAM size
(256,384,512,640,768,896,1024,1152,1280) =

Table 2-4 RAM Size Values

Target device	Set value
uPD78330	1024
uPD78334	
uPD78P334	

Enter 1024 <cr>.

Internal RAM size
(256,384,512,640,768,896,1024,1152,1280) = 1024 <cr>

(3) Setting the uPD78330 + TAM emulation

If a uPD78334 is to be emulated using a uPD78330 and TAM (if an ASTB signal is already input to pin $\overline{EA}$ on the target system), enter Y <cr> in response to the following message.

. Message asking a user for the emulation using a uPD78330 and TAM

uPD78330 + TAM emulation (Y/N)

Enter Y <cr> (if the ASTB signal is already input to pin $\overline{EA}$).

uPD78330 + TAM emulation (Y/N) Y <cr>

(4) Setting the high-speed download mode

If the high-speed download mode is to be used, enter Y <cr> in response to the following message. Otherwise enter N <cr>.

. Message asking a user whether the user uses the high-speed download mode

Do you use high speed down load mode? (Y/N) = N <cr>

CHAPTER 3 FUNCTION OVERVIEW

This chapter explains the overview of the command functions and the setting of an operational environment for the IE-78330-R.

Conventions

o ____: Indicates that the underlined item needs to be entered from the keyboard.

o <cr>: Indicates that the return key (CR (0DH)) needs to be pressed.

o <ESC>: Indicates that the escape key needs to be pressed.

o ^: Indicates that the character on the right side of a caret (^) needs to be pressed while the control key is held down.

o ■: Indicates the cursor.

o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

3.1 Setting the Environment for the Target Device

This section explains the setting of the environment for the target device when the IE-78330-R is connected to the host machine (a PC-9800 or IBM PC series personal computer).

(1) Setting the operational environment
(2) Clock selection function
(3) Mapping

3.1.1 Setting the operational environment

The following message is displayed after the power for the target system is turned on.

. Create new set up mode (Y or N) :

Enter Y <cr> to set the operational environment.

The set operational environment is stored in the setup file (SETUP.STR).

When the user need not change the set values the next time and later, enter N <cr> in response to the above message. The operational environment is then automatically set from the setup file.

The following four items must be set.

(a) Setting the ROM size
(b) Setting the RAM size
(c) Setting the uPD78330 + TAM emulation
(d) Setting the high-speed download mode

The following shows a screen display example when the operational environment is set.

o Developing a uPD78330, uPD78334, or uPD78P334 (screen display example)

```
       A>IE78330 <cr>
         IE-78330 Controller (PC-9801 SERIES) Vx.x  [Dd Mmm Yy]
           Copylight (C) 1989 by NEC corporation

(1)      Power on target system (Y/N) Y <cr>
(2)      Create new set up mode (Y or N) : Y <cr>
(3)      Internal ROM size (8K, 16K, 24K, 32K, 40K, 48K, 56K) = 32K <cr>
(4)      Internal RAM size (256, 384, 512, 640, 768, 896, 1024, 1152,
                            1280) = 1024 <cr>
         Tracer  initialize
         Breaker initialize
(5)      uPD78330 + TAM emulation (Y/N) N <cr>
(6)      Do you use high speed down load mode? (Y/N) = Y <cr>
       brk:0> ■
```

(1) Request and check of turning on the power for the target system

(2) Setting the operational environment

(3) Setting the ROM size

(4) Setting the RAM size

(5) Setting the uPD78330 + TAM emulation

(6) Setting the high-speed download mode

(1) Request and check of turning on the power for the target system

When the control program is correctly started, a message is issued requesting a user to turn on the power for the target system.

After turning on the power for the target system, enter Y <cr>.

Enter Y <cr> even if the target system is not connected.

. Message requesting a user to turn on the power for the target system

Power on target system (Y/N) Y <cr>

(2) Setting the operational environment

Once the environment items (3) through (6) are set after the system is started, these setting values can be set automatically the next time and later when the values need not be changed.

The system asks a user whether the user needs to change the previous set values. When the values need not be changed, enter N <cr>.

When one or more of the set values are to be changed, enter Y <cr>.

. Message requesting a user to set the operational environment

Create new setup mode (Y/N) Y <cr>

(3) Setting the ROM size

The setting of the ROM size varies with the level of pin $\overline{EA}$ on the target system.

When the target system is not connected, it is assumed that the level of pin $\overline{EA}$ is high.

(a) When the level of pin $\overline{EA}$ is high

When the system displays the following message, enter 32K <cr> for uPD78334 emulation.

. Message requesting a user to set the ROM size

Internal ROM size
(8K,16K,24K,32K,40K,48K,56K) = 32K <cr>

(b) When the level of pin $\overline{EA}$ is low

The system displays the following message, but the ROM size is automatically to 0.

. Message requesting a user to set the ROM size

Internal ROM size = 0K

(4) Setting the RAM size

When the system displays the following message, enter 1024 <cr> for the emulation of a uPD78330, uPD78334, and uPD78P334.

. Message requesting a user to set the RAM size

Internal RAM size
(256,384,512,640,768,896,1024,1152,1280) =
1024 <cr>

(5) Setting the uPD78330 + TAM emulation

If a uPD78334 is to be emulated using a uPD78330 and TAM (if the ASTB signal is already input on pin $\overline{EA}$ in the target system), enter Y <cr> in response to the following message.

. uPD78330 + TAM emulation (Y/N) Y <cr>

When Y <cr> is entered, mapping the memory space of the target device except the internal RAM space and SFR space is released forcibly. (Non-mapping)

For details, refer to uPD78330, uPD78334, and TAM (uPD71P302) User's Manuals.

(6) Setting the high-speed download mode

If the high-speed download mode is to be used, enter Y <cr> in response to the following message. Otherwise enter N <cr>.

. Message asking a user whether the user uses the high-speed download mode

  Do you use high speed down load mode? (Y/N) = N <cr>

3.1.2 Clock selection function (CLK command)

The CLK command is used to select an operation clock that is supplied to the emulation CPU.

Executing the CLK command resets the emulation CPU.

One of the following clock sources is selected.

(1) Clock in the emulator (CLK I)

When the quartz oscillator connected to an emulation CPU provides a frequency of 16 MHz, use a CLK I command (select the clock in the emulator).

(2) User-set clock (CLK U)

When debugging is to be done at a frequency other than 16 MHz, use a CLK U command (select the user-set clock).

In this case, install a quartz oscillator which provides double of the set frequency to the clock socket (OPCK) on the emulation board in the IE-78330-R.

No clock can be supplied from the external clock generator (the target system) to the IE-78330-R.

Example: When an emulation CPU is to be operated at 12 MHz, install a 24 MHz quartz oscillator.

For details of clock selection, refer to Chapter 3 in "IE-78330-R In-circuit Emulator User's Manual: Hardware."

3.1.3 Mapping function (MAP command)

The MAP command specifies memory to be used by a target device.

Executing the MAP command resets the emulation CPU.

The following six memory attributes can be specified.

- o Internal ROM (MAP I command)
- o Turbo access manager alternate memory (MAP T command)
- o Alternate memory (MAP W command)
- o Write-protected alternate memory (MAP R command)
- o User memory (MAP U command)
- o Mapping release (non-mapping) (MAP K command)

For a 56K-byte memory area from 0H to 0DFFFH, specify the following six memory attributes in 8K bytes.

- o Internal ROM
- o Turbo access manager alternate memory
- o Alternate memory
- o Write-protected alternate memory
- o User memory
- o Mapping release

For an 8K-byte memory area from 0E000H to 0FFFFH except the internal RAM and SFR(Note) area, specify the following two memory attributes.

- o User memory
- o Mapping release

Note: SFR stands for special function register.

(1) Internal ROM (MAP I command)

The memory area specified for internal ROM is treated in the same way as for the ROM of a target device.

The emulation CPU accesses memory in the IE-78330-R.

When the emulation CPU attempts to write data into this memory area, a write protect break occurs. See Section 3.2.5 for details.

The MAP I command can be used to change the internal ROM size set during start of the IE-78330-R.

Normally, the mapping capacity of the internal ROM is set to the ROM size specified during start of the IE-78330-R.

(2) Turbo access manager alternate memory (MAP T command)

The memory area specified for turbo access manager alternate memory is treated in the same way as when TAM is connected to the target device.

The emulation CPU accesses memory in the IE-78330-R.

When the emulation CPU attempts to write data into this memory area, a write protect break occurs. See Section 3.2.5 for details.

(3) Alternate memory (MAP W command)

The memory area specified for alternate memory is treated in the same way as when a normal memory such as a static or dynamic RAM is connected to the target device.

The emulation CPU accesses memory in the IE-78330-R.

When the emulation CPU makes a turbo access to the memory area (this is a high-speed burst program fetch), a turbo access break occurs. See Section 3.2.5 for details.

(4) Write-protected alternate memory (MAP R command)

The memory area specified for write-protected alternate memory is treated in the same way as when the ROM is connected to the target device.

The emulation CPU accesses memory in the IE-78330-R.

When the emulation CPU attempts to write data into this memory area, either of the following two breaks occurs. See Section 3.2.5 for details.

. Turbo access break: For attempts of turbo access
. Write protect break: For attempts of write operation

(5) User memory (MAP U command)

The memory area specified for user memory is used for access to memory in the target system.

The emulation CPU accesses memory in the target system.

When the emulation CPU makes a turbo access to the memory area (this is a high-speed burst program fetch), a turbo access break occurs. See Section 3.2.5 for details.

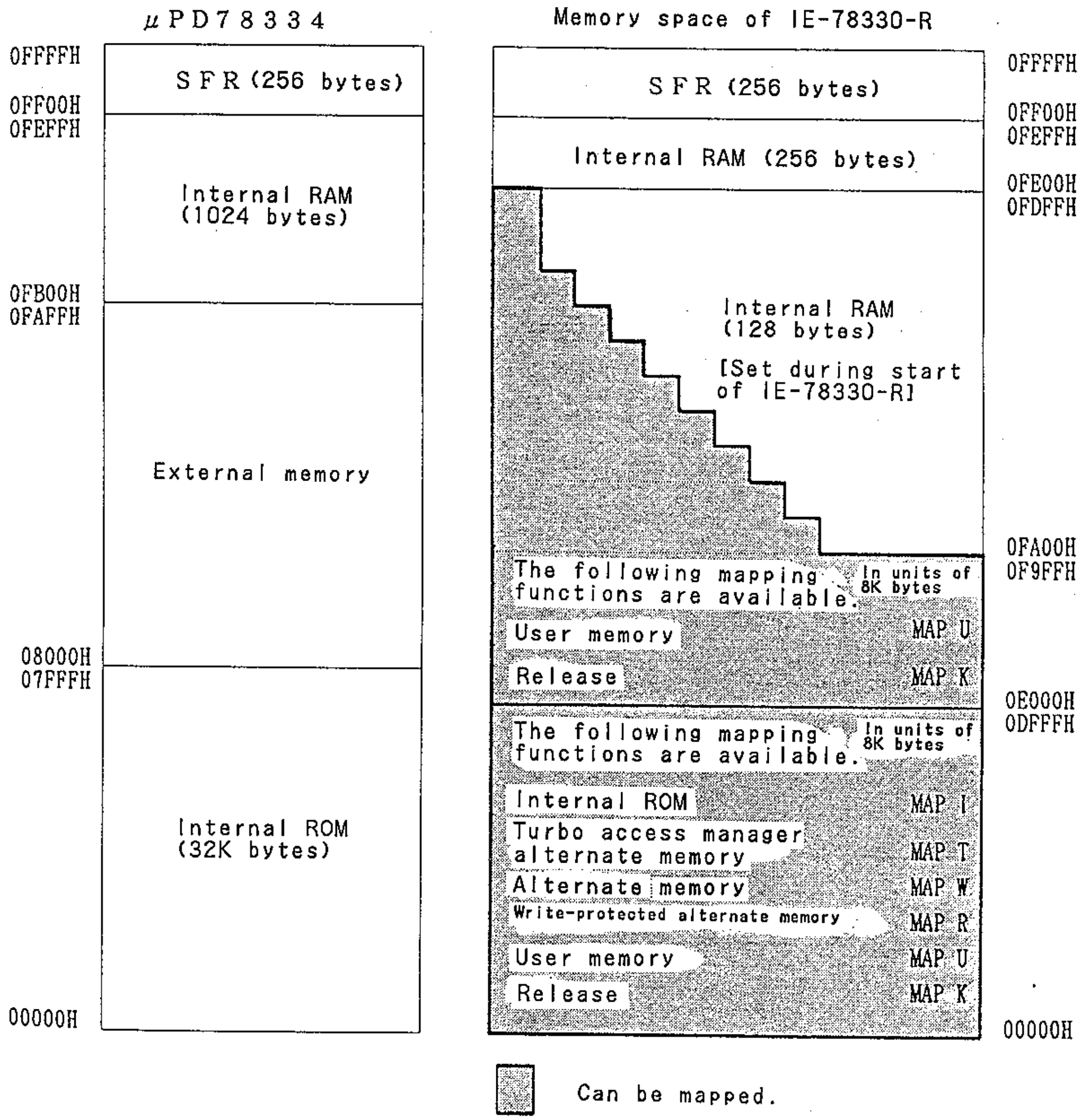
(6) Mapping release (MAP K command)

The memory area specified for mapping release is an unassigned memory area.

When the emulation CPU attempts to access this memory area, a non-map access break occurs. See Section 3.2.5 for details.

o Figure 3-1 shows the memory space of a target device and the IE-78330-R.

Fig. 3-1 Memory Space of Target Device and IE-78330-R



3.2 Emulation

The IE-78330-R has the following six emulation functions.

(1) Reset function
(2) Memory manipulation function
(3) Register manipulation function
(4) Execution function
(5) Break function
(6) Operation status display function

3.2.1 Reset function (RES command)

There are two reset functions (RES commands).

(1) Resetting only the emulation/device (RES)

The RES command resets the emulation device.

Information about mapping, memory, events, and the analyzer is not affected.

(2) Resetting the IE-78330-R system (RES H)

Control returns to the setting of the operational environment during start of the IE-78330-R.

3.2.2 Memory manipulation function (ASM, DAS, MEM, and MOV commands)

Various memory manipulation functions are provided such as display, change, and transfer for memory the emulation CPU can operate.

The following commands are available.

- o Change or display with hexadecimal numbers (MEM)
- o Change or display with mnemonic (ASM, DAS)
- o Transfer of data between alternate memory and user memory (MOV)

Memory manipulation is done for memory areas specified by the mapping functions.

3.2.3 Register manipulation function (REG and SFR commands)

The contents of an internal register or SFR of a uPD78330, uPD78334, or uPD78P334 can be displayed or changed.

The REG command is used for internal registers.

For PSWs, data can be displayed or changed in units of flags.

The SFR command is used for special function registers (SFRs).

3.2.4 Execution function (RUN command)

The RUN command starts execution of a program by the emulation CPU.

There are two execution functions.

(1) Real-time execution
(2) Nonreal-time execution

(1) Real-time execution function

There are two types of real-time execution functions.

- o Nonbreak real-time execution (RUN N command)
- o Real-time execution under break conditions (RUN B command)

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

This function starts executing a target program from a specified address and stops the real-time tracer and internal RAM data sampler if a delay condition is satisfied. See Figure 3-2.

The executing target program can be stopped only through forced breaks such as fail-safe breaks (input of the STP command).

Fig. 3-2 Operation of Hardware during Real-time Execution without Breaks

Hardware		Event		Execution start RUN N ↓	Enable condition ENB ↓	Qualified trace condition TRX ↓	Trigger condition BRM ↓	Check point condition CHK ↓		Disable condition DSB ↓	Pass condition PAS ↓	Delay condition DLY ↓	Analyzer start TRG ↓	Analyzer stop STP T ↓	Emulation termination STP ↓
	Emulation CPU	Operation	Stop	Real-time execution	→	→	→	Stop	Real-time execution	→	→	→	→	→	Stop
	Emulation CPU	Interrupt	Held	Accepted	→	→	→	Held	Accepted	→	→	→	→	→	Held
	Prompt		brk:0>	trc:0>	→	→	→	→	→	→	→	emu:0>	trc:0>	emu:0>	brk:0>
	Event detector		Stop	Detected	→	→	→	Stop	Detected	→	Stop	→	Detected	Stop	→
Analyzer	Real-time tracer	All trace mode	Stop	Trace	→	→	→	Stop	Trace	→	→	Stop	Trace	Stop	→
Analyzer	Real-time tracer	Section trace mode	Stop	→	Trace	→	→	Stop	Trace	Stop	→	→	→	→	→
Analyzer	Real-time tracer	Qualified trace mode	Stop	→	→	Trace Stop	→	→	→	→	→	→	→	→	→
Analyzer	Real-time tracer	Check point trace mode	Stop	→	→	→	→	Trace	Stop	→	→	→	→	→	→
Analyzer	Internal RAM data sampler	Sampler memory	Stop	Writing at sample timing	→	→	→	Stop	Writing at sample timing	→	→	Stop	Writing at sample timing	Stop	→
Analyzer	Performance evaluation	Execution time measurement	Stop	→	Measure	→	→	Stop	Measure	Stop	→	→	→	→	→
Analyzer	Performance evaluation	Executed instruction count	Stop	→	Measure	→	→	Stop	Measure	Stop	→	→	→	→	→
Analyzer	C0 coverage		Stop	Measure	→	→	→	Stop	Measure	→	→	→	→	→	Stop
	External trigger output		Low	→	→	→	→	→	→	→	High	Low	→	→	→

Remark: The arrows in the table indicate that the operation on the left continues.

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

This function starts executing a target program from the specified address and stops the emulation CPU, real-time tracer, and internal RAM data sampler if a delay condition is satisfied. See Figure 3-3.

To forcibly stop the execution of the analyzer and emulation CPU, enter an STP command.

When the emulation CPU and analyzer are stopped because of the satisfied delay condition, the system automatically enters the one-step execution mode.

When <cr> is pressed in this mode, the next instruction (one step) is executed.

When / <cr> is entered, the procedure is executed (see (b) of (2) in this section).

To suspend the one-step execution, press the escape key.

Fig. 3-3 Operation of Hardware during Real-time Execution with Breaks

Hardware \ Event				Execution start RUN B ↓	Enable condition ENB ↓	Qualified trace condition TRX ↓	Trigger condition BRM ↓	Check point condition CHK ↓		Disable condition DSB ↓	Pass condition PAS ↓	Delay condition DLY ↓	Re-execution <cr>or/<cr> ↓		Emulation terminations <ESC> ↓
Emulation CPU		Operation	Stop	Real-time execution	→	→	→	Stop	Real-time execution	→	→	Stop	Step execution Procedure execution	→	Stop
Emulation CPU		Interrupt	Held	Accepted	→	→	→	Held	Accepted	→	→	Held	→	→	→
Prompt			brk:0>	trc:0>	→	→	→	→	→	→	→	None	→	→	brk:0>
Event detector			Stop	Detected	→	→	→	Stop	Detected	→	Stop	→	→	→	→
Analyzer	Real-time tracer	All trace mode	Stop	Trace	→	→	→	Stop	Trace	→	→	Stop	All trace All trace for main routine	→	Stop
Analyzer	Real-time tracer	Section trace mode	Stop	→	Trace	→	→	Stop	Trace	Stop	→	→	All trace All trace for main routine	→	Stop
Analyzer	Real-time tracer	Qualified trace mode	Stop	→	→	Trace / Stop	→	→	→	→	→	→	All trace All trace for main routine	→	Stop
Analyzer	Real-time tracer	Check point trace mode	Stop	→	→	→	→	Trace	Stop	→	→	→	→	→	→
Analyzer	Internal RAM data sampler	Sampler memory	Stop	Writing at sample timing	→	→	→	Stop	Writing at sample timing	→	→	Stop	→	→	→
Analyzer	Performance evaluation	Execution time measurement	Stop	→	Measure	→	→	Stop	Measure	Stop	→	→	→	→	→
Analyzer	Performance evaluation	Executed instruction count	Stop	→	Measure	→	→	Stop	Measure	Stop	→	→	→	→	→
Analyzer	C0 coverage		Stop	Measure	→	→	→	Stop	Measure	→	→	→	Measure	→	Stop
External trigger output			Low	→	→	→	→	→	→	→	High	Low	→	→	→

Remarks 1. The arrows in the table indicate that the operation on the left continues.
2. <cr> indicates the point at which the return key is pressed.

(2) Nonreal-time execution function

There are two types of nonreal-time execution functions.

- o Step execution (RUN T command)
- o Procedure execution (RUN T command with PRC)

(a) Step execution (RUN T command)

This function executes one instruction in a target program from a specified address, then performs tracing.

Step execution continues intermittently until the specified register conditions match or the execution of a specified number of instructions is completed.

When step execution ends, the system automatically enters the one-step execution mode.

When <cr> is pressed in this mode, the next instruction (one step) is executed.

When / <cr> is entered, the procedure is executed (see (b) of (2) in this section).

To suspend the execution, press the escape key.

Remark: All interrupts are put on hold during step execution.

Fig. 3-4 Operation of Hardware during Step Execution

Hardware \ Event				Command input RUN T ↓	Re-execution <cr>or/<cr> ↓		Emulation termination <ESC> ↓
Emulation CPU		Operation	Stop	Step execution	Step execution / Procedure execution	→	Stop
Emulation CPU		Interrupt	Held	→	→	→	→
Prompt			brk:0>	None	→	→	brk:0>
Event detector			Stop	→	→	→	→
Analyzer	Real-time tracer	All-trace mode	Stop	Trace	All trace / Main routine trace	→	Stop
Analyzer	Real-time tracer	Section trace mode	Stop	Trace	All trace / Main routine trace	→	Stop
Analyzer	Real-time tracer	Qualified trace mode	Stop	Trace	All trace / Main routine trace	→	Stop
Analyzer	Real-time tracer	Check point trace mode	Stop	→	→	→	→
Analyzer	Internal RAM data sampler	Sampler memory	Stop	→	→	→	→
Analyzer	Performance evaluation	Execution time measurement	Stop	→	→	→	→
Analyzer	Performance evaluation	Executed instruction count	Stop	→	→	→	→
Analyzer	C0 coverage		Stop	Measure	→	→	Stop
External trigger output			Low	→	→	→	→

Remark: The arrows in the table indicate that the operation on the left continues.

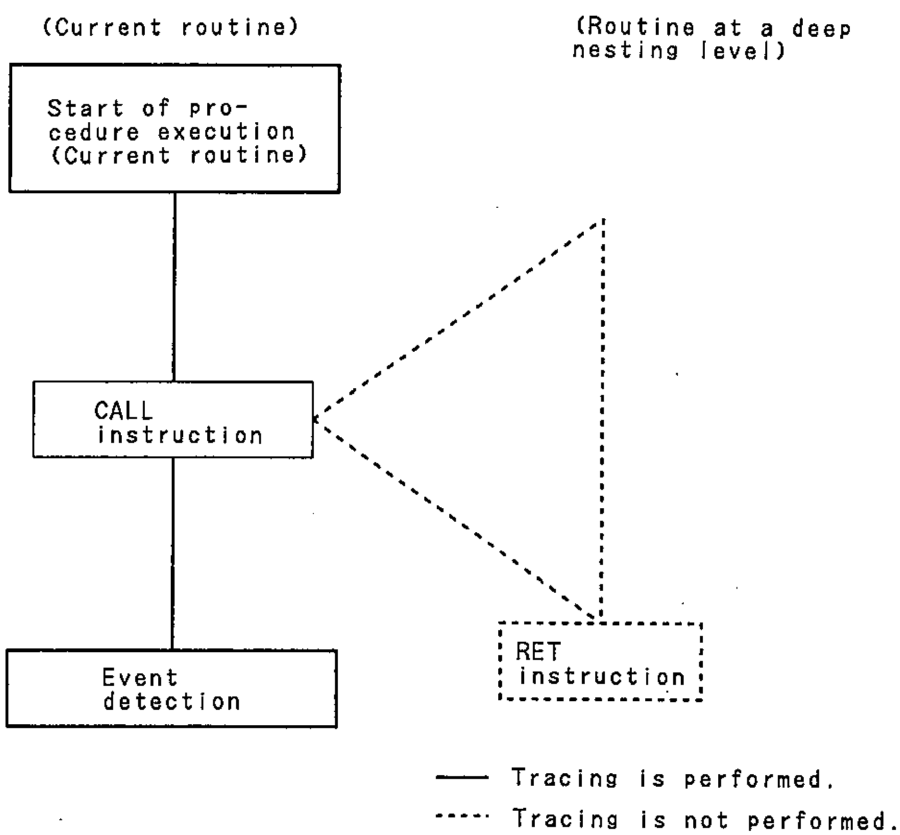
(b) Procedure execution (RUN T command with PRC)

This function effectively debugs the current routine and the routine whose nesting level is shallower than that of the current routine, that is, the main routine.

The function executes one instruction in a target program from a specified address to perform tracing.

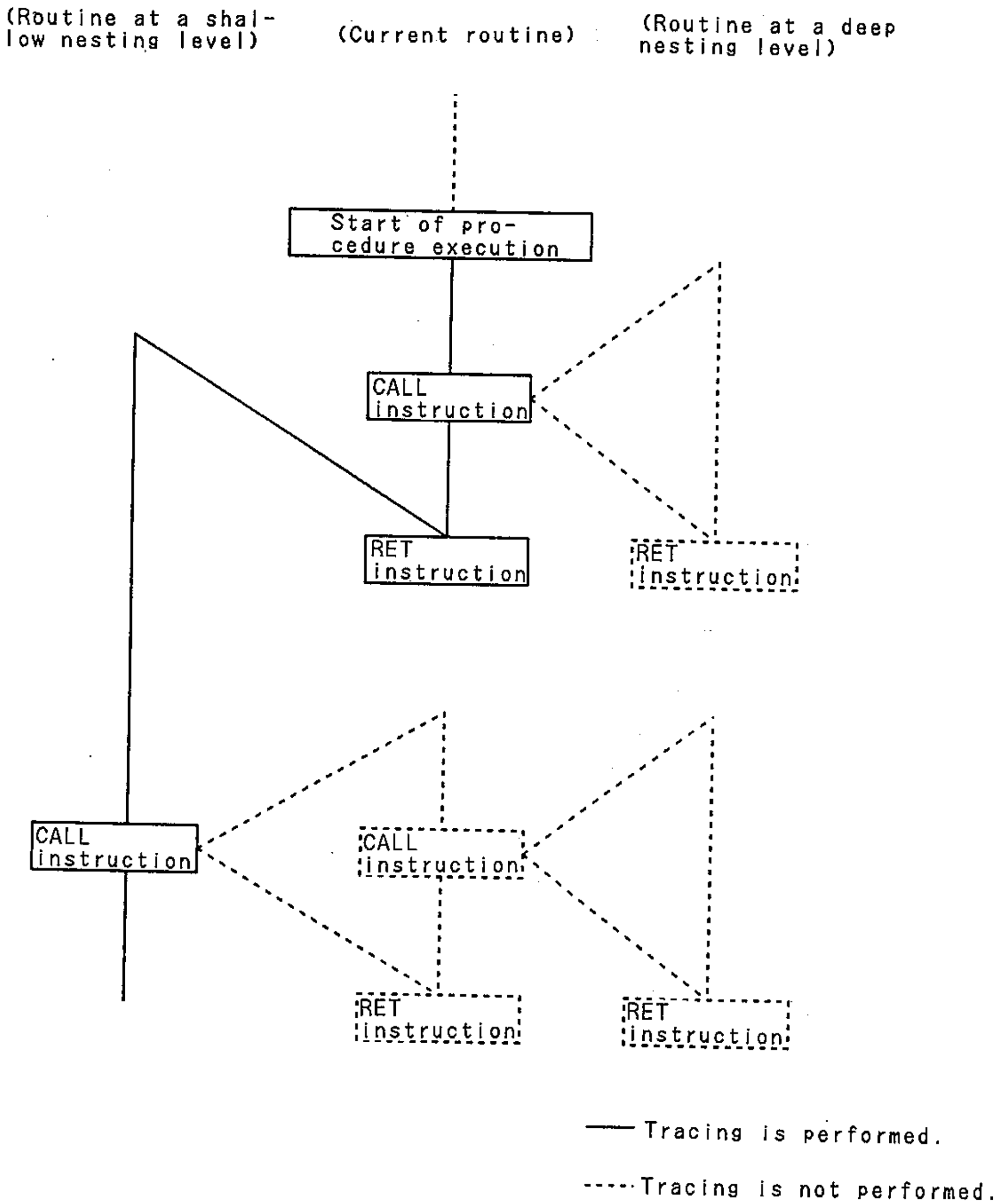
Step execution is performed and tracing is not performed for the routine, for example the routine enclosed by CALL and RET instructions, whose nesting level is deeper than that of the routine from which procedure execution started, or the current routine. See Figure 3-5 below.

Fig. 3-5 Procedure Execution and Nesting



Control can return to a routine whose nesting level is shallower than the nesting level of the routine from which procedure execution started (current routine). See Figure 3-6 below.

Fig. 3-6 Procedure Execution when Returning to the Routine at a Shallow Nesting Level



Procedure execution continues intermittently until the specified register conditions match or the execution of a specified number of instructions is completed.

No event is detected and all interrupts are put on hold during procedure execution.

When procedure execution ends, the system automatically enters the one-step execution mode.

When <cr> is pressed in this mode, the next instruction (one step) is executed.

When / <cr> is entered, the procedure is executed.

To suspend the execution, press the escape key.

Fig. 3-7 Operation of Hardware during Procedure Execution

Hardware \ Event				Command input RUN T	Re-execution <cr>or/<cr>		Emulation termination <ESC>
Emulation CPU	Operation		Stop	Procedure execution	Step execution / Procedure execution	→	Stop
Emulation CPU	Interrupt		Held	→	→	→	→
Prompt			brk:0>	None	→	→	brk:0>
Event detector			Stop	→	→	→	→
Analyzer	Real-time tracer	All-trace mode	Stop	All trace of main routine	All trace / Main routine trace	→	Stop
Analyzer	Real-time tracer	Section trace mode	Stop	All trace of main routine	All trace / Main routine trace	→	Stop
Analyzer	Real-time tracer	Qualified trace mode	Stop	All trace of main routine	All trace / Main routine trace	→	Stop
Analyzer	Real-time tracer	Check point trace mode	Stop	→	→	→	→
Analyzer	Internal RAM data sampler	Sampler memory	Stop	→	→	→	→
Analyzer	Performance evaluation	Execution time measurement	Stop	→	→	→	→
Analyzer	Performance evaluation	Executed instruction count	Stop	→	→	→	→
Analyzer	C0 coverage		Stop	Measure	→	→	Stop
External trigger output			Low	→	→	→	→

Remark: The arrows in the table indicate that the operation on the left continues.

3.2.5 Break function (STP and RES commands and ESC key)

The STP and RES commands and the ESC key stop the execution of a program by the emulation CPU.

There are four break functions.

- o Event detection break by the hardware comparator
- o Break by a register value during step execution
- o Fail-safe break
- o Manual break

The table below shows the relationship between the break functions and the execution functions.

Table 3-1 Relationship between Break and Execution Functions

Execution function / Break function	Nonbreak real-time execution (RUN N)	Real-time execution with breaks (RUN B)	Step execution (RUN T)	Procedure execution (RUN T with PRC)
Event detection break	Invalid	Valid	Invalid	Invalid
Break by a register value	Invalid	Invalid	Valid	Valid
Fail-safe break	Valid	Valid	Valid	Valid
Manual break	Valid	Valid	Valid	Valid

(a) Event detection break

An event detection break is valid only during execution of a RUN B command.

There are three event trigger sources.

. Bus event detection (BRA 1 to BRA 4)
. External data detection (BRD)
. Program execution (BRS 1 to BRS 4)

The messages shown in Table 3-2 are displayed when the execution of an emulation CPU stops through event detection.

Table 3-2 Messages Displayed for Event Detection Breaks

Event trigger source	Message
Bus event detection	Bus detection break terminated (The name of the detector for the event trigger source is displayed after "terminated.")
External data detection	External break terminated
Program execution detection	Execution break terminated (The name of the detector for the event trigger source is displayed after "terminated.")

(b) Break by a register value during step execution

The following break conditions can be specified in the operand.

. Match of internal register conditions, completion of the execution of a specified number of instructions (RUN T)

The above two conditions are set in the operand of a RUN T command.

This break can be specified only in this command (RUN T).

The following message is displayed when the execution of an emulation CPU stops by a command line.

. terminated

(c) Fail-safe break

A fail-safe break is a break which occurs forcedly when the emulation CPU malfunctions.

A fail-safe break is valid for all the execution functions.

The fail-safe break is classified into the following four types.

Fail-safe break
- Nonmap area
- Write protect
- Turbo access
- Manual (Note)
  - Execution stop command (STP)
  - Reset command (RES)
  - ESC key

Note: A manual break is one of the fail-safe break functions, but it is caused by an operator, not a machine. For details of the manual break, see item (d) below.

Table 3-3 lists the messages displayed when the execution of an emulation CPU stops through a fail-safe break.

Table 3-3 Messages Displayed for Fail-safe Breaks

Event trigger source	Message
Nonmap area	Non map area break terminated
Write protect	Write protect break terminated
Turbo access	Turbo access break terminated
Manual	Escape break terminated

o Nonmap area break

A nonmap area break occurs when an attempt is made to access a memory area which is not mapped.

o Write protect break

A write protect break occurs when an attempt is made to write data into one of the following memory areas.

. Internal ROM (MAP I)
. Turbo access manager alternate memory (MAP T)
. Write-protected alternate memory (MAP R)

o Turbo access break

A turbo access break occurs when a turbo access is attempted to one of the following three memory areas.

. Write-protected alternate memory (MAP R)
. Alternate memory (MAP W)
. User memory (MAP U)

Only the following areas are available for turbo access operations.

. Internal ROM (MAP I)
. Turbo access manager alternate memory (MAP T)
. Internal RAM (except 0FE00H to 0FFFFH)

(d) Manual break

A manual break is caused by an operation through key entry. This break is valid for all the execution functions.

There are three manual breaks.

o Forced break with an execution stop command (STP)

When an execution stop command (STP) is entered, the emulation CPU and analyzer operations stop.

A forced break caused with an STP command is valid only during real-time execution.

The internal RAM and SFR are retained in the state immediately before a break.

o Forced break with a reset command (RES)

When a reset command (RES) is entered, the emulation device and analyzer operations stop.

A forced break caused with an RES command is valid only during real-time execution.

The emulation device is reset when the execution of the program stops.

o Forced break with the ESC key

When <ESC> is pressed, the emulation device and analyzer operations stop.

A forced break caused by pressing the escape key is valid only during nonreal-time execution.

The internal RAM and SFR are retained in the state immediately before a break.

3.2.6 Operation status display function

The prompt to be displayed (see below) depends on the operation statuses of the emulation CPU and the analyzer.

- trc:0>
- emu:0>
- brk:0>

Table 3-4 lists the operation statuses of the Emulation CPU and analyzer corresponding to each prompt.

Table 3-4 Prompts and Operation Statuses of Emulation CPU and Analyzer

Displayed prompt	Emulation CPU operation	Analyzer operation	Remark
trc:0>	Running	Running	tracing
emu:0>	Running	Stopped	emulating
brk:0>	Stopped break	Stopped	break

Commands can be entered only when one of the above prompts is displayed.

3.3 Event Detection Function (BRA, BRD, BRS, DSB, ENB, CHK, TRX, and DLY Commands)

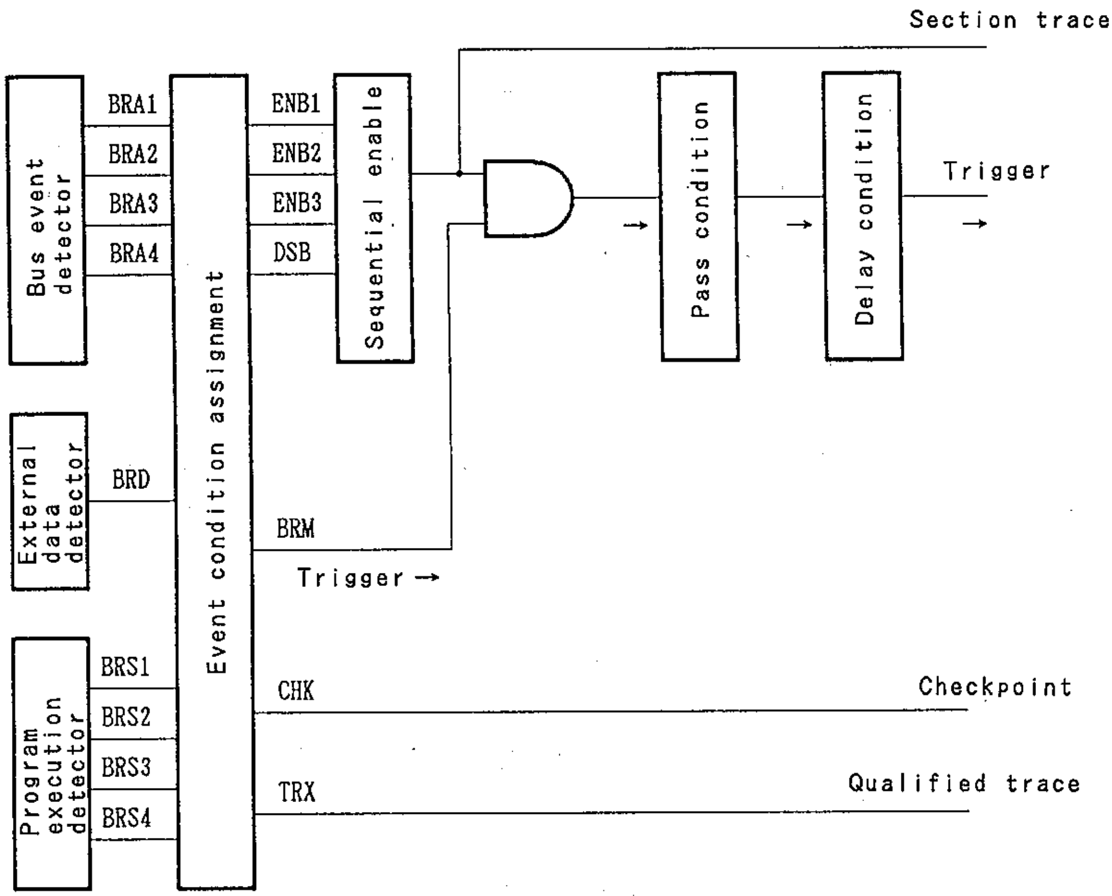
The event detection function monitors the running state of the emulation CPU at all times, and outputs a trigger signal under specified conditions to trigger the emulation CPU and stop the analyzer operating.

The event detection function is effective only when the emulation CPU runs in real time.

The event detection function provides the following capabilities:

(1) Bus event detection (BRA)
(2) External data detection (BRD)
(3) Program execution detection (BRS)
(4) Event condition assignment (BRM, ENB1 to ENB3, DSB, CHK, TRX)
(5) Pass condition (PAS)
(6) Delay condition (DLY)

Fig. 3-8 Block Diagram of the Event Detection Section



3.3.1 Bus event detection (BRA command)

There are four event detectors (BRA1 to BRA4) made up of comparators connected to the bus of the emulation CPU.

Read and write of data, and program (OP code) fetch are detected at the beginning of the bus cycle.

A program fetch is thus detected at a point when prefetch is performed. (When a program fetch is to be detected at execution of an instruction, the BRS command (program execution detection) must be used.)

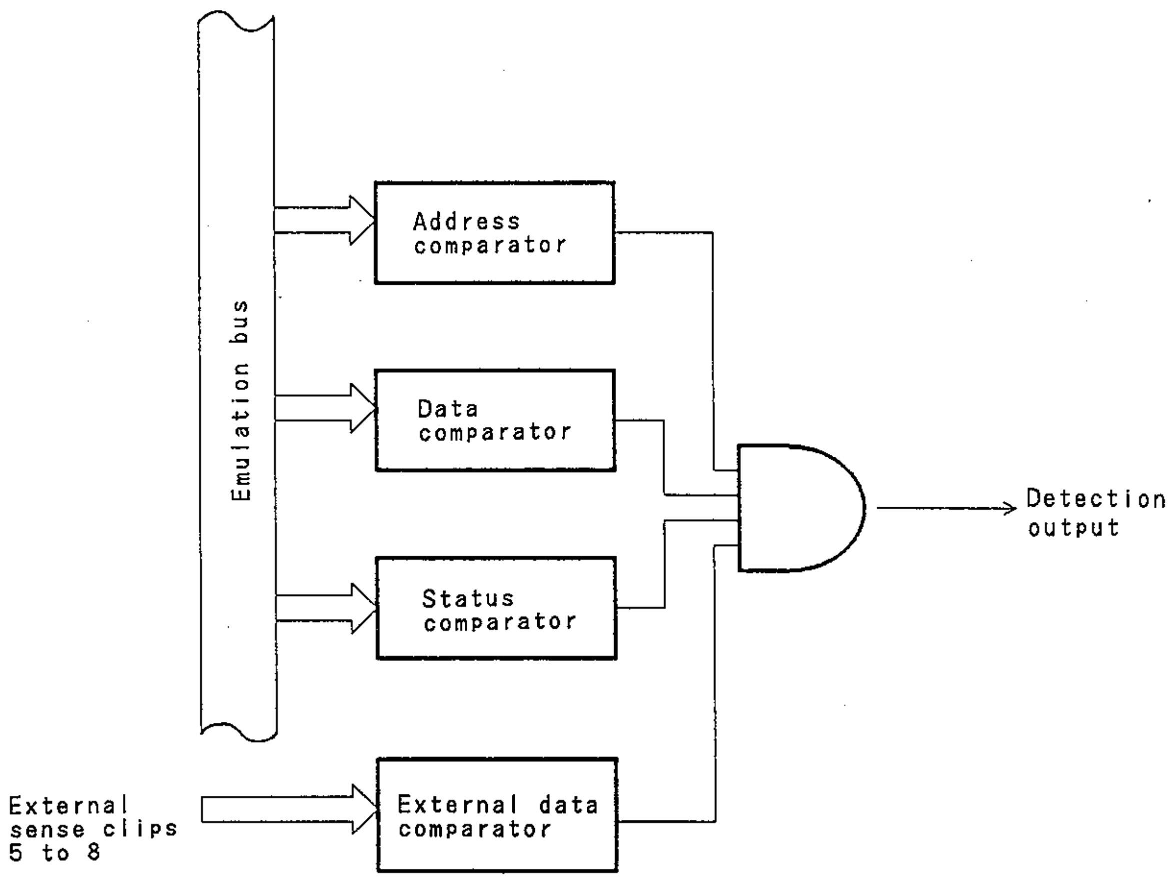
The following conditions of bus event detection can be set with one event detector:

- . Two addresses
- . Data (byte)
- . Detection status
- . External data of four bits

— ANDed

Remark: Addresses, data, and external data can be specified with masking.

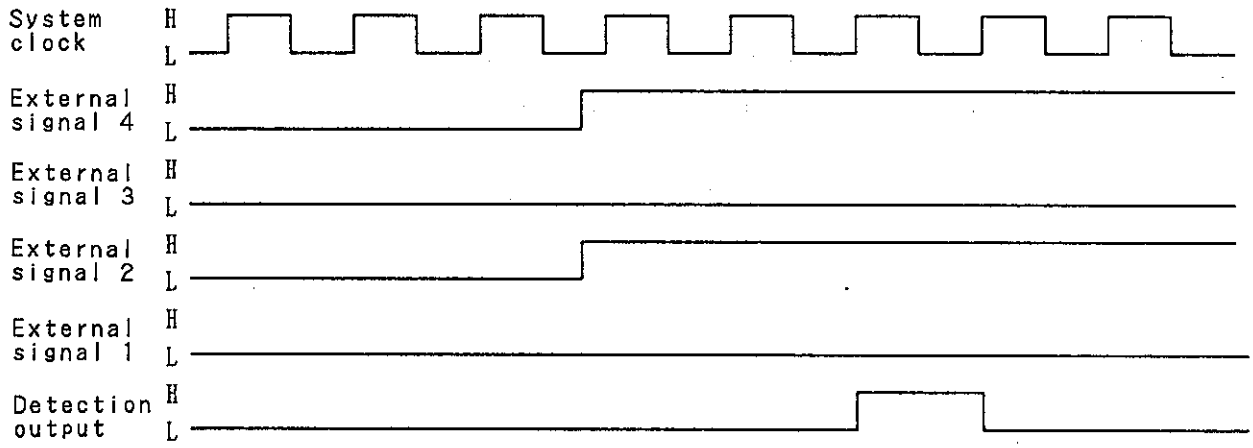
Fig. 3-9 Block Diagram of the Bus Event Detector



3.3.2 External data detection (BRD command)

A signal applied to external sense clips 1 to 4 is detected. Figure 3-10 shows the timing of detection.

Fig. 3-10 Detecting External Data 1010_Y ($0A_H$)



. When external data have shown a specified value for at least two system clocks, those data are detected (edge detection).

Mask specification is allowed for data to be detected.

3.3.3 Program execution detection (BRS command)

Four event detectors (BRS1 to BRS4) are provided to detect program execution. The following condition is set for detecting program execution with one event detector:

. One address

3.3.4 Event condition assignment (BRM, ENB1 to ENB3, DSB, CHK, and TRX commands)

Outputs of event detectors are assigned to different functions.

o Trigger condition (BRM command)

When the trigger condition is detected, the emulator operates as follows:

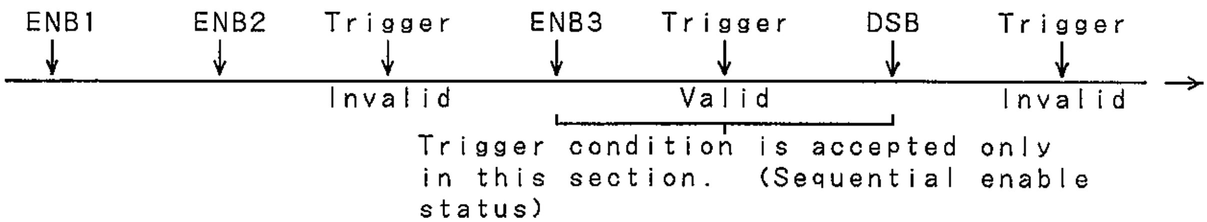
. The analyzer stops.
. The emulation CPU causes a break (only in real-time execution under break conditions).
. A trigger signal is output (when the trigger signal is enabled).

o Sequential enable (ENB1 to ENB3, and DSB commands)

The trigger condition is enabled or disabled.

A trigger condition becomes valid when it occurs after an enable point has been passed and until a disable point is passed.

If more than one enable point is set, that is, sequential enable is specified, a trigger condition becomes valid only when it occurs after the enable points have been passed in specification order.



If the periodic trace mode is selected, a trace operation performed only in the section of the sequential enable status.

- Checkpoint (CHK command)

  When the emulator detects an event assigned to a checkpoint, it temporarily breaks the execution of the user program. The emulator then writes the contents of registers, memory, and SFRs in trace memory under conditions set beforehand, then restarts the target program.

  The data written by the emulator are displayed as a checkpoint when trace data are displayed. (This function is called a snap shot dump.)

- Qualified trace (TRX command)

  When an event assigned to qualified trace is detected in a bus cycle, only that bus cycle is traced. Only bus event detectors (BRA1 to BRA4) can perform normal qualified trace operation.

3.3.5 Pass condition (PAS command)

Events are counted. When a set value has been reached, the delay counter is enabled and started for delay counting. If a delay condition is set with the DLY L command, the analyzer stops immediately. (For details of DLY L, see Section 3.3.6.)

If sequential enable specification is made, events detected in the enable state are counted.

3.3.6 Delay condition (DLY command)

The trace area of the real-time tracer or internal data sampler is specified with respect to the preset trigger point. This command determines to which location tracing should be continued after the trigger point is traced.

To select a trace area, the trigger point is actually set to one of the first, middle, and last of trace or sample memory.

. First location of trace or sample memory (the trigger point and following part are traced) DLY F
. Middle location of trace or sample memory (the part around the trigger point is traced) DLY M
. Last location of trace or sample memory (the trigger point and preceding part are traced) DLY L

(1) First location of trace or sample memory (the trigger point and following part are traced)

```
Trigger point    ↓
                 F        M        L
                 |        |        |
Trace memory    -*------------------->
                 |← Trace area →|
```

In this case, trace or sample data from the trigger point can be referenced.

(2) Middle location of trace or sample memory (the part around the trigger point is traced)

```
Trigger point             ↓
                 F        M        L
                 |        |        |
Trace memory    ----------*---------->
                 |← Trace area →|
```

Trace or sample data around the trigger point can be referenced.

(3) Last location of trace or sample memory (the trigger point and preceding part are traced)

```
Trigger point                              ↓
                 F          M          L
                 |          |          |
Trace memory   ---------------------*>
                 |← Trace area →|
```

Trace or sample data up to the trigger point can be referenced.

3.4 Analyzer

The analyzer checks the execution history of the target program. It is stopped by trigger generation.

The IE-78330-R provides the following four analyzer capabilities:

(1) Real-time trace
(2) Measurement of execution time and the number of instructions
(3) Internal RAM data sampling
(4) CO coverage

3.4.1 Real-time trace (RUN, TRD, TRF, TRG, and TRM commands)

The address bus, data bus, and status signals are traced every bus cycle in real-time execution.

After the analyzer is stopped, trace data are displayed to show the user the flow of program execution. In a midway point of the real-time execution of the emulation CPU, trace data can also be displayed.

Real-time trace has the following features:

. Display of trace data considering operation of the emulation CPU

Instructions which were fetched, but not executed are indicated differently from executed instructions. Data read cycles and data write cycles are indicated with instructions that caused these cycles.

- Trace data search function

  Particular trace data are searched, and displayed with preceding and following five lines or so. In addition, trace data, branch instructions, and checkpoints that satisfy particular conditions are displayed.

- Qualified trace

  Only bus cycles that satisfy the qualified conditions specified with the TRX command are selected and traced.

- Sectional trace

  A section starting at a point when the event conditions specified with the ENB command are satisfied and ending at a point when event conditions specified with the DSB command are satisfied is traced.

- Time tag

  The execution time between the previous write of trace data and the next write of trace data is indicated as the number of system clocks. This can be used to estimate execution time.

The uPD78330, uPD78334, and uPD78P334 have the following two buses to meet the address space to be accessed.

Main bus: Used for normal memory access

Access space: 0H to 0FDFFH

Target memory: Internal ROM
Alternative memory
Write-protected alternative memory
Turbo-access-manager alternative memory
User memory
Internal RAM (0FA00H to 0FDFFH)

Internal bus of the target device:
Used for accessing built-in RAM or an SFR at high speed

Access space: 0FE00H to 0FFFFH

Target memory: Internal RAM (0FE00H to 0FEFFH)
SFRs (0FF00H to 0FFFFH)

Since these two buses operate in parallel, they perform access operation at the same time. To analyze the operation of the uPD78330, uPD78334, or uPD78P334, these two buses need be traced at the same time.

The IE-78330-R can trace the two buses at a time for up to approximately 8000 frames.

External data, however, are traced only when they are selected over time tag (with the TRS command), because of the limited size of trace memory.

Data on external sense clips 1 to 8 are traced as external data.

Time tag is traced by counting execution clocks between trace frames. When the 16-MHz operation clock is used as the clock source, one system clock is 125 ns (8MHz).

Caution: If the internal RAM (0FE00H to 0FEFFH) is read with addressing other than short direct addressing, the trace data value is undefined. However, instructions (target program) are executed normally.

Table 3-5 Trace Data for the Main Bus and the Internal Bus of the Target Device

Trace data	Main bus		Internal bus of target device
Address	16 bits		7 bits
Data	8 bits		16 bits
Status	6 bits		7 bits
External data	8 bits	Selected	
Time tag	8 bits		
Others	7 bits		
Total	75 bits x approximately 8000 frames		

3.4.2 Measurement of execution time and instruction count

The time it took to execute instructions from a point when the event conditions specified with the ENB command were satisfied to a point when the event conditions specified with the DSB command were satisfied is measured, and also the executed instructions are counted.

The resolution is 0.2 us. The possible measurement time ranges from 0.4 us to approximately 14 minutes.

Up to 65535 instructions can be counted.

3.4.3 Internal RAM data sampling (PSA, PSD, PSP, and PST commands)

The contents of internal RAM at up to three addresses specified are sampled at the timing (cycles) specified in words. The sampled data can be displayed after the analyzer is stopped.

The data can also be displayed during real-time execution of the emulation CPU.

The sampling timing may be 0.4, 0.6, 0.8, or 1 to 10000 us (in steps of 1 us).

The memory size for sampled data is 3 words by approximately 2000 frames.

3.4.4 CO coverage (CVD and CVM commands)

The execution of the program is recorded for each address. When a break occurs, the program execution state can be displayed using a CVD command on an address basis.

In addition, the percentage of the executed area to the whole program area is indicated.

The program area is automatically recognized when the program is loaded with the LOD command. If the program is enlarged by patch operations, the area must be added to the measurement range with the add coverage measurement range command (CVM A command).

3.5 Assembly Language Debugging Function

The assembly language debugging function provides the following two capabilities:

(1) Line assembly and disassembly (ASM and DAS commands)

The ASM and DAS commands can change and display memory contents in mnemonics. It facilitates patch of downloaded object code and checking of a program.

In addition to normal mnemonics, pseudo instructions DB, DW, DS, ORG, and END can be used. (See Chapter 10 for details.)

(2) Symbolic debugging function (LOD and SYM commands)

When a symbol table file output by the NEC relocatable assembler is loaded, this function enables the symbols in the symbol table file to be used instead of numerics.

Symbols can be registered with the LOD file S command on a module basis.

Up to approximately 2000 symbols can be registered. Registered symbols can be displayed with the SYM D command on a module basis.

If the same symbol is registered in the symbol table file (LOD file S command) and as an IE symbol twice, the second symbol registration is disabled. If this occurs, the duplicate symbol (with the SYM K command) must be deleted, then it is again registered.

When symbols are registered on a module basis, all symbols of the modules that were registered duplicately are disabled.

3.6 PROM Programmer Control Function (PGM Command)

When a PROM programmer manufactured by NEC (PG-1500 or PG-2000) is connected to channel 2, the IE-78330-R can be used as a terminal of the PROM programmer.

When the IE-78330-R and the PROM programmer are so connected, object code can be uploaded and downloaded between them.

See the description of the usage of the PROM programmer (PG-1500, PG-2000) in Chapter 11 for details.

The communication mode of channel 2 is specified with the channel 2 communication mode setting function (MOD command).

Note that when the IE-78330-R is used with channel 2 connected to an external device other than an NEC PROM programmer, some control characters cannot be used.

In this case, control characters should be changed or the restriction on the use of control characters should be released with the PGM C command.

3.7 Auxiliary Functions

(1) Uploading and downloading of object code
(2) Trigger signal output function
(3) IE symbol manipulation function
(4) Uploading and downloading of debugging environment
(5) Channel 2 communication mode setting function
(6) Command input from file
(7) Command execution result output function
(8) Command file creation function
(9) Command history display function
(10) Command help display function
(11) Directory display function
(12) Math function
(13) Sub-process execution function
(14) Termination of IE-78330-R
(15) Memory word setting function
(16) Other functions

3.7.1 Uploading and downloading of object code (LOD, SAV, and VRY commands)

Object code is loaded and stored in an appropriate memory area according to the memory map.

. Uploading of object code (SAV file C command)

Object code in memory is saved in a specified file in hexadecimal form.

. Downloading of object code (LOD file C command)

Object code in hexadecimal form output by the NEC relocatable assembler package or created with a save command (SAV file C command) is loaded in memory.

Object code is uploaded or downloaded into a particular area specified by the mapping function.

Remark: Verification can be specified by adding $V at the end of a command line.

3.7.2 Trigger signal external output function (OUT command)

The OUT command causes a trigger signal for synchronization with an external tester to output when a trigger point is detected.

The output trigger signal goes high when an event condition set with a trigger point condition (BRM command) is satisfied, and when a trace delay set in trigger point setting (DLY command) has elapsed, the signal goes low. (See Figure 2-13.)

This trigger signal can be used to trigger a logic analyzer.

Cautions 1. The IE-78330-R has eight external sense clips 1 to 8 to trace input data and detect events.

Normally, the eight external sense clips are set as input lines. External sense clip 1 can also be set as a trigger signal output with the OUT ON command.

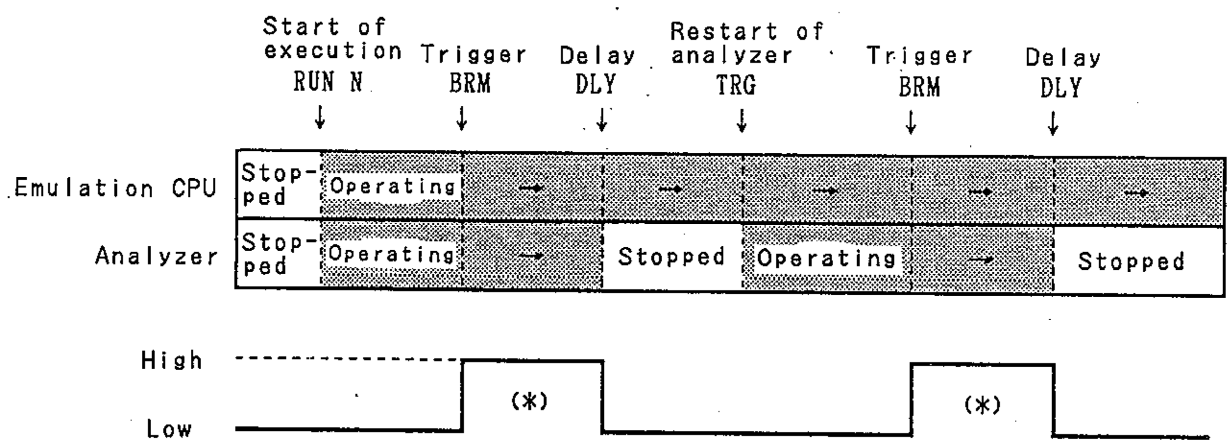
2. When external sense clip 1 is set as a trigger signal output with the OUT ON command, NEVER connect the sense clip to the signal output line of the target system. Otherwise, the target system and IE-78330-R may be damaged.

The following shows the output timings of the trigger signal in different execution modes.

Real-time execution

Fig. 3-11 Trigger Signal Output Timing (1/2)

o Nonbreak real-time execution (RUN N command)

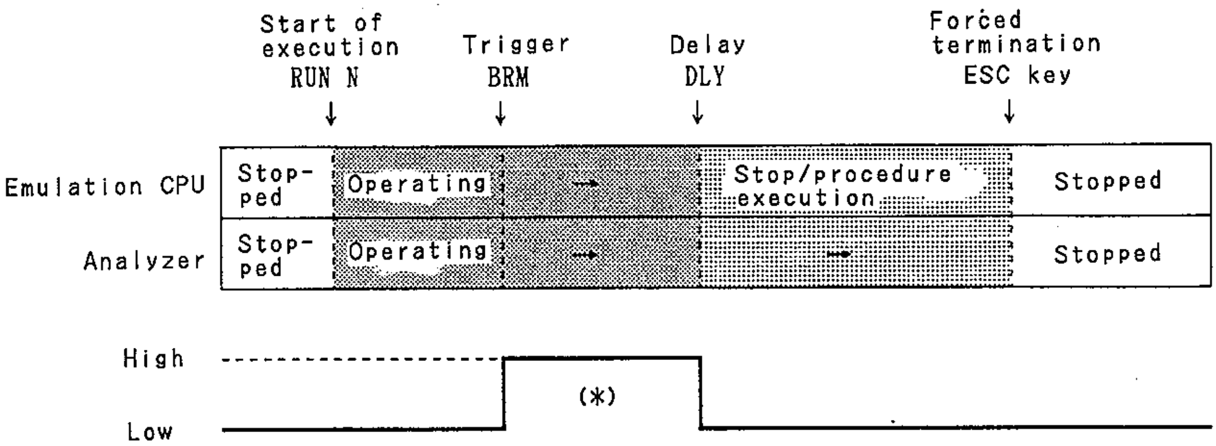


* Even when DLY L is specified in setting a delay condition (to stop the emulation CPU and analyzer as soon as the event detection condition in BRM is satisfied), the trigger signal is held high for at least 1 ms.

Remark: In the above chart, an arrow → indicates that the operation on the left continues.

Fig. 3-11 Trigger Signal Output Timing (2/2)

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



* Even when DLY L is specified in setting a delay condition (to stop the emulation CPU and analyzer as soon as the event detection condition in BRM is satisfied), the trigger signal is held high for at least 1 ms.

Remark: In the above chart, an arrow → indicates that the operation on the left continues.

Nonreal-time execution

o Execution of a step or procedure (RUN T command)

During nonreal-time execution of step operation with the RUN T command, no trigger signal is output.

3.7.3 IE symbol manipulation function (SYM command)

IE symbols entered from the keyboard can be manipulated.

IE symbols are treated as equivalent to symbols stored in the symbol table file of the assembler.

The module of the registered IE symbols is automatically assigned a name IESYMBOL.

The IE symbols can be uploaded or downloaded with the command. Then, the following file name is automatically assigned:

. IE78330.SYM

The IE symbol manipulation function provides the following eight capabilities:

. Registration (SYM A)
. Change (SYM C)
. Display (SYM D)
. Delete (SYM E)
. Delete of all symbols (SYM K)
. Downloading (SYM L)
. Uploading (SYM S)
. Current module specification (SYM M)

3.7.4 Uploading and downloading of debugging environment (LOD and SAV commands)

When a debugging environment is used many times, it can be saved in a file or loaded into the IE-78330-R automatically.

Downloading a debugging environment (LOD file D command) is to load the debugging environment stored in a specified file into the IE-78330-R.

Uploading a debugging environment (SAV file D command) is to save the debugging environment on the IE-78330-R in a specified file.

The following debugging environment information is uploaded or downloaded automatically:

. Internal ROM size
. Internal RAM size
. uPD78330 + TAM emulation
. Download mode
. Commands for debugging environment setup

BRA	BRD	BRS	BRM	CHK	CLK	CVM	DSB	ENB	DLY	MAP
MOD	PGM	PAS	PSA	PST	TRM	TRF	TRS	TRX	WRD	

3.7.5 Channel 2 communication mode setting function (MOD command)

The following communication modes can be set for channel 2:

. Handshaking mode
. Baud rate
. Character length
. Parity bit
. Stop bit

For details on the communication modes, refer to Chapter 8 in "IE-78330-R In-Circuit Emulator: Hardware."

3.7.6 Command input from file (STR command)

Commands or data which used to be entered from the keyboard can be read from a command file and executed automatically.

The command file must be created with the command file creation function (COM command) or the editor.

The file created with the editor can hold formal parameters.

When a file holding formal parameter descriptions is entered, the formal parameters are replaced by actual parameters (up to four).

See Section 8.41 for how to set actual parameters associated with formal parameters.

When ^L is entered during execution of the STR command, the execution is temporarily stopped.

To restart execution, ^L must be entered again.

Entering ^K terminates the STR command.

3.7.7 Command execution result output function (LST command)

Results of execution of the command entered can be output to a desired device.

If a file is specified as the output device (LST file command), the results of execution are stored in the file.

If a list device is specified (LST LST command), the results are output to the printer.

The whole results displayed on the console are output to a file or printer.

Output of execution results is started by entering ^P.

3.7.8 Command file creation function (COM command)

Commands or data entered from the keyboard can be output to a desired device.

If a file is specified as the output device (COM file command), the commands or data are stored in the file.

If a list device is specified (COM LST command), the commands or data are output to the printer.

All the commands or data entered from the keyboard are output to the file or printer.

Output is started by entering ^O.

Remark: The COM command does not allow specification of a formal parameter.

3.7.9 Command history display function (HIS, !n(Note) command)

The most recently entered commands for 20 lines can be stored, and they can be displayed with their command numbers.

When all the stored commands are displayed with the HIS command, a command number is added at the top of each command line.

To display a particular command, its command number must be specified (!n command).
When the specified command is being displayed, entering <cr> causes that command to be executed.

The latest command line can be called by entering !!<cr>.

Note: !n The number of a desired command must be specified in n.

3.7.10 Command help display function (HLP command)

Commands and their use can be displayed.

The following commands are displayed:

ASM	BRA	BRD	BRS	BRM
CHK	CLK	CNT	COM	CVD
CVM	DAS	DIR	DLY	DOS
DSB	ENB	EVN	EXT	HIS
HLP	LOD	LST	MAP	MAT
MEM	MOD	MOV	OUT	PAS
PGM	PSA	PSD	PST	REG
RES	RUN	SAV	SFR	STP
STR	SYM	TRD	TRF	TRG
TRM	TRS	TRX	VRY	WRD

3.7.11 Directory display function (DIR command)

File names can be displayed. This function is equivalent to the DIR/W command of the OS used.

When a directory name is to be displayed, it must be enclosed with < >.

3.7.12 Math function (MAT command)

This function evaluates an expression representation coded in an operand, and displays the results in hexadecimal, decimal, octal, or binary notation.

3.7.13 Sub-process execution function (DOS command)

This function passes control to the OS temporarily.

This function allows OS internal commands or external commands to be executed without the IE-78330-R terminated.

To return control to the control program, enter EXIT<cr>.

3.7.14 IE-78330-R termination function (EXT command)

This function terminates the IE-78330-R, and passes control to the OS.

Caution: To terminate the IE-78330-R, be sure to use this function.

3.7.15 Memory word setting function (WRD command)

A memory length in manipulating registers or memory is set.

The memory length can be changed in bytes (WRD B command) or words (WRD W command).

3.7.16 Other functions

The IE-78330-R provides the following three special functions:

- o Manipulating memory or registers when the emulation CPU is operating.
- o Operation of the IE-78330-R in response to the HALT or STOP instruction
- o Operation of the IE-78330-R in response to a latch-up

(1) Manipulating memory or registers when the emulation CPU is operating

When the following memory or register manipulation is performed during real-time execution without breaks (RUN N command), the operation of the emulation CPU is stopped temporarily. (The time of stopped period varies depending on the command executed.)

- . Memory manipulation function (MEM C or MEM D command)
- . Register manipulation function (REG or SFR command)

Caution: After memory or register manipulation, the emulation CPU restarts operation. However, the CPU performs the operation after executing the above command stops the program temporarily (that is, does not enter the real-time execution mode).

Remark: When the emulation CPU is operating, memory or a register can be manipulated only when analyzer is stopped (prompt emu:> appears).

(2) Operation of IE-78330-R in response to the HALT or STOP instruction

When the emulation CPU executes a HALT or STOP instruction in real-time execution (RUN N or B command), the CPU is placed in the standby mode. If the standby mode continues for at least five seconds, the operation state of the IE-78330-R is as follows:

Table 3-6 IE-78330-R Operation State in the Standby Mode

Mode	Message	State of emulation CPU	Command to cancel standby mode forcibly: Trace mode trc:0>	Command to cancel standby mode forcibly: Emulation mode emu:0>
Standby mode	E-CPU(*) STOP mode!	Stopped state	RES STP	MEM REG RES SFR STP
	E-CPU(*) HALT mode!	Halt state		

* E-CPU: Emulation CPU

Caution: When a HALT or STOP instruction is executed in nonreal-time execution (RUN T command), the emulation CPU does not enter the standby mode.

(3) Operation of the IE-78330-R when a latch-up occurs

If the emulation CPU in the IE-78330-R or a peripheral CMOS device causes a latch-up, the IE-78330-R stops power supply to that device, and displays the following message:

E-CPU(Note) latch up! Restart? (Y)

Note: E-CPU: Emulation CPU

If Y<cr> is entered in reply to the message, the IE-78330-R restarts.

If a latch-up still occurs in a second attempt, contact your nearest NEC distributor.

CHAPTER 4 DEBUGGING PROCEDURE AND SCREEN DISPLAY EXAMPLES

This chapter explains the debugging procedure and the commands used during each step.

It also explains the functions specific to the IE-78330-R using the screen images based on examples of debugging sample programs.

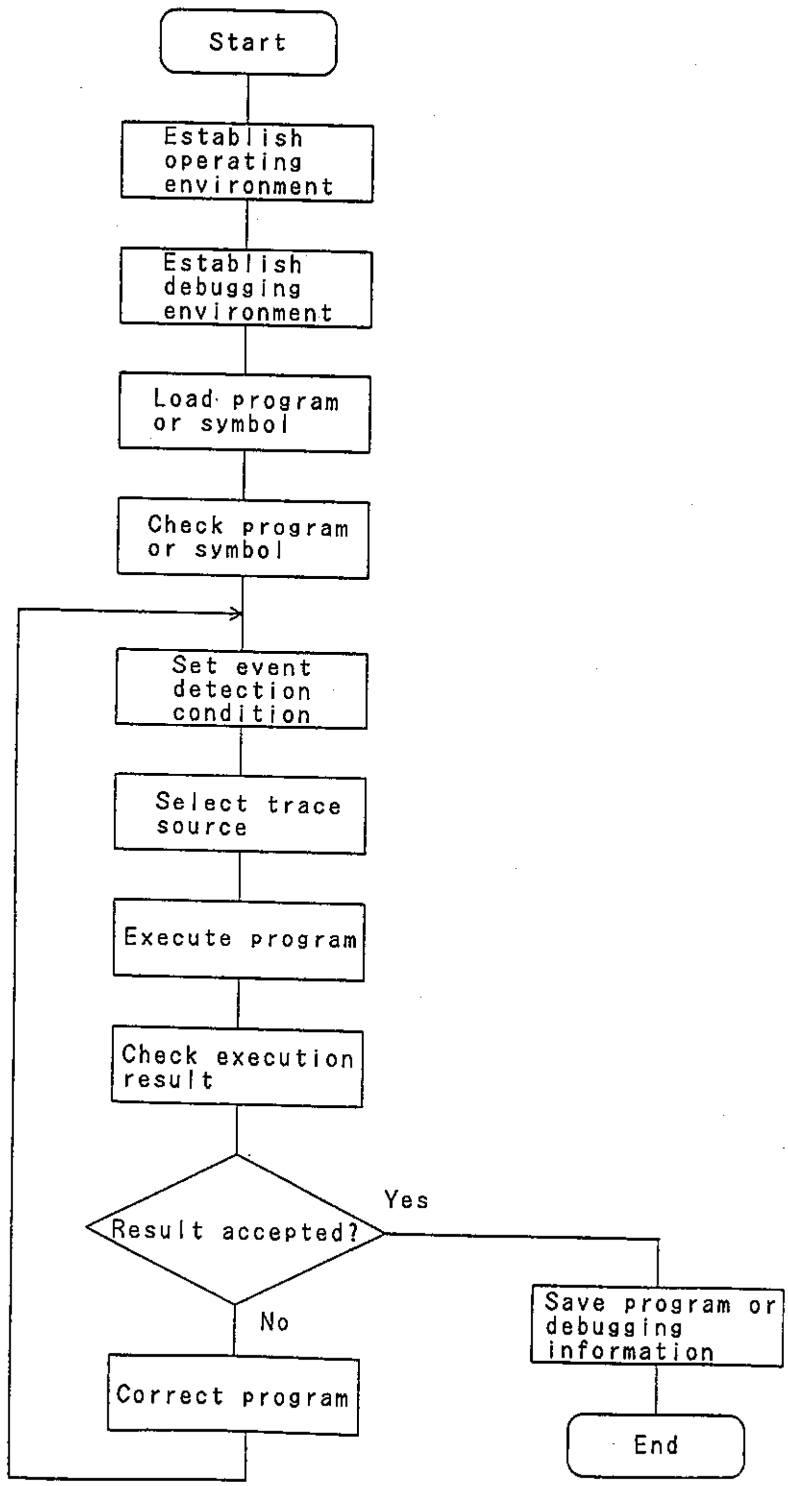
Conventions

- o ___: Indicates that the underlined item needs to be entered from the keyboard.
- o <cr>: Indicates that the return key (CR (0DH)) needs to be pressed.
- o R/O: Read only
- o R/W: Read/write
- o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

4.1 Example of the Debugging Procedure

Figure 4-1 shows an example of the debugging procedure using the IE-78330-R.

Fig. 4-1 Example of Debugging Procedure



The debugging procedure consists of the following nine steps, which are explained below.

- o Establishing the debugging environment
- o Loading a program or symbol
- o Checking the loaded program or symbol
- o Setting an event detection condition
- o Selecting the trace source
- o Executing the loaded program
- o Checking the execution result
- o Correcting the program
- o Saving the program or the debugging information

(1) Establishing the debugging environment

After the operating environment is established, the system waits for a command to be entered.

Establish the debugging environment generally using the following commands before debugging.

. CLK:	Selects a clock.
. LOD file D:	Loads the debugging environment from a file.
. MAP:	Mapping
. OUT:	Selects output of a trigger signal.
. RES:	Resets the emulation device.
. WRD:	Specifies the memory element length.

(2) Loading a program or symbol

Load a program or symbol after establishing the debugging environment.

Use one of the following commands to load a program.

. LOD file C: Loads a program from a file.
. PGM: Loads a program from an NEC PROM programmer.

Use one of the following commands to load a symbol.

. LOD file S: Loads a symbol from a file.
. SYM L: Loads an IE symbol.

(3) Checking the loaded program or symbol

Check the loaded program or symbol using the following commands.

. DAS: Disassembles the program.
. MEM D: Checks data.
. SYM D: Checks the symbol.

(4) Setting an event detection condition

At the initial stage of debugging, the program is debugged when an event is detected at each point.

Use the following commands to set event detection conditions.

. BRA: Sets a bus event detection condition.
. BRD: Sets an external data detection condition.
. BRM: Sets a trigger condition.

- . BRS: Sets a program execution detection condition.
- . DLY: Sets a delay condition.
- . DSB: Sets a disable condition.
- . ENB: Sets an enable condition.
- . PAS: Sets a pass condition.

(5) Selecting the trace source

Select trace information required for debugging using the following commands.

- . PSA: Sets a sample address.
- . PST: Sets the sample timing.
- . TRM: Selects a trace mode.
- . TRS: Selects trace data.
- . TRX: Sets a qualified trace condition.
- . CHK: Sets a checkpoint condition.

(6) Executing the loaded program

Execute the loaded program using one of the following five commands.

. RUN N:	Executes the program in real time without breaks.
. RUN B:	Executes the program in real time with breaks.
. RUN T:	Executes the program step-by-step.
. RUN T (with PRC):	Executes the procedure.
. TRG:	Restarts the analyzer.

(7) Checking the execution result

Check the execution result using the following commands after the emulation CPU and analyzer are stopped when an event is detected.

. CNT: Checks the elapsed execution time and the number of executed instructions.
. CVD: Displays the result of C0 coverage measurement.
. MEM D: Checks data.
. PSD: Displays sample data.
. REG D: Checks the contents of general registers and flags.
. SFR D: Checks the contents of the SFR.
. TRD: Checks trace data.
. TRF: Specifies a trace data retrieval condition.

(8) Correcting the program

If a major correction is required, correct the program at the source level and reload it in the IE-78330-R.

If a minor correction is required, correct the program or data using the following commands.

. ASM: Corrects the program at the mnemonic level.
. MEM C: Changes the contents of memory.

(9) Saving the program or the debugging information

Save the program or the debugging information after debugging.

Use one of the following commands to save the program.

- SAV file C: Saves the program into a file.
- PGM: Saves the program into an NEC PROM programmer.

Use the following command to save symbol information. A symbol loaded using an LOD command, however, cannot be saved.

- SYM S: Saves the IE symbol.

Use the following command to save the debugging environment.

- SAV file D: Saves the debugging environment into a file.

4.2 Screen Display Examples

This section explains the functions specific to the IE-78330-R using the screen images based on examples of debugging sample programs.

These sample programs are written to explain the specific functions and have no special meaning.

4.2.1 Detecting an event related to macro service and interrupt service and displaying trace data

An explanation is given with debugging of a sample program by macro service as an example.

(1) Overview of the operation of the sample program, flowchart, and sample program

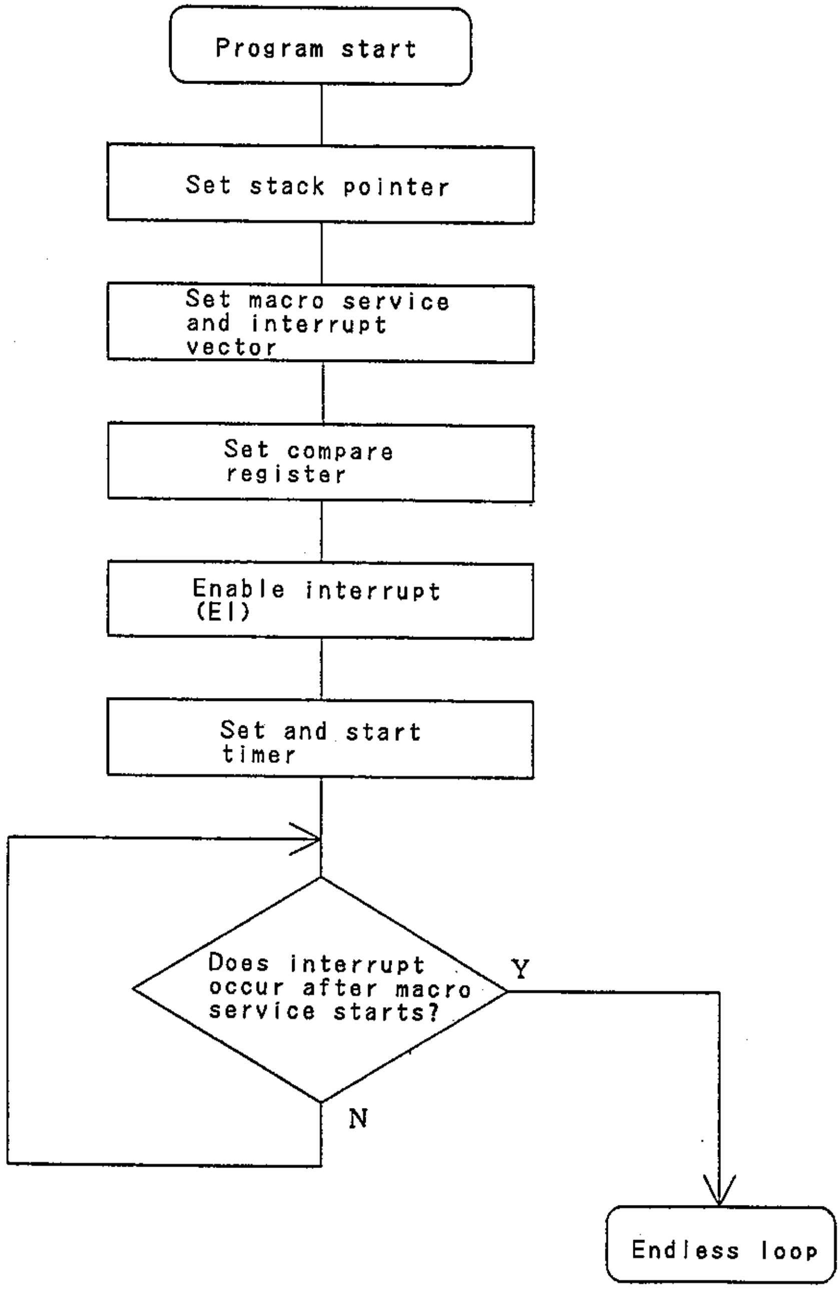
Overview of the operation of the sample program

. Makes timer 0 run freely. When the contents of timer 0 indicate the predetermined compare register value, starts the block transfer macro service. (Saves data read from port 2 into address 0FD00H.)

. Causes an interrupt after the macro service starts.

. Executes an endless loop using an interrupt routine.

Flowchart of the sample program



Sample program

```
        ORG     100H
MAIN:
        MOVW    SP,#0FE00H          ← Sets the stack pointer.
        MOV     R1,#14H          ┐
        MOV     !0FE1EH,A        |
        MOV     R1,#41H          |
        MOV     !0FE1F,A         |
        MOV     0FE40H,#1H       |
        MOV     0FE41H,#2H       |
        MOVW    0FE3E,#0FD00H    |
        MOV     MK0H,#0EFH       |  ← Sets the block transfer macro
        MOV     ISM0H,#10H       |    service and interrupt vector.
        MOVW    RP7,#1000H       |
        MOVW    !1EH,RP7         ┘
        MOVW    CC10,#0H            ← Sets the compare register.
        EI                          ← Enables an interrupt.
        MOV     TMC0,#80H           ← Sets and starts the timer.
        BR      $$
;
        END


        ORG     1000H
INT:
        BR      $INT                ← Endless loop
;
        END
```

(2) Outline of debugging the sample program, flowchart, and screen display examples

Outline of debugging

. Sets block transfer by the macro service as the event detection condition. (Sets data read from port 2 as the event detection condition.)

. Sets a vector reference involved in interrupt processing as the event detection condition.

. Displays trace data after detecting an event.

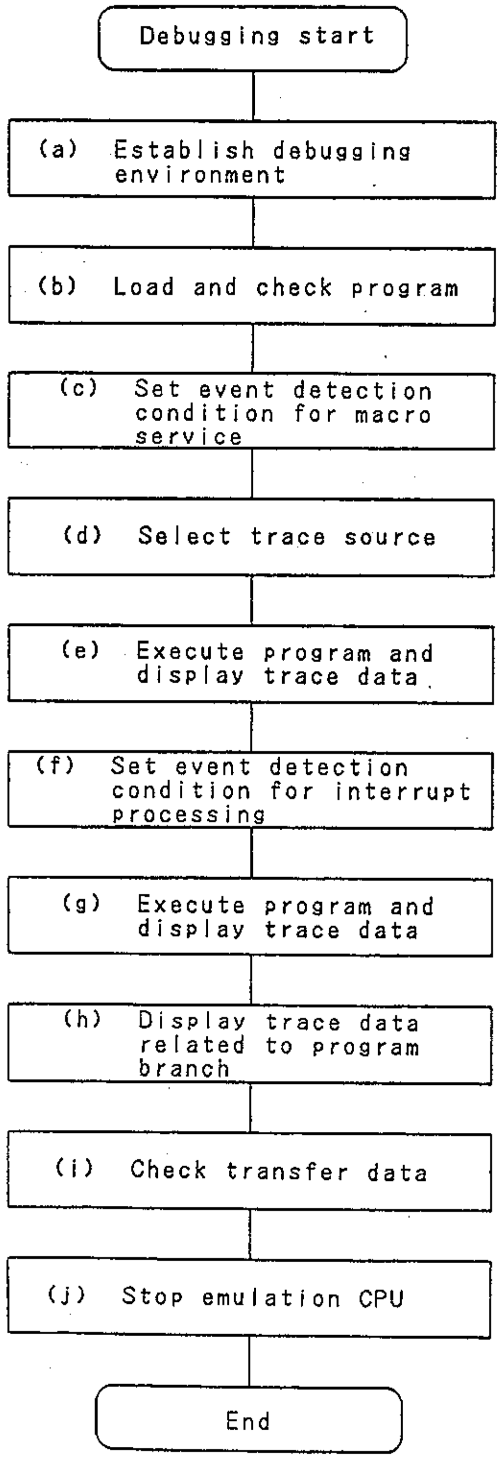
. Retrieves and displays trace data related to the program branch.

. Checks data transferred by the macro service. (Checks the contents of memory in operation in the emulation CPU.)

. Stops the emulation CPU.

Flowchart of debugging

- (a) to (j) correspond to screen display examples on the following pages.



(3) Screen display examples

(a) Establishing the debugging environment

```
brk:0>CLK I <cr>
brk:0>RES <cr>
brk:0>OUT OFF <cr>
brk:0>MAP I 0K <cr>
brk:0>MAP W 0,3FFF <cr>
brk:0>MAP <cr>
 0000-3FFF     R/W emulation
 4000-FC7F     Non map
 FC80-FEFF     <Internal RAM>
brk:0>WRD W <cr>
brk:0>
```

∘ Explanation of the commands used in (a)

. CLK I:	Selects the clock in the emulator.
. RES:	Resets the emulation device.
. OUT OFF:	Disables trigger signal output.
. MAP I 0K:	Sets the size of internal ROM to 0K bytes.
. MAP W 0,3FFF:	Maps alternate memory into addresses 0H to 03FFFH.
. MAP:	Checks the mapping state.
. WRD W:	Sets the memory element length to one word.

(b) Loading and checking the program

```
brk:0>LOD MACRO C S <cr>
 object load complete
 symbol table loading
 PUBLIC     load complete
brk:0>DAS MAIN <cr>
 Addr Object             Mnemonic
                     PUBLIC¥MAIN::
 0100  0B FC 00 FE       MOVW    SP,#0FE00H
                           •
                           •
 011A  2B E5 EF          MOV     MKOL,#0EFH
brk:0>DAS INT <cr>
 Addr Object             Mnemonic
                     PUBLIC¥INT::
 1000  14 FE             BR      $INT
 1002  00                NOP
                           •
                           •
brk:0>
```

∘ Explanation of the commands used in (b)

- LOD MACRO C S: Loads an object or symbol.
- DAS MAIN: Checks the program.
- DAS INT: Checks the program.

(c) Setting an event detection condition for the macro service

```
brk:0>BRA 1 <cr>
 A 0XXXXH
 V 0XXH
 C NC
 E 0XXXXY

 A = P2 <cr>
 V = <cr>
 OP   (OPecode fetch)             RP   (Read by program)
 RW   (Read Write)                WP   (Write by program)
 R    (Read)                      RWM  (Read Write by Macro service)
 W    (Write)                     RM   (Read by Macro service)
 RWP  (Read Write by program)     WM   (Write by Macro service)
                                  NC   (No condition)
 C = RM <cr>
 E = 0FH <cr>
brk:0>BRM BRA1 <cr>
brk:0>DLY M <cr>
```

∘ Explanation of the commands used in (c)

. BRA 1: Sets a bus event detection condition interactively. (BRA 1)
  . Displays the current setting.
  . Specifies macro service read access to P2.

. BRM BRA1: Sets bus event detection 1 as a trigger condition.

. DLY M: Sets the trigger point at the center of trace memory.

(d) Selecting the trace source

```
brk:0>TRM ALL<cr>
brk:0>TRS E<cr>
```

- Explanation of the commands used in (d)

  . TRM ALL: Specifies the trace of execution from the beginning to a break.

  . TRS E: Specifies the trace of external data.

(e) Executing the program and displaying trace data

```
brk:0>RUN B MAIN <cr>
 User-system Vcc-OFF    Emulation start at 100
 <External data trace mode>
trc:0>
 Bus detection break     terminated BRA1
brk:0>TRD  I <cr>
  Frame  Status  Address Data    Label Mnemonic                    EX
  0079    MSRD    FF02    XX                                       00
 T0079    MSRD    FF02    XX                                       00
  0081    MSWR    FD00    00                                       00
  <INTCC10>
  0082    INTRD   001E    00                                       00
  0083    INTRD   001F    10                                       00
  0085    INTWR   FDFE    08                                       00
  0086    INTWR   FDFF    00                                       00
  0088    INTWR   FDFC    2F                                       00
  0089    INTWR   FDFD    01                                       00
                                PUBLIC¥INT::
  0090            1000    14FE        BR      $INT
  Total frame = 4174T     (L/F/T/+/cr/-/Frame No./.)?. <ESC>
brk:0>
```

° Explanation of the commands used in (e)

RUN B MAIN: Starts real-time execution with breaks.
. Displays the current setting of the trace mode.

TRD I: Displays trace data with instructions.
. Displays the trigger frame in the frame mode.
. Displays the symbol enclosed by angle brackets that indicates the interrupt source.

(f) Setting an event detection condition for interrupt processing

```
brk:0>BRA 2 <cr>
 A 0XXXXH
 V 0XXH
 C NC
 E 0XXXXY


 A = 1EH <cr>
 V = 0 <cr>
 OP    (OPecode fetch)              RP    (Read by program)
 RW    (Read Write)                 WP    (Write by program)
 R     (Read)                       RWM   (Read Write by Macro service)
 W     (Write)                      RM    (Read by Macro service)
 RWP   (Read Write by program)      WM    (Write by Macro service)
                                    NC    (No condition)
 C = R <cr>
 E = <ESC>
brk:0>BRM BRA2 <cr>
brk:0>DLY <cr>
 TRIGGER POINT     F     M     L
                   |     |     |

  TRACE MEMORY   ------- * ------- >
 brk:0>
 brk:0>TRM <cr>
  ALL
 brk:0>
```

∘ Explanation of the commands used in (f)

BRA 2:	Sets a bus event detection condition interactively. (BRA2) . Displays the current setting. . Reads data 00H from address 01EH.
BRM BRA2:	Sets bus event detection 2 as a trigger condition.
DLY:	Checks the set position of the trigger point in trace memory. . The trigger point is set at the center indicated by *.
TRM:	Checks the trace mode. . Sets the trace of execution from the beginning to a break.

(g) Executing the program and displaying trace data

```
brk:0>RES <cr>
brk:0>RUN N MAIN <cr>
 User-system Vcc-ON        Emulation start at 0100
  <External data trace mode>
trc:0>
 Bus detection break     terminated BRA2
emu:0>TRD I <cr>
  Frame  Status  Address Data    Label Mnemonic                 EX
  0079    MSRD    FF02    XX                                    00
  0081    MSWR    FD00    00                                    00
   <INTCC10>
  0082    INTRD   001E    00                                    00
 T0082    INTRD   001E    00                                    00
  0083    INTRD   001F    00                                    00
  0085    INTWR   FDFE    08                                    00
  0086    INTWR   FDFE    00                                    00
  0088    INTWR   FDFC    2F                                    00
  0089    INTWR   FDFD    01                                    00
                                 PUBLIC¥INT::
  0090            1000    14FE          BR      $INT
  Total frame = 4177T   (L/F/T/+/cr/-/Frame No./.)? <ESC>
emu:0>
```

- Explanation of the commands used in (g)

RUN N MAIN: Starts real-time execution without breaks.
. Displays the trace mode setting.

TRD I: Displays trace data with instructions.
. Displays the trigger frame in the frame mode.
. Displays the symbol enclosed by angle brackets that indicates the interrupt source.

(h) Displaying trace data related to a program branch

```
emu:0>TRD I $J <cr>
 Frame Status Address Data           Label  Mnemonic          EX
<INTCC10>
 0082   INTRD  001E   00                                       00
T0082   INTRD  001E   00                                       00
 0083   INTRD  001F   10                                       00
 0085   INTWR  FDFE   08                                       00
 0086   INTWR  FDFF   00                                       00
 0088   INTWR  FDFC   2F                                       00
 0089   INTWR  FDFD   01                                       00
                                      PUBLIC¥INT::
 0090          1000   14FE                  BR      $INT
                                      PUBLIC¥INT::
 0094          1000   14FE                  BR      $INT
                                      PUBLIC¥INT::
 0098          1000   14FE                  BR      $INT
 Total frame = 4177T  (L/F/T/+/cr/-/Frame No./.) ? <ESC>
emu:0>
```

- Explanation of the command used in (h)

TRD I $J: Displays only trace data related to the program branch.

(i) Checking transfer data

```
emu:0>SFR D P2 <cr>
P2        00
emu:0>
```

∘ Explanation of the command used in (i)

SFR D P2: Displays the contents of the SFR.

. Stops the emulation CPU, displays the contents of the SFR, and automatically restarts the device.

(j) Stopping the emulation CPU

```
emu:0>STP <cr>
Escape break    terminated
 PC    SP  PSW: UF   RBS2 RBS1 RBS0  S    Z   RSS  AC   IE   P/V  LT  CY
1007  FDFE       0    0    0    0    0    0    0    0    0    0    1   0
  R0   R1   R2   R3   R4   R5   R6   R7        RP4       RP5       RP6      RP7
  X    A    C    B                             VP        UP        DE       HL
  00   00   00   00   00   00   00   00        0000      0000      0000     0000
brk:0>
```

∘ Explanation of the command used in (j)

STP: Stops the emulation CPU.

. Displays the contents of the registers in the current bank when the emulation CPU stops.

4.2.2 Detecting an event using external data and displaying trace data

This section is explained based on an example of debugging of a sample program using bank change.

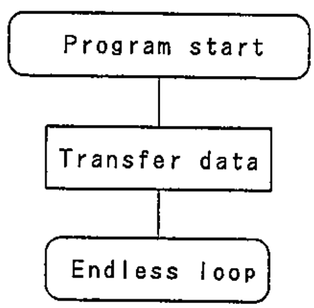
(1) Overview of the operation of the sample program, flowchart, and sample program

Overview of the operation of the sample program

. Writes data in memory starting at address 84000H in the system where the 64KB space of the target device which can be accessed is expanded to 1M bytes using bank change. (It is assumed that Nos. 5 to 8 of the external sense clip are connected at addresses 16 to 19 of the target system for IE setting.)

. Executes an endless loop.

Flowchart of the sample program



Sample program

```
          ORG     100H
EXTACSS:
          MOV     R1,#12      ← Sets transfer data.
          MOV     !4000H,A    ← Transfers data.
          NOP
          NOP
          BR      $$          ← Endless loop
          END
```

(2) Outline of debugging the sample program, flowchart, and screen display examples

Outline of debugging

. Detects an event with the AND of data input from Nos. 5 to 8 of the external sense clip and the bus cycle set as the condition. (Detects access to address 84000 as an event.)

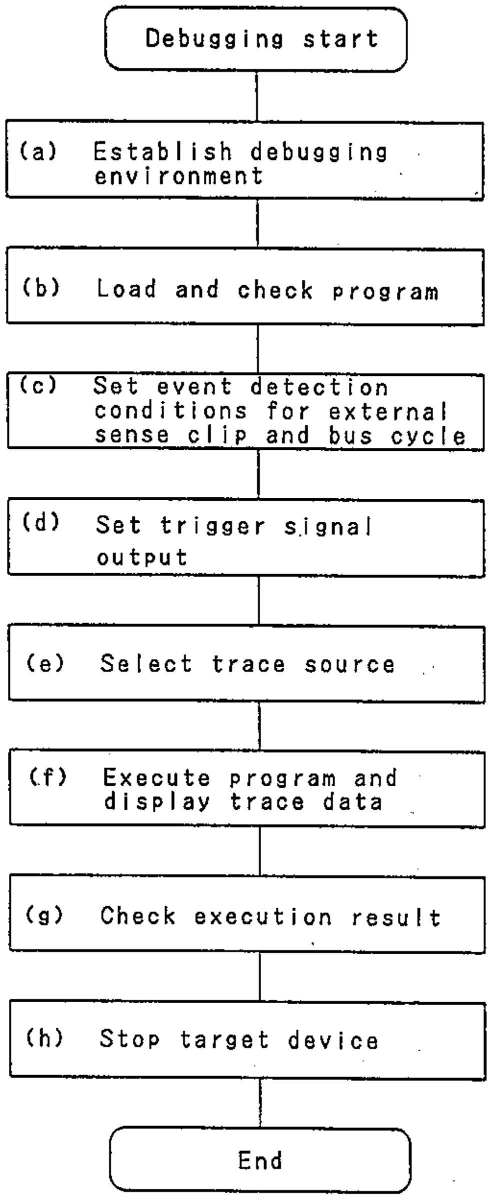
. Outputs the trigger signal from No. 1 of the external sense clip when detecting an event.

. Displays trace data after detecting an event.

. Checks data transferred to memory in the target system. (Checks the contents of memory in operation in the emulation CPU.)

. Stops the emulation CPU.

Flowchart of debugging



(3) Screen display examples

(a) Establishing the debugging environment

```
brk:0>CLK I <cr>
brk:0>RES <cr>
brk:0>OUT OFF <cr>
brk:0>MAP I 16K <cr>
brk:0>MAP U 4000,7FFF <cr>
brk:0>MAP <cr>
 0000 - 3FFF   Internal ROM
 4000 - 7FFF   User
 8000 - FC7F   Non map
 FC80 - FEFF  <Internal RAM>
brk:0>WRD B <cr>
brk:0>
```

- Explanation of the commands used in (a)

. CLK I:	Selects the clock in the emulator.
. RES:	Resets the emulation device.
. OUT OFF:	Disables trigger signal output.
. MAP I 16K:	Sets the size of internal ROM to 16K bytes.
. MAP U 4000,7FFF:	Allocates addresses 04000H to 07FFFH for user memory.
. MAP:	Checks the mapping state.
. WRD B:	Sets the memory element length to one byte.

(b) Loading and checking the program

```
brk:0>LOD EXSIGNAL C S <cr>
 object load complete
 symbol table loading
 PUBLIC     load complete
brk:0>DAS EXTACSS <cr>
 Addr  Object              Mnemonic
                       PUBLIC¥EXTACSS::
 0100  B9 12               MOV      R1,#12H
 0102  09 F1 00 40         MOV      !4000H,A
 0106  00                  NOP
                            •
                            •
brk:0>
```

- Explanation of the commands used in (b)

  . LOD EXSIGNAL C S: Loads an object or symbol.
  . DAS EXTACSS: Checks the program.

(c) Setting event detection conditions for the external sense clip and bus cycle

```
brk:0>BRA 1 <cr>
 A 0XXXXH
 V 0XXH
 C NC
 E 0XXXXY

 A = 4000 <cr>
 V = 12H <cr>
 OP  (OPecode fetch)              RP  (Read by program)
 RW  (Read Write)                 WP  (Write by program)
 R   (Read)                       RWM (Read Write by Macro service)
 W   (Write)                      RM  (Read by Macro service)
 RWP (Read Write by program)      WM  (Write by Macro service)
                                  NC  (No condition)
 C = WP <cr>
 E = 8H <cr>
brk:0>BRM BRA1 <cr>
brk:0>DLY M <cr>
brk:0>
```

◦ Explanation of the commands used in (c)

. BRA 1: Sets a bus event detection condition interactively. (BRA1)
  . Displays the current setting.
  . Set an AND condition by specifying data below:

Address:	04000H
External data:	8H
Transfer data:	12H
Bus status:	WP

. BRM BRA1: Sets bus event detection 1 as a trigger condition.

. DLY M: Sets the trigger point at the center of trace memory.

(d) Setting trigger signal output

```
brk:0>OUT ON <cr>
```

- Explanation of the command used in (d)

  OUT ON: Enables trigger signal output.

(e) Selecting the trace source

```
brk:0>TRM ALL <cr>
brk:0>TRS E <cr>
```

- Explanation of the commands used in (e)

  . TRM ALL: Sets the trace of execution from the beginning to a break.

  . TRS E: Sets the trace of external data.

(f) Executing the program and displaying trace data

```
brk:0>RUN N EXTACSS <cr>
 User-system Vcc-OFF    Emulation start at 0100
  <External data trace mode>
  <Trigger output mode>
  trc:0>
   Bus cycle event    terminated
  emu:0>TRD I <cr>
    Frame Status Address Data             Label  Mnemonic            EX
    0017    WR    4000   12                                          80
   T0017    WR    4000   12                                          80
    0010          0107   00                      NOP                 81
    0012          0108   14FE                    BR        $108H     81
    0018          0108   14FE                    BR        $108H     81
    0022          0108   14FE                    BR        $108H     81
    0026          0108   14FE                    BR        $108H     81
    Total frame = 4112T   (L/F/T/+/cr/-/Frame No./.) ? <ESC>
```

- Explanation of the commands used in (f)

  . RUN N EXTACSS: Starts real-time execution without breaks.
    . Displays the trace mode setting.
    . Displays the specification of trigger signal output.

  . TRD I: Displays trace data with instructions.
    . Displays the trigger frame in the frame mode.
    . Displays output of a high-level trigger signal from No. 1 of the external sense clip at the same time when an event is detected with trace.

(g) Checking the execution result

```
emu:0>MEM D 4000,4000 <cr>
 4000   00
emu:0>
```

- Explanation of the command used in (g)

  MEM D 4000,4000: Displays the contents of memory.
  - Stops the emulation CPU, displays the contents of memory, and automatically restarts the device.

(h) Stopping the emulation CPU

```
emu:0>STP <cr>
 Escape break    terminated
  PC    SP   PSW: UF   RBS2 RBS1 RBS0  S    Z    RSS  AC   IE   P/V  LT  CY
 0108  0000       0     0    0    0    0    0    0    0    0    0    0   0
   R0    R1   R2   R3   R4   R5   R6   R7        RP4       RP5      RP6      RP7
   X     A    C    B                             VP        UP       DE       HL
   00    12   00   00   00   00   00   00        0000      0000     0000     0000
brk:0>
```

- Explanation of the command used in (h)

  STP: Stops the emulation CPU.
  - Displays the contents of the registers in the current bank when the emulation CPU stops.

4.2.3 Detecting an event related to procedure execution and displaying trace data

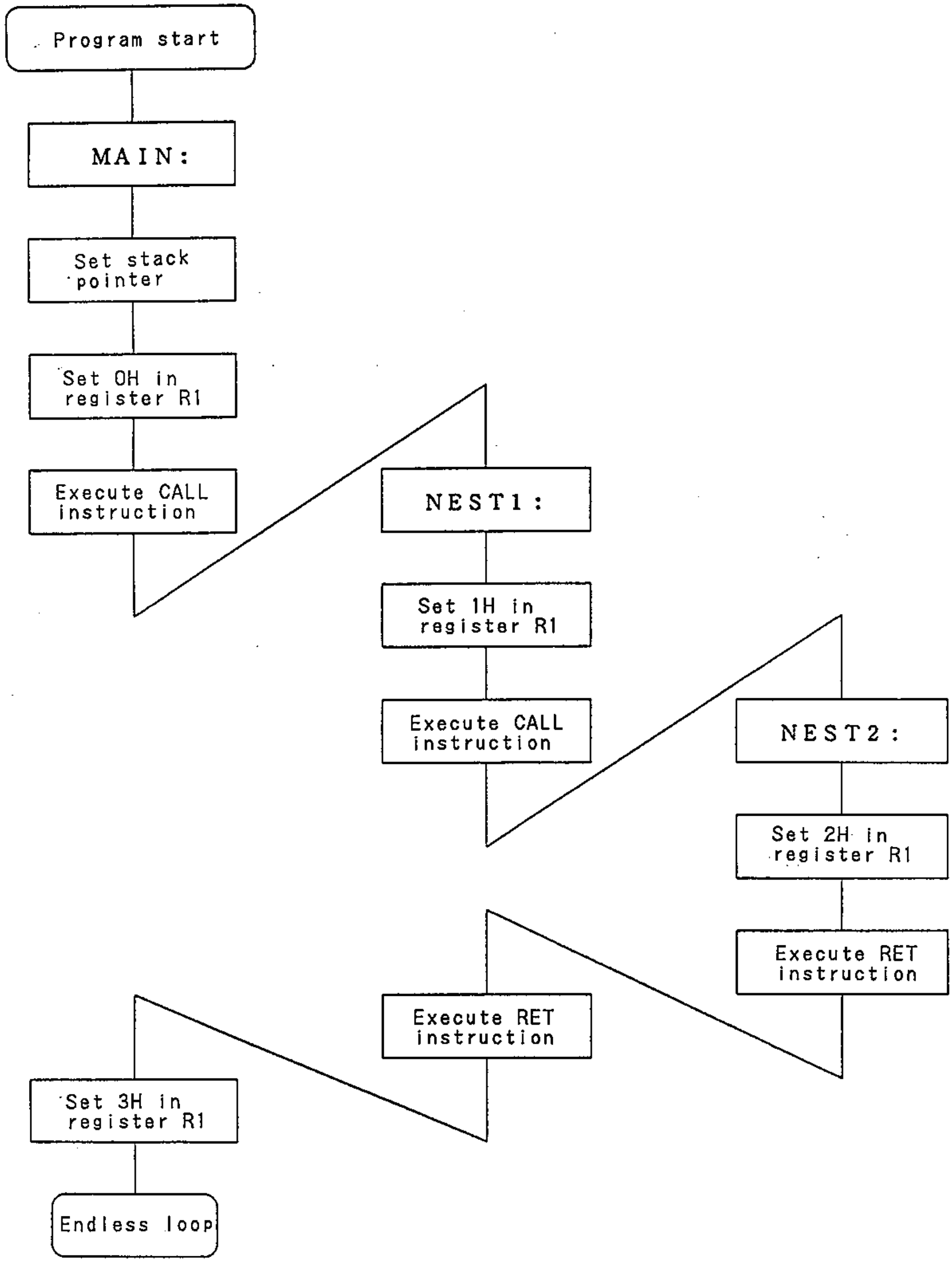
This section explains the procedure execution function based on an example of debugging of a sample program.

(1) Overview of the operation of the sample program, flowchart, and sample program

Overview of the operation of the sample program

- In MAIN:, sets 0H in register R1, executes a CALL instruction, and passes control to the deeper nest level (NEST1:).
- In NEST1:, sets 1H in register R1, executes a CALL instruction, and passes control to the deeper nest level (NEST2:).
- In NEST2:, sets 2H in register R1, executes an RET instruction, and returns control to NEST1:.
- In NEST1: (return destination), executes an RET instruction and returns control to MAIN:.
- In MAIN: (return destination), sets 3H in register R1 and executes an endless loop.

Flowchart of the sample program



Sample program

```
        ORG     100H
MAIN:
        MOVW    SP,#0FE00H      ← Sets the stack pointer.
        MOV     R1,#0H          ← Sets 0H in register R1.
        CALL    !NEST1          ← Passes control to the deeper nest
                                  level.
        MOV     R1,#3H          ← Sets 3H in register R1.
        BR      $$              ← Endless loop
        END


        ORG     1000H
NEST1:
        MOV     R1,#1H          ← Sets 1H in register R1.
        CALL    !NEST2          ← Passes control to the deeper nest
                                  level.
        NOP
        RET                     ← Returns control to MAIN:.


        ORG     2000H
NEST2:
        MOV     R1,#2H          ← Sets 2H in register R1.
        NOP
        RET                     ← Returns control to NEST1:.
        END
```

(2) Outline of debugging the sample program, flowchart, and screen display examples

Outline of debugging

. Specifies register R1 = 3H for an event condition and executes a procedure from MAIN:.

. Checks the value of the PC and the contents of register R1.

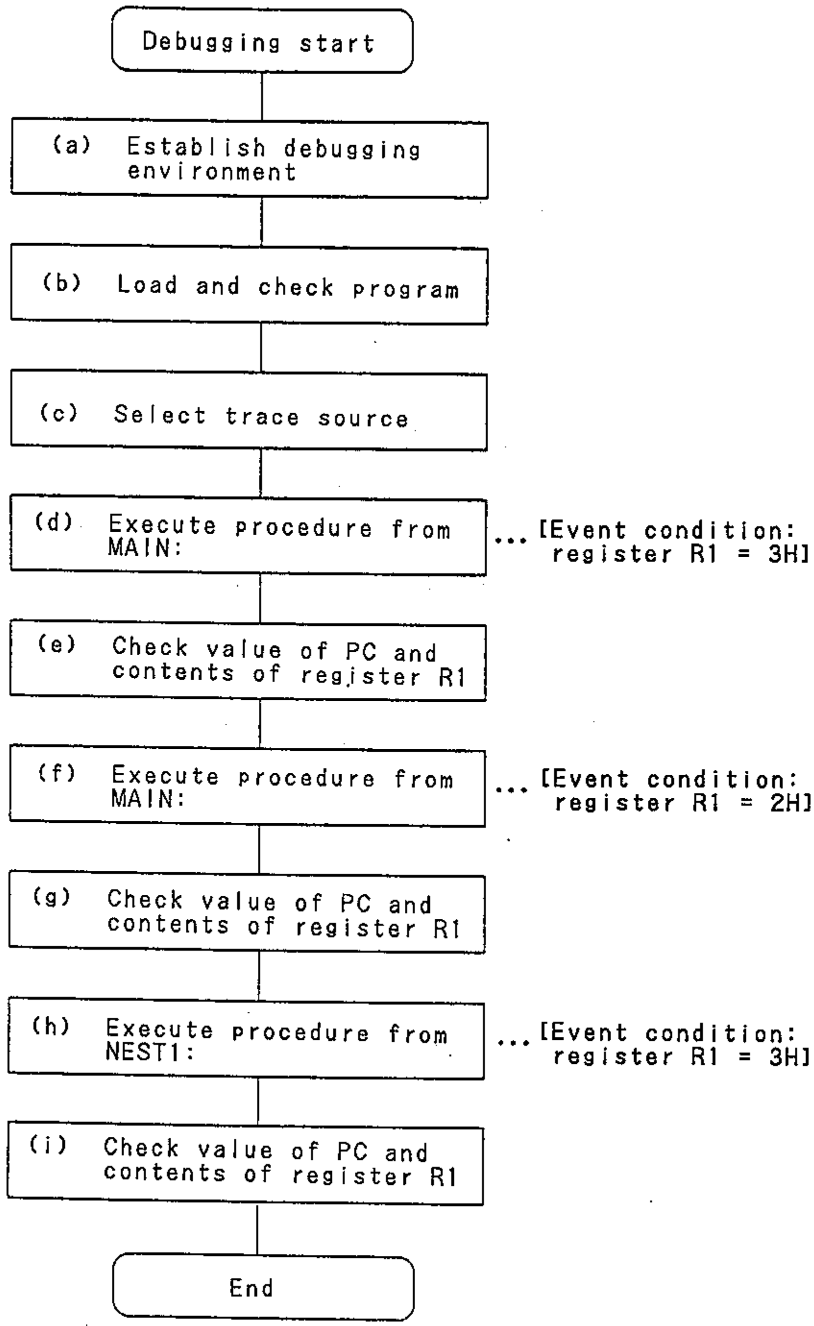
. Specifies register R1 = 2H for an event condition and executes a procedure from MAIN:.

. Checks the value of the PC and the contents of register R1.

. Specifies register R1 = 3H for an event condition and executes a procedure from NEST1:.

. Checks the value of the PC and the contents of register R1.

Flowchart of debugging



(3) Screen display examples

(a) Establishing the debugging environment

```
brk:0>CLK I <cr>
brk:0>RES <cr>
brk:0>OUT OFF <cr>
brk:0>MAP I 16K <cr>
brk:0>MAP <cr>
 0000 - 3FFF   Internal ROM
 4000 - FC7F   Non map
 FC80 - FEFF  <Internal RAM>
brk:0>WRD B <cr>
brk:0>
```

- Explanation of the commands used in (a)

. CLK I:	Selects the clock in the emulator.
. RES:	Resets the emulation device.
. OUT OFF:	Disables trigger signal output.
. MAP I 16K:	Sets the size of internal ROM to 16K bytes.
. MAP:	Checks the mapping state.
. WRD B:	Sets the memory element length to one byte.

(b) Loading and checking the program

```
brk:0>LOD PROCED C S <cr>
 object load complete
 symbol table loading
 PUBLIC     load complete
brk:0>DAS MAIN <cr>
 Addr  Object              Mnemonic
                       PUBLIC¥MAIN::
 0100  0B FC 00 FE         MOVW     SP,#0FE00H
 0104  B9 00               MOV      R1,#0H
                             •
                             •
brk:0>DAS NEST1 <cr>
 Addr  Object              Mnemonic
                       PUBLIC¥NEST1::
 1000  B9 01               MOV      R1,#1H
 1002  28 00 20            CALL     !NEST2
                             •
                             •
brk:0>DAS NEST2 <cr>
 Addr  Object              Mnemonic
                       PUBLIC¥NEST2::
 2000  B9 02               MOV      R1,#2H
 2002  00                  NOP
                             •
                             •
brk:0>
```

o Explanation of the commands used in (b)

. LOD PROCED C S: Loads an object or symbol.
. DAS MAIN: Checks the program.
. DAS NEST 1: Checks the program.
. DAS NEST 2: Checks the program.

(c) Selecting the trace source

```
brk:0>TRS E<cr>
```

- Explanation of the command used in (c)

  TRS E: Sets the trace of external data.

(d) Executing the procedure from MAIN:

```
brk:0>RUN T MAIN,R1=03H TRD PRC <cr>
 User-system Vcc-OFF    Emulation start at 0100
  <External-Sence-Cable trace mode>
 Frame Status Address Data            Label  Mnemonic                EX
                                      PUBLIC¥MAIN::
 0000          0100   0BFC00FE               MOVW      SP,#0FE00H
 0004    WR    FDFE   0104                                           00
 0006          0104   B900                   MOV       R1,#0H
 0010          0106   280010                 CALL      !NEST1
 0015    WR    FDFE   09                                             00
 0016    WR    FDFF   01                                             00
 0019          0109                          MOV       R1,#3H
 terminated
```

- Explanation of the command used in (d)

  RUN T MAIN,R1=3H TRD PRC: Starts executing the procedure.

  . Specifies register R1 = 3H for the event condition.
  . Performs event condition detection and trace only for the current routine.
  . Terminates the execution when detecting the event condition, and displays the contents of the registers in the current bank.

(e) Checking the value of the PC and the contents of register R1

```
 PC     SP   PSW: UF    RBS2 RBS1 RBS0  S    Z   RSS  AC    IE    P/V  LT  CY
010B   FE00        0     0    0    0    0    0    0    0     0     0    0   0
 R0     R1    R2   R3    R4   R5   R6   R7       RP4      RP5      RP6     RP7
  X      A     C    B                            VP       UP       DE      HL
 00     03    00   00    00   00   00   00      0000     0000     0000    0000
One step emulation standby
brk:0>
```

◦ Explanation of the command used in (e)

. PC = 010BH, register R1 = 3H

(f) Executing the procedure from MAIN:

```
brk:0>RUN T MAIN,R1=02H TRD PRC <cr>
 User-system Vcc-OFF    Emulation start at 0100
  <External data trace mode>
  Frame Status Address Data          Label  Mnemonic                   EX
                                    PUBLIC¥MAIN::
 0000           0100    0BFC00FE              MOVW      SP,#0FE00H
 0004    WR     FFFC    00FE                                           00
 0006           0104    B900                  MOV       R1,#0H
 0010           0106    280010                CALL      !NEST1
 0015    WR     FDFE    09                                             00
 0016    WR     FDFF    01                                             00
 terminated
```

◦ Explanation of the command used in (f)

RUN T MAIN,R1=2H TRD PRC: Starts executing the procedure.

- Specifies register R1 = 2H for the event condition.
- Performs event condition detection and trace only for the current routine.

. Detects the event condition in the return value of register R1 at return from the CALL instruction.

. Terminates the execution when detecting the event condition, and displays the contents of the registers in the current bank.

(g) Checking the value of the PC and the contents of register R1

```
 PC    SP   PSW: UF    RBS2 RBS1 RBS0  S    Z   RSS  AC    IE    P/V  LT   CY
0109  FE00        0     0    0    0    0    0    0    0     0     0    0    0
 R0    R1   R2   R3    R4   R5   R6   R7       RP4       RP5       RP6      RP7
  X     A    C    B                             VP        UP        DE       HL
 00    02   00   00    00   00   00   00       0000      0000      0000     0000
One step emulation standby
brk:0>
```

。 Explanation of the command used in (g)

. PC = 0109H, register R1 = 2H

(h) Executing the procedure from NEST1:

```
brk:0>RUN T NEST1,R1=03H TRD PRC <cr>
 User-system Vcc-OFF    Emulation start at 1000
 <External data trace mode>
 Frame Status Address Data             Label  Mnemonic                EX
                                       PUBLIC¥NEST1::
 0000          1000    B901                   MOV      R1,#1H
 0004          1002    280020                 CALL     !NEST2H
 0009   WR     FDFE    05                                              00
 0010   WR     FDFF    10                                              00
 0013          1005    00                     NOP
 0016          1006    56                     RET
 0018   RD     FE00    0030                                            00
 0022          3000                           MOV      R1,#3H
 terminated
```

- Explanation of the command used in (h)

RUN T NEST1,R1=3H TRD PRC:

Starts executing the procedure.

. Specifies register R1 = 3H for the event condition.

. Performs event condition detection and trace only for the current routine.

. Control can be passed from the current routine to the routine at a low nesting level. Performs event condition detection and trace.

. Displays the contents of the registers in the current bank.

(i) Checking the value of the PC and the contents of register R1

```
 PC    SP  PSW: UF  RBS2 RBS1 RBS0  S    Z   RSS  AC   IE   P/V  LT   CY
1006  FE00       0   0    0    0    0    0    0    0    0    0    0    0
 R0    R1   R2   R3  R4   R5   R6   R7        RP4       RP5       RP6       RP7
  X     A    C    B                           VP        UP        DE        HL
 00    03   00   00  00   00   00   00       0000      0000      0000      0000
One step emulation standby
brk:0>
```

- Explanation of the command used in (i)

. PC = 1006H, register R1 = 03H

4.2.4 C0 coverage measurement using the coverage measurement function and displaying the result of measurement

This section explains the C0 coverage measurement function based on an example of debugging of a sample program.

(1) Overview of the operation of the sample program, flowchart, and sample program

Overview of the operation of the sample program

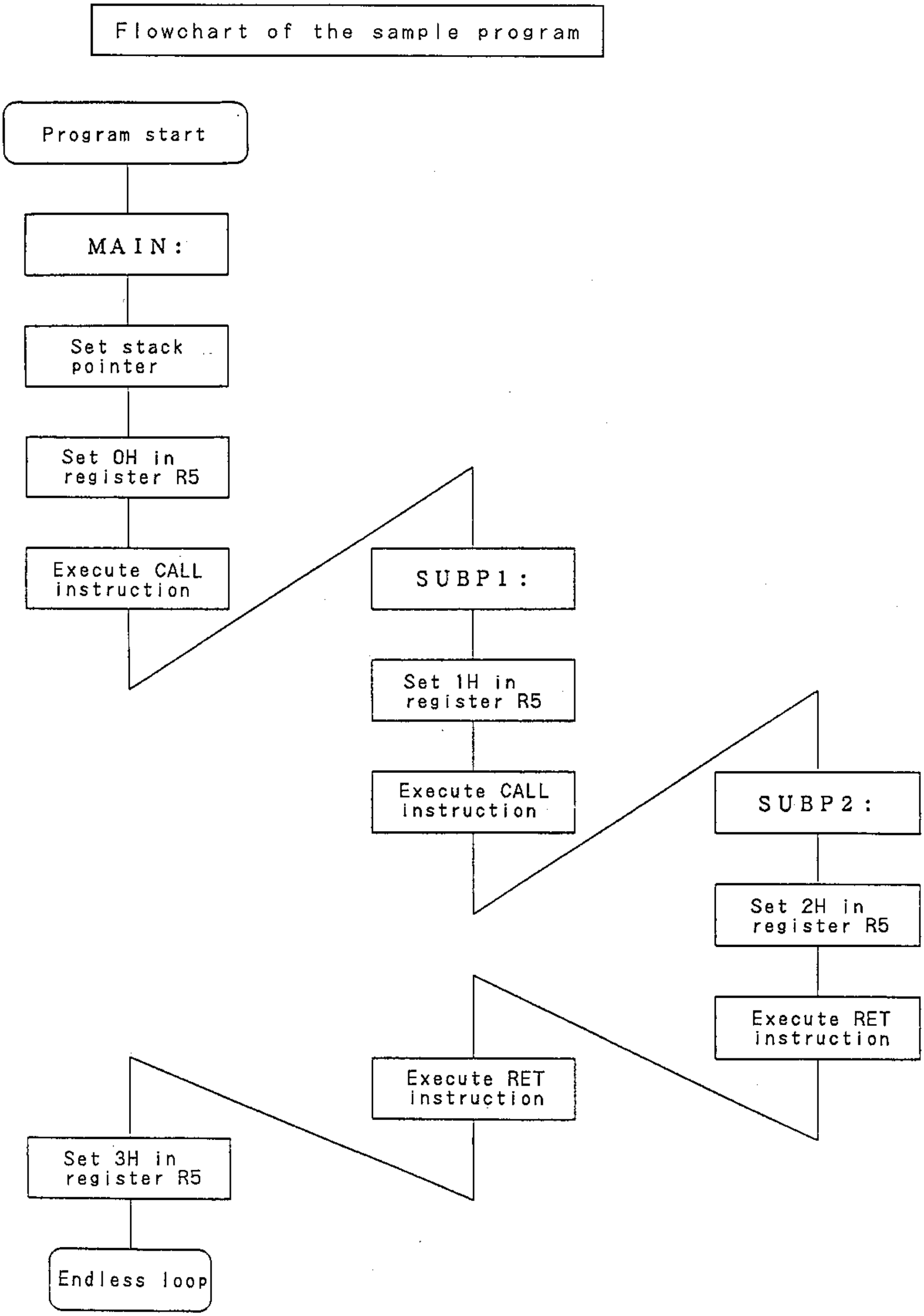
- In MAIN: starting at address 00300H, sets 0H in register R5, executes a CALL instruction, and passes control to SUBP1:.

- In SUBP1: starting at address 00380H, sets 1H in register R5, executes a CALL instruction, and passes control to SUBP2:.

- In SUBP2: starting at address 00480H, sets 2H in register R5, executes an RET instruction, and returns control to SUBP1:.

- In SUBP1: (return destination), executes an RET instruction and returns control to MAIN:.

- In MAIN: (return destination), sets 3H in register R5 and executes an endless loop.


Flowchart of the sample program
Program start
MAIN:
Set stack pointer
Set 0H in register R5
Execute CALL instruction
SUBP1:
Set 1H in register R5
Execute CALL instruction
SUBP2:
Set 2H in register R5
Execute RET instruction
Execute RET instruction
Set 3H in register R5
Endless loop

Sample program

```
        ORG     300H
MAIN:
        MOVW    SP,#0FE00H      ← Sets the stack pointer.
        MOV     R5,#0H          ← Sets 0H in register R5.
        CALL    !SUBP1          ← Passes control to the deeper nest level.
        NOP
        MOV     R5,#3H          ← Sets 3H in register R5.
        BR      $$              ← Endless loop
        END


        ORG     380H
SUBP1:
        MOV     R5,#1H          ← Sets 1H in register R5.
        CALL    !SUBP2          ← Passes control to SUBP2:.
        NOP
        RET                     ← Returns control to MAIN:.


        ORG     400H
SUBP2:
        MOV     R5,#2H          ← Sets 2H in register R5.
        NOP
        RET                     ← Returns control to SUBP1:.
        END
```

(2) Outline of debugging the sample program, flowchart, and screen display examples

Outline of debugging

. Initializes the coverage measurement range and the result of measurement.

. Maps the coverage measurement range by loading the program.

. Checks the mapped coverage measurement range.

. Checks the result of coverage measurement before execution of the program.

. Checks the result of coverage measurement after execution of the program.

Flowchart of debugging

Debugging start

(a) Establish debugging environment

(b) Initialize coverage measurement range and result of measurement

(c) Load and check program

(d) Check mapped coverage measurement range

(e) Check result of coverage measurement

(f) Execute program from MAIN:

(g) Check result of coverage measurement

End

(3) Screen display examples

(a) Establishing the debugging environment

```
brk:0>CLK I <cr>
brk:0>RES <cr>
brk:0>OUT OFF <cr>
brk:0>MAP I 16K <cr>
brk:0>MAP <cr>
 0000 - 3FFF   Internal ROM
 4000 - 7FFF   User
 8000 - FC7F   Non map
 FC80 - FEFF  <Internal RAM>
brk:0>WRD W <cr>
brk:0>BRS 1 A=10C <cr>
brk:0>BRM BRS1 <cr>
brk:0>
```

- Explanation of the commands used in (a)

  - CLK I: Selects the clock in the emulator.
  - RES: Resets the emulation device.
  - OUT OFF: Disables trigger signal output.
  - MAP I 16K: Sets the size of internal ROM to 16K bytes.
  - MAP: Checks the mapping state.
  - WRD W: Sets the memory element length to one word.
  - BRS 1 A=10C: Sets program execution detection condition 1 (Event point: Address 0010CH).
  - BRM BRS1: Sets program execution detection 1 as the trigger condition.

(b) Initializing the coverage measurement range and the result

```
brk:0>CVM K <cr>
brk:0>
```

- Explanation of the commands used in (b)

  CVM K: Deselects the coverage measurement range.

(c) Loading and checking the program

```
brk:0>LOD CVDPROG C S <cr>
 object load complete
 symbol table loading
 PUBLIC     load complete
brk:0>DAS MAIN <cr>
 Addr  object              Mnemonic
                        PUBLIC¥MAIN::
 0300  0B FC 00 FE         MOVW     SP,#0FE00H
 0304  BD 00               MOV      R5,#0H
                            •
                            •
brk:0>DAS SUBP1 <cr>
 Addr  Object              Mnemonic
                        PUBLIC¥SUBP1::
 0380  BD 01               MOV      R5,#1H
 0382  28 00 04            CALL     !SUBP2
                            •
                            •
brk:0>DAS SUBP2 <cr>
 Addr  Object              Mnemonic
                        PUBLIC¥SUBP2:
 0400  BD 02               MOV      R5,#2H
 0402  00                  NOP
                            •
                            •
brk:0>
```

- Explanation of the commands used in (c)

LOD CVDPROG C S:	Loads an object and symbol.
DAS MAIN:	Checks the program.
DAS SUBP1:	Checks the program.
DAS SUBP2:	Checks the program.

(d) Checking the mapped coverage measurement range

```
brk:0>CVM D <cr>
 addr  000 100 200 300  400 500 600 700  800 900 A00 B00  C00 D00 E00 F00
 0000              . . .
 1000
 2000
 3000
 4000
 5000
 6000
 7000
 8000
 9000
 A000
 B000
 C000
 D000
 E000
 F000
brk:0>
```

- Explanation of the command used in (d)

CVM D: Displays the coverage measurement range.

. MAIN: is mapped into addresses 00300H to 0033FH, SUBP1: is mapped into addresses 00380H to 003BFH, and SUBP2: is mapped into addresses 00400H to 0043FH.

(e) Checking the result of coverage measurement

```
brk:0>CVD D 300,4FF <cr>
 addr 00               1F 20               3F 40               5F 60               7F
 0300 .......
 0380 ....
 0400 ..
 0480
 coverage 00.0%
brk:0>
```

- Explanation of the command used in (e)

  CVD D 300,4FF: Displays the result of coverage measurement.
  . The program execution rate is 00.0% because execution of the program does not start.

(f) Executing the program from MAIN:

```
brk:0>RUN N MAIN <cr>
 User-system Vcc-OFF   Emulation start at 0100
  <External data trace mode>
 trc:0>
 Execution break   terminated BRS1
emu:0>STP  <cr>
 Escape break   terminated
  PC    SP   PSW: UF   RBS2 RBS1 RBSO  S    Z   RSS  AC   IE   P/V  LT  CY
 010C  FE00        0    0    0    0    0    0    0    0    0    0    0   0
  R0    R1   R2   R3   R4   R5   R6   R7       RP4       RP5      RP6      RP7
   X     A    C    B                           VP        UP       DE       HL
  55    02   FF   FD   4C   03   30   02      08FD      A600     29DB     FEFC
brk:0>
```

o Explanation of the commands used in (f)

RUN N MAIN: Starts real-time execution without breaks.

STP: Stops the emulation CPU.

. Displays the contents of the registers in the current bank when the emulation CPU stops.

(g) Checking the result of coverage measurement

```
brk:0>CVD D 300,4FF <cr>
 addr 00              1F 20                3F 40                5F 60               7F
 0300 *******
 0380 ****
 0400 **
 0480
 coverage  100.0%
brk:0>
```

o Explanation of the command used in (g)

CVD D 300,4FF: Displays the result of coverage measurement.

. Because MAIN:, SUBP1:, and SUBP2: are executed, 100.0% is displayed for the program execution rate.

CHAPTER 5 COMMAND INPUT FUNCTION

This chapter explains how to enter the IE-78330-R commands, their input formats, and control keys.

Conventions

- o ______: Indicates that the underlined item needs to be entered from keyboard.
- o <cr>: Indicates that the return key (CR(ODH)) needs to be pressed.
- o <ESC>: Indicates that the escape key needs to be pressed.
- o ^ : Indicates that the character on the right side of a caret (^) needs to be pressed while the control key is held down.
- o ■: Indicates the cursor.
- o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

5.1 IE-78330-R Command Input Function

This section explains the command input function available with the IE-78330-R connected with a PC-9800 series or IBM PC series machine.

5.1.1 Command input formats

There are two types of command input formats.

(1) Single-line command input

The single-line command input format enables the user to enter command bodies, subcommands, and operands separated by spaces, which are not longer than 128 characters in total.

The user can end command input by entering <cr>.

The user can enter <cr> at any point on a command line.

After command line editing, the user need not move the cursor to the end of the command line.

When <ESC> is entered while a command is being entered, the command is canceled.

(2) Interactive command input

The single-line command input format is the basic command input format with the IE-78330-R. However, the user can interactively enter those commands (including BRA, MOD, and TRF) whose setting is complicated and those commands (including MEM C, PGM C and REG C) with which data are sequentially changed.

The user is to sequentially enter data after entering <cr>.

When <ESC> is entered, the data that have been entered so far are validated, and command input is terminated.

Example 1: Interactive command input

```
brk:0>MOD <cr>              ← Enter command body only
 Mode CHAR = FLOW <cr>
 Baud 9600 = 4800 <cr>
 Long    8 = 7 <cr>         ← Interactive input
 Par   NON = EVEN <cr>
 Stop    2 = 1 <cr>
brk:0>
```

Example 2: Termination of interactive command input

```
brk:0>MEM C 500 <cr>
 0500 32 = 45 <cr>
 0501 33 = <ESC>            ← Terminate command input
brk:0>
```

5.1.2 Command input methods

There are four command input methods available with the IE-78330-R:

- Command input based on key entry
- Command input from a file
- Command abbreviation input
- Recalling a command line

(1) Command input based on key entry

To terminate command input, enter <cr>.

To cancel or terminate command input, enter <ESC>.

- Command input cancellation:
  When <ESC> is entered with the single-line command input format

- Command input termination:
  When <ESC> is entered with the interactive command input format

To enter a command by key entry, the following edit functions can be used within the command line:

(a) One character deletion (DEL key, BS key)

Deletes the one character just before the cursor position.

(b) One-line deletion (^X key)

Deletes the entire command line being entered.

(c) Replacement

Replaces the character at the cursor position with a keyed character.

(d) Insertion (^A key)

Inserts a keyed character string at the cursor position.

Character string insertion is possible when the symbol < is displayed at the cursor position, that is, in the insertion mode. Toggle control based on ^A key input is used for the insert mode.

(2) Command input from a file

Commands can be entered from a file that already contains the commands (data entered with the COM command or editor).

To enter commands from a file, the STR command is used.

After command input from a file is completed, the user can enter subsequent commands only at the keyboard.

Example: Command input from a specified file

```
        . File name        TEST.STR
        . File contents    MAP R 2000,3FFF
                           MAP W 4000,5FFF
                               .
                               .
                               .
                           MEM D 300X

brk:0>STR TEST.STR              <- Specify command input
brk:0>MAP R 2000,3FFF              from TEST.STR file
brk:0>MAP W 4000,5FFF           <- Commands entered
          :                        from TEST.STR file
brk:0>MEM D 300X
brk:0>
```

(3) Command abbreviation input

Each command listed in Table 5-1 can be entered using its abbreviation, which consists of the first character of the command name.

When entering only a command body, enter the first character of the command name followed by <cr>.

When entering an operand, press the space key after entering the first character of the command name; then the system waits for operand input after command body display.

Table 5-1 List of Abbreviations

Command	Abbreviation	Command	Abbreviation	Command	Abbreviation
ASM	A	HLP	H	SAV	S
CLK	C	LOD	L	TRD	T
DAS	D	MEM	M	VRY	V
EXT	E	RUN	R		

Example 1: Input of CLK by using its abbreviation

```
brk:0>C <cr>     ← When first character followed
                   by <cr> is entered, command is
                   executed after command body
        ↓          display.
brk:0>CLK <cr>
 Internal
brk:0>
```

Example 2: Input of CLK U by using its abbreviation

```
brk:0>C <space key> ← When space key is pressed
                      after entering first
                      character, system waits
                      for operand input after
          ↓           command body display.
brk:0>CLK ■(Note)   ← Enter operand in this
                      state
brk:0>CLK U <cr>    ← Executed after operand
                      input
brk:0>
```

Note: Indicates the cursor position.

(4) Recalling a command line

The IE-78330-R memorizes the latest 20 command lines.

To recall a command, enter !command-number then <cr>.

To recall the most recently entered command, enter !!<cr>.

To display all memorized command lines and command numbers, enter HIS <cr>.

Example 1: Displaying all memorized commands and command numbers

```
brk:0>HIS <cr>  ←
        1   MEM F 0XXX 0
        2   MEM D 0,0FFH
        :
        20  HIS  ←          ← Command entered most
brk:0>                        recently
```

Example 2: Calling command line number 2

```
brk:0>!2 <cr>
 MEM D 0,0FFH■
```

Then enter <cr> to execute the command. To modify the command line, use control keys for editing.

Example 3: Calling the command line entered most recently

```
brk:0>!! <cr>
 HIS■
```

5.1.3 Control keys

Table 5-2 lists the control keys usable with the IE-78330-R.

Table 5-2 Control Keys

Control key	Function	Control key	Function
^A	Insert mode toggle switch	^X	Deletes one line.
^C	Terminates control program forcibly.	BS	Deletes one character.
^H	Deletes one character.	CR	Terminates line input.
^I	Same as space key	DEL	Deletes one character.
^J	Terminates line input.	ESC	Terminates command execution.
^K	Terminates STR command forcibly.	LF	Terminates line input.
^L	Terminates STR command temporarily and restarts STR command.	TAB	Same as space key
^M	Terminates line input.	;	Indicates start of comment.
^O	COM command output switch	!	Calls history.
^P	LST command output switch	←	Moves cursor to left.
^Q	Releases temporary termination using ^S.	→	Moves cursor to right.
^S	Temporary termination		

CHAPTER 6 CODING CONVENTIONS FOR NUMERICS, SYMBOLS, AND EXPRESSIONS

This chapter explains the coding conventions for numerics, symbols, and expressions used to enter commands.

6.1 Coding Conventions for Numerics

There are two rules for coding numerics.

(1) Bit length

Specify a numeric with one of the following bit lengths that is defined in the command being used:

- 16-bit
- 8-bit
- 4-bit
- 1-bit

Code a numeric character (0 to 9) in the high-order digit of a numeric.

When a sign (+ or -) is to be prefixed to a numeric, code a numeric character (0 to 9) after the sign.

(2) Radix

A radix can be specified by adding one of the following symbols at the end of a numeric:

- H: Hexadecimal number
- T: Decimal number
- Q: Octal number
- Y: Binary number

Usually, a radix is defined for each command according to the function. If no symbol for representing a radix is specified, the numeric is processed according to the defined radix. When a numeric is to be entered by using a radix other than that defined, add the symbol for representing the radix.

See Section 7.3 for radix setting for each command.

Table 6-1 indicates the correspondence between the sizes of numerics and radixes.

Table 6-1 Correspondence between Sizes of Numerics and Radixes

			Radix	
			Hexadecimal number	Decimal number
Size of numeric	Minimum		0	0
	Maximum	32-bit	0FFFFFFFFH	4294967295T
		16-bit	0FFFFH	65535T
		8-bit	0FFH	255T
		4-bit	0FH	15T
		1-bit	1H	1T

			Radix	
			Octal number	Binary number
Size of numeric	Minimum		0	0
	Maximum	32-bit	37777777777Q	11111111111111111111111111111111Y
		16-bit	177777Q	1111111111111111Y
		8-bit	377Q	11111111Y
		4-bit	17Q	1111Y
		1-bit	1Q	1Y

6.2 Coding Conventions for Special Numerics

A special numeric represents a set of multiple numerics.

To code a special numeric, the character X for representing an arbitrary numeric is used.

An X corresponds to 0H to FH for hexadecimal, 0Q to 7Q for octal, and 0Y and 1Y for binary, respectively.

Caution: Decimal X representation is not allowed.

There are two rules for coding special numerics.

(1) Coding a numeric range

When coding a numeric range by using Xs, code Xs starting from the low-order digit to the high-order digit successively.

When coding Xs successively up to the high-order digit, prefix 0 to the Xs to indicate that the Xs represent numerics.

Example: Coding a numeric range

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

(2) Coding mask data

Mask data can be coded using an X.

When coding an X for the high-order digit, prefix 0 to the X to indicate that the X represents a numeric.

Example: Coding 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
```

6.3 Coding Conventions for Symbols

A symbol is used for a numeric.

(1) Symbolic name (constituent characters, number of valid characters)

The following characters can be used to make up a symbolic name.

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

The first character of a symbolic name must be a character other than 0 to 9. The lowercase letters (a to z) are handled as the uppercase letters (A to Z).

A symbolic name can consist of up to eight characters. If more than eight characters are coded, only the first eight characters are used.

(2) Symbol value (number of valid digits)

A symbol value is treated as 16-bit data.

(3) Module name

A different coding method applies to a module name coded together with the three types of symbols described below.

(a) Public symbol

When coding a public symbol, code only the symbolic name.

The module name (PUBLIC¥(Note)) need not be coded.

Note: ¥ is replaced with \ when an IBM PC series machine is used as the host machine.

(b) Local symbol

When coding a local symbol, prefix a module name to the symbolic name.

When the current module is specified (SYM M command), code only the symbolic name without specifying the module name.

When the public module contains the same symbolic name, the public symbol is selected.

See Section 8.42 for details.

(c) IE symbol

When coding an IE symbol, code only the symbolic name.

The module name (IESYMBOL¥(Note)) need not be coded.

Note: ¥ is replaced with \ when an IBM PC series machine is used as the host machine.

(4) Maximum number of symbols (number of symbols that can be registered)

The two types of symbols listed below can be coded (registered), and the IE-78330-R allows up to about 2,000 symbols to be coded in total.

(a) Symbol table file
(b) IE symbols

For example, if 1,500 symbols have been registered in the symbol table file, up to 500 IE symbols can be registered.

6.4 Coding Conventions for Expressions

This section explains the coding conventions for an expression, which combines a numeric with another numeric, a symbol with another symbol, or a numeric with a symbol by using an operator.

The coding conventions consist of three items.

(1) Operators

The operators described below can be used for an expression.

Up to 32 levels of parentheses can be used.

. ( )	↑High
. *, /	
. +, -	Priority
. AND	
. OR, XOR	↓Low

(2) Representation

When a numeric and symbol are coded together with an operator (AND, OR, or XOR), one or more spaces are required before and after the operator.

Example: SYM XOR 10101100B

When a numeric and symbol are coded together with an operator (( ), *, /, +, -), no space is required before and after the operator.

Example: (SYM2+36H)

(3) Operation

All operations are performed on a 16-bit integer basis.

If the intermediate or final result of an operation is longer than 16 bits, all bits beyond the 16 bits are discarded.

An operation only on numerics and symbols is valid; an operation on reserved words or special numerics causes an error.

CHAPTER 7 OUTLINE OF THE COMMANDS

This chapter explains the structure of the commands used with the IE-78330-R, defines the terms and notation of the commands, and provides a command list and initial value table.

Conventions

- o <cr>: Indicates that the return key (CR(0DH)) needs to be pressed.

- o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

7.1 Command Structure

An IE-78330-R command consists of the three items below. Each item is separated from another item by a space.

(1) Command body

A command body consists of three characters.

(2) Subcommand

A subcommand specifies processing to be performed by the command body.

Some commands have no subcommand.

(3) Operand

The operand depends on the command. For a command that has a subcommand, its operand cannot be entered without specifying the subcommand.

o The command structure is indicated below.

```
brk:0>command-body subcommand operand
   ↑               ↑         ↑
   Prompt(Note)    space     space
```

Note: The prompt is not included in a command.

7.2 Definition of Command Notation and Terms

This section explains the notation and terms used with the commands.

(1) { }

This specifies that one of the enclosed character strings is to be selected.

(2) [ ]

The character strings enclosed in the brackets can be omitted.

(3) 

A dashed box explains subcommands or operands.

(4) word

This represents a 16-bit numeric (numeric representation).

Radix: Hexadecimal number [H]

(5) pass8

This represents an 8-bit pass count.

Radix: Decimal number [T]

(6) partition

This represents a special numeric.

Radix: Hexadecimal number [H]

(7) mask4

This represents a 4-bit numeric or mask data.

Radix: Binary number [Y]

(8) mask8

This represents an 8-bit numeric or mask data.

Radix: Hexadecimal number [H]

(9) mask16

This represents a 16-bit numeric or mask data.

Radix: Hexadecimal number [H]

(10) data-string

This represents a collection of multiple byte data; up to 10 data items can be represented.

Radix: Hexadecimal number [H]

(11) bit

This represents a 1-bit numeric (numeric representation).

Radix: Binary number [Y]

(12) step16

This represents the number of 16-bit execution steps.

Radix: Decimal number [T]

(13) point

This represents a 13-bit sample frame pointer value.

Radix: Decimal number [T]

(14) number

This represents a sample timing from 1 to 10000.

Radix: Decimal number [T]

(15) expression

This represents an expression.

Radix: Hexadecimal number [H]

(16) parameter

This represents a parameter.

(17) register name

This represents a register name. See (19) for special function registers.

The following register names are used:

Register type	Register name
Program counter	PC
Stack pointer	SP
Program status word	PSW
8-bit general register	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15
16-bit general register	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
8-bit implied register	X, A, B, C, D, E, H, L
16-bit implied register	AX, BC, VP, UP, DE, HL

(18) PSW flag name

The following PSW flag names are available:

Flag type	Flag name
User flag	UF
Register bank selection flag 2	RBS2
Register bank selection flag 1	RBS1
Register bank selection flag 0	RBS0
Sign flag	S
Zero flag	Z
Register set selection flag	RSS
Auxiliary carry flag	AC
Interrupt enable flag	IE
Parity/overflow flag	P/V
Interrupt level transient flag	LT
Carry flag	CY

(19) sfr name

This defines an SFR name.

o The readable and writable SFR names are indicated below.

P0	P1	P3	P4	P5	P9	TLA	BRG	RTP
RTPR	PRDC	RTPS	PWMC	PWM0	ADM	PWM1	CM11	CM12
CM20	CM21	CM30	CSIM	SBIC	SIO	ASIM	CMX0	CM01R
CM02R	CM03R	CM04R	CC00R	CC01R	TMC0	BRGM	TMC1	TUM0
TUM1	TOC0	TOC1	PPOS	STBC	CCW	WDM	MM	PWC
FCC	IF0	IF1	MK0	MK1	PB0	PB1	ISM0	ISM1
CSE0	CSE1	INTM0	INTM1	PRSL				

EXTSFR0	EXTSFR1	EXTSFR2	EXTSFR3	EXTSFR4	EXTSFR5	EXTSFR6	EXTSFR7
EXTSFR8	EXTSFR9	EXTSFR10	EXTSFR11	EXTSFR12	EXTSFR13	EXTSFR14	EXTSFR15

o The read-only SFR names are indicated below.

P2	P7	P8	TM2	CT00	CT01	CT02	CT10	TM0
TM1	TM3	ASIS	RXB	ADCR0	ADCR1	ADCR2	ADCR3	ADCR4
ADCR5	ADCR6	ADCR7	ISPR					

o The write-only SFR names are indicated below.

PM0	PM1	PM3	PM5	PM9	PMC0	PMC1	PMC3	TXS

(20) symbol

This represents a symbolic name.

(21) file

This represents a file name.

(22) module name¥(Note)

This represents a module name in the symbol table.

Note: ¥ is replaced with \ when an IBM PC series machine is used as the host machine.

(23) command

This represents a command body.

7.3 Command List

(1) Reading the command list

The list of all commands used with the IE-78330-R follows on the subsequent pages.

A command can be entered or cannot be entered, depending on the system configuration or operation status. A command that can be entered is marked with o , and a command that cannot be entered is marked with x.

The commands of IE-78330-R are basically entered in single-line input format.

However, the commands indicated below can be entered interactively.

o Commands whose setting is complicated

  . BRA, MOD, and TRF commands

o Commands with which data are changed successively (allowing interactive input only)

  . ASM, MEM C, PGM C, REG C, and SFR C commands

Caution: The command list is provided based on the single-line command input format.

A list of the initial values set with the IE-78330-R commands is provided after the command list.

Command type		Command body	Sub-command	Operand	Default	Operation status trc:	emu:	brk:
Line assemble		ASM	None	[word] word Assemble start address	0H	x	x	o
Condition setting for detection of each event	Condition setting for bus event detection	BRA	1 2 3 4	[A=mask16] [V=mask8] [C=status(*)] [E=mask4] mask16 16-bit detection address range (up to 2 points) mask8 8-bit detection data mask4 4-bit external data status Detection status	A=0XXXXH V=0XXH C=NC E=0XXXXY	x	o	o
	Condition setting for external data detection	BRD	None	[mask4] mask4 Signal levels of external sense clip No. 1 to 4	0XXXXY	x	o	o
	Condition setting for program execution	BRS	1 2 3 4	[A=word] word Detection address	A=0H	x	o	o

(to be continued)

* Select one of the following items for status:

OP
RW
R
W
RWP
RP
WP
RWM
RM
WM
NC

(Cont'd)

Command type	Command body	Subcommand	Operand	Default	Operation status		
					trc:	emu:	brk:
Trigger condition setting	BRM	None	[{[BRA1] [BRA2] [BRA3] [BRA4] [BRD] [BRS1] [BRS2] [BRS3] [BRS4] / OFF}] BR? Each trigger condition OFF Releases each trigger condition.	BRA1	x	o	o
Check-point condition setting	CHK	None	[{[BRA1] [BRA2] [BRA3] [BRA4] [BRD] [BRS1] [BRS2] [BRS3] [BRS4] [{REG / sfr / partition}] / OFF}] BR? Each check condition OFF Releases setting. REG Specifies register. sfr Specifies sfr names (up to 5 names). partition Valid internal RAM range	OFF	x	o	o
Clock selection	CLK	None	[{I / U}] I Clock within emulator U User-set clock	I	x	x	o
Display of elapsed execution time and number of executed instructions	CNT	None	None	None	x	o	o
Command file creation	COM	None	[{file / LST / CON:}] file File name LST: Printer CON: Console	CON:	o	o	o

(to be continued)

(Cont'd)

Command type		Command body	Subcommand	Operand		Default	Operation status		
							trc:	emu:	brk:
Display of CO coverage measurement results	Measurement result display	CVD	D(*)	[partition]	partition Display range	None	x	x	o
	Measurement result initialization	CVD	K	[partition]	partition Initialization range	None	x	x	o
Manipulation of CO coverage measurement range	Measurement range addition	CVM	A	[partition]	partition Coverage range	None	x	x	o
	Measurement range display	CVM	D(*)	[partition]	partition Display range	None	x	x	o
	Release of measurement range specification	CVM	K	[partition]	partition Initialization range	None	x	x	o
Disassemble		DAS	None	[{word / partition}]	word Disassemble start address partition Disassemble address range	0H	x	x	o

(to be continued)

* This subcommand is assumed to be specified and is executed when only the command body is entered.

(Cont'd)

Command type	Command body	Subcommand	Operand	Default	Operation status		
					trc:	emu:	brk:
Directory display	DIR	None	[file] file File name	Current directory of current drive	o	o	o
Setting of trigger point location	DLY	None	[F / M / L] F Sets trigger point location at start of trace memory. M Sets trigger point location in middle of trace memory. L Sets trigger point location at end of trace memory.	L	x	o	o
Subprocess execution	DOS	None	None Executable only with MOS-DOS or PC DOS based machine. User can return to control program by entering EXIT<cr>.	None	o	o	o
Disable condition setting	DSB	None	[[BRA1] [BRA2] [BRA3] [BRA4] [BRD] [BRS1] [BRS2] [BRS3] [BRS4] / OFF] BR? Each disable condition OFF Releases each disable condition.	OFF	x	o	o
Enable condition setting	ENB	1 / 2 / 3	[[BRA1] [BRA2] [BRA3] [BRA4] [BRD] [BRS1] [BRS2] [BRS3] [BRS4] / ON] BR? Each enable condition ON Releases each enable condition.	ON	x	o	o
Display of event detector setting status	EVN	None	None	None	o	o	o

(to be continued)

(Cont'd)

Command type	Command body	Sub-command	Operand	Default	Operation status		
					trc:	emu:	brk:
Control program termination	EXT	None	None	None	x	x	o
Command history display	HIS	None	None	None	o	o	o
Command help display	HLP	None	[command] command: Command body	None	o	o	o
Loading of object, symbol, and debugging environment	LOD	None	file[module name¥(*)][D][C][S][$V] file: File name module name¥: Name of module D: Specifies debugging environment. C: Specifies object. S: Specifies symbol. $V: Specifies verification.	None	x	x	o
Result output to file	LST	None	[-{file / LST: / CON:}-] file: File name LST: Printer CON: Console	CON:	o	o	o

(to be continued)

* When an IBM PC series machine is used as the host machine, \ is used in place of ¥.

(Cont'd)

<table>
<tr><th rowspan="2" colspan="2">Command type</th><th rowspan="2">Com-mand body</th><th rowspan="2">Sub-com-mand</th><th rowspan="2">Operand</th><th rowspan="2">Default</th><th colspan="3">Operation status</th></tr>
<tr><th>trc:</th><th>emu:</th><th>brk:</th></tr>
<tr><td rowspan="2" colspan="2">Mapping</td><td rowspan="2">MAP</td><td>I</td><td>[partition / 0K / 8K / 16K / 24K / 32K / 40K / 48K / 56K] (*1)

I ROM internal to emulation
partition Mapping range</td><td>None</td><td>x</td><td>x</td><td>o</td></tr>
<tr><td>T / W / R / U / K</td><td>[partition] (*1)

T Emulation turbo access manager
W Emulation RAM
R Emulation ROM
U User memory
K Releases mapping (non-mapping)
partition Mapping range</td><td>None</td><td>x</td><td>x</td><td>o</td></tr>
<tr><td colspan="2">Mathematical operation</td><td>MAT</td><td>None</td><td>expression

expression Expression</td><td>None</td><td>o</td><td>o</td><td>o</td></tr>
<tr><td rowspan="3">Memory manipulation</td><td>Memory contents change</td><td>MEM</td><td>C</td><td>[word]

word Change start address</td><td>0H</td><td>x</td><td>o</td><td>o</td></tr>
<tr><td>Memory contents display</td><td>MEM</td><td>D (*2)</td><td>[word / partition]

word Display start address
partition Display address range</td><td>0H</td><td>x</td><td>o</td><td>o</td></tr>
<tr><td>Memory check</td><td>MEM</td><td>E</td><td>[partition]

partition Check address range</td><td>None</td><td>x</td><td>x</td><td>o</td></tr>
</table>

(to be continued)

*1 Settable in units of 8K.

*2 This subcommand is assumed to be specified and is executed when only the command body is entered.

(Cont'd)

Command type		Command body	Sub-command	Operand	Default	Operation status trc:	emu:	brk:
Memory manipulation	Memory contents initialization	MEM	F	partition data-string partition: Initialization address range data-string: Initialization data string	None	x	x	o
	Memory contents retrieval	MEM	G	partition data-string partition: Retrieval address range data-string: Retrieval data string	None	x	x	o
	Memory contents copy	MEM	M	partition word word: Copy destination start address partition: Copy source address range	None	x	x	o
	Memory contents comparison	MEM	V	partition word word: Comparison destination start address partition: Comparison source address range	None	x	x	o
	Memory contents exchange	MEM	X	partition word word: Exchange destination start address partition: Exchange source address range	None	x	x	o
Channel 2 mode setting		MOD	None	[MODE= {CHAR / FLOW}] [BAUD= {19200 / 9600 / 4800 / 2400 / 1200 / 600 / 300}] [LONG= {7 / 8}] [PAR= {NON / EVEN / ODD}] [STOP= {1 / 2}] MODE: Handshaking mode; BAUD: Baud rate; LONG: Character length; PAR: Parity bit; STOP: Stop bit	MODE= CHAR BAUD= 9600 LONG=8 PAR=NON STOP=2	x	o	o

(to be continued)

(Cont'd)

Command type	Com-mand body	Sub-com-mand	Operand	Default	Operation status trc:	emu:	brk:
Data transfer between alternative memory and user memory	MOV	[U / I]	partition word[$V] U: From alternative memory to user memory I: From user memory to alternative memory word: Transfer destination start address partition: Transfer source address range $V: Specifies verification.	None	x	x	o
Trigger signal external output specification	OUT	None	[OFF / ON] OFF: Sets external sense clip No.1 as trace signal input. ON: Sets external sense clip No.1 as trigger signal output.	OFF	x	o	o
Pass condition setting	PAS	None	pass8 pass8: Pass count	1T	x	o	o
PROM programmer control, control character change and cancellation	PGM	C	C: Specifies control character change/cancellation.	None	x	x	o
Sampling address setting	PSA	None	[word][word][word] word: Sample address	None	x	o	o

(to be continued)

(Cont'd)

Command type	Com-mand body	Sub-com-mand	Operand	Operand description	Default	Operation status trc:	emu:	brk:
Sample data display	PSD	None	[ALL / F / L / T / point] [$Y / $B / $W]	ALL Displays all sample data. point Number of sample pointer shifts F Sets sample pointer at start of sample memory. L Sets sample pointer at end of sample memory. T Sets sample pointer at event detection point. $Y Specifies bit display. $B Specifies byte display. $W Specifies word display.	None	x	o	o
Sample timing setting	PST	None	[number / .4 / .6 / .8]	number Sample timing .4 0.4 us .6 0.6 us .8 0.8 us	.4	x	o	o
Register manipulation: Register change	REG	C	[PSW flag name / register name]	PSW flag name Name of PSW flag register name Name of register	RO	x	o	o
Register manipulation: Register display	REG	D(*)	[ALL / PSW flag name / register name]	ALL Specifies all registers of all register banks. register name Name of register PSW flag name Name of PSW flag	All regis-ters of current bank	x	o	o
Resetting of IE-78330-R and emulation CPU	RES	None	[H]	H Resets IE-78330-R.	None	o	o	o

(to be continued)

* This subcommand is assumed to be specified and is executed when only the command body is entered.

(Cont'd)

Command type		Command body	Sub-command	Operand	Default	Operation status trc:	emu:	brk:
Execution	Real-time execution without breaks	RUN	N	[word] word Execution start address	0H	x	x	o
	Real-time execution under break conditions	RUN	B	[word] word Execution start address	0H	x	x	o
	Step-by-step execution	RUN	T	[word][,{(*1) | step16}][REG][TRD]	0H 1T	x	x	o
	Procedure execution	RUN	T	[word][,{(*1) | step16}][REG][TRD]PRC word Execution start address step16 Number of 16-bit steps REG Specifies register display. TRD Specifies trace display. PRC Specifies procedure execution. *1 See below.				

(to be continued)

*1 This is to be replaced with the following table:

```
┌register  ┌ =  ┐  ┌mask8(*2)┐ ┐
│name      │ >  │ ─┤         ├─│
│          │ <  │  └mask16   ┘ │
│          │ => │              │
│          │ >= │              │
│          │ =< │              │
│          │ <= │              │
│          │ >< │              │
│          └ <> ┘              │
│PSW                           │
└flag name   =   bit           ┘
```

mask8	8-bit mask data
mask16	16-bit mask data
bit	1-bit numeric

*2 Only =, ><, or <> can be used with a mask representation.

(Cont'd)

Command type	Com-mand body	Sub-com-mand	Operand	Default	Operation status trc:	emu:	brk:
Saving of object and debugging environment	SAV	None	file[partition(*1)][C][D][$V] file File name C Specifies object. D Specifies debugging environment. partition Save address range $V Specifies verification.	None	x	x	o
SFR manipulation: SFR change	SFR	C	[sfr name] sfr name Name of sfr	PO	x	o	o
SFR manipulation: SFR display	SFR	D(*2)	[sfr name] sfr name Name of sfr	None	x	o	o
Termination of execution	STP	None	[T] T Terminates analyzer only.	None	o	(*3) o	x
Command input from file	STR	None	file[parameter] file File name parameter Actual parameter	None	o	o	o

(to be continued)

*1 Up to five save address ranges can be specified in partition.

*2 This subcommand is assumed to be specified and is executed when only the command body is entered.

*3 T specification is not allowed.

(Cont'd)

Command type		Command body	Subcommand	Operand	Default	Operation status		
						trc:	emu:	brk:
IE symbol manipulation	IESYMBOL registration	SYM	A	symbol word symbol Symbolic name word Symbol value	None	x	o	o
	IESYMBOL change	SYM	C	symbol word symbol Symbolic name word Symbol value	None	x	o	o
	IESYMBOL deletion	SYM	E	symbol symbol Symbolic name	All IE-SYMBOLs registered	x	o	o
	IESYMBOL load	SYM	L	None	None	x	o	o
	IESYMBOL save	SYM	S	None	None	x	o	o
Symbol manipulation	Deletion of all symbols	SYM	K	None	All symbols registered	x	o	o
	Display of all symbols	SYM	D(*1)	[module name¥(*2)] module name¥ Name of module	All symbols registered	x	o	o
	Current module specification	SYM	M	None	None	x	o	o

(to be continued)

*1 This subcommand is assumed to be specified and is executed when only the command body is entered.

*2 When an IBM PC series machine is used as the host machine, \ is used in place of ¥.

(Cont'd)

Command type	Command body	Subcommand	Operand	Default	Operation status trc:	emu:	brk:
Trace data display	TRD	{F / I}	[ALL] [{$F / $J / $Q / $C}] F: Frame display I: Instruction display ALL: Specifies display of all trace data and all data that match retrieval condition. $F: Specifies display of five lines before and after frame that matches retrieval condition. $J: Specifies display of frames related to program branch processing. $Q: Specifies display of frames that match retrieval condition. $C: Specifies display of checkpoint frames.	None	x	o	o
Setting of trace data retrieval condition	TRF	None	[A= {word / partition}] [V=mask8] [C=status(*)] [E=mask8] word: Retrieval address partition: Retrieval address range status: Retrieval status mask8: 8-bit mask data	A=0XXXXH V=0XXH C=NC E=0XXH	x	o	o

(to be continued)

* Select one of the following items for status:

BROP
OP
RWI
RI
WI
RW
R
W
RWP
RP
WP
RWM
RM
WM
NC

(Cont'd)

Command type	Command body	Sub-command	Operand	Default	Operation status trc:	emu:	brk:
Analyzer reactivation	TRG	None	None	None	x	o	x
Trace mode setting	TRM	None	[ALL / TRX / SEC] ALL All traces TRX Qualified trace SEC Sectional trace	ALL	x	o	o
Trace data selection	TRS	None	[E / T] E → EXIT External data T → TIME Time tag	EXT	x	o	o
Condition setting for qualified trace	TRX	None	[[BRA1] [BRA2] [BRA3] [BRA4] / OFF] BR? Qualify condition OFF Qualify release	OFF	x	o	o
Comparison of object file with memory contents	VRY	None	file file File name	None	x	x	o
Memory word length setting	WRD	None	[B / W] B Byte specification W Word specification	B	x	o	o

(2) List of initial command values

The table below lists the initial values of the IE-78330-R commands.

Command name	Sub-command	Initial value
ASM		0H
BRA	1	A=0XXXXH V=0XXH C=NC E=0XXXXY
	2	A=0XXXXH V=0XXH C=NC E=0XXXXY
	3	A=0XXXXH V=0XXH C=NC E=0XXXXY
	4	A=0XXXXH V=0XXH C=NC E=0XXXXY
BRD		0XXXXY
BRS	1	A=0H
	2	A=0H
	3	A=0H
	4	A=0H
BRM		BRA1
CHK		OFF
CLK		I
CNT		None
COM		CON:
CVD	D(*)	None
	K	None

(to be continued)

* This subcommand is assumed to be specified and is executed when only the command body is entered.

(Cont'd)

Command name	Sub-command	Initial value
CVM	A	None
	D(*)	None
	K	None
DAS		0H
DIR		Current directory of current drive
DLY		L
DOS		None
DSB		OFF
ENB	1	ON
	2	ON
	3	ON
EVN		None
EXT		None
HIS		None
HLP		None
LOD		None
LST		CON:

(to be continued)

* This subcommand is assumed to be specified and is executed when only the command body is entered.

(Cont'd)

Command name	Sub-command	Initial value
MAP	I	None
	T	None
	W	None
	R	None
	U	None
	K	None
MAT		None
MEM	C	0H
	D(*)	0H
	E	None
	F	None
	G	None
	M	None
	V	None
	X	None
MOD		MODE=CHAR BAUD=9600 LONG=8 PAR=NON STOP=2
MOV	U	None
	I	None
OUT		OFF
PAS		1T

(to be continued)

* This subcommand is assumed to be specified and is executed when only the command body is entered.

(Cont'd)

Command name	Sub-command	Initial value
PGM	C	None
		Termination of pgm/LOAD/SAVE/PGM ^Z/^B/^F/^Z Beginning of HEX-LOAD/HEX-SAVE /SYM-LOAD ^A/^E/^N Break of LOAD/SAVE ^W
PSA		None
PSD		None
PST		4
REG	C	R0
	D(*)	All registers in the current bank
RES		None
RUN	N	0H
	B	0H
	T	0H, 1T
SAV		None
SFR	C	P0
	D(*)	All readable SFRs
STP		None
STR		None

(to be continued)

* This subcommand is assumed to be specified and is executed when only the command body is entered.

(Cont'd)

Command name	Sub-command	Initial value
SYM	A	None
	C	None
	D(*)	All registered symbols
	E	All registered IESYMBOLs
	K	All registered symbols
	L	None
	S	None
	M	None
TRD	F	None
	I	None
TRF		A=0XXXXH V=0XXH C=NC E=0XXH
TRG		None
TRM		ALL
TRS		EXT
TRX		OFF
VRY		None
WRD		B

* This subcommand is assumed to be specified and is executed when only the command body is entered.

(3) Correspondence between numerics and radixes

This section explains the correspondence between the numerics and radixes used when IE-78330-R commands are entered. As described below, a unique radix is defined for each numeric according to the function of each command.

When the symbol used to represent a radix is not specified, the numeric is processed according to the defined radix. When a numeric is entered using a radix other than those defined, the symbol representing the radix is to be added.

Table 7-1 Correspondence between Numerics and Radixes

Representation	Meaning	Radix
word	16-bit numeric	Hexadecimal: H
mask16	16-bit mask data	Hexadecimal: H
mask8	8-bit mask data	Hexadecimal: H
mask4	4-bit mask data	Binary: Y
pass8	8-bit pass count	Decimal: T
bit	1-bit numeric	Binary: Y
step16	Number of 16-bit execution steps	Decimal: T
point	Number of 11-bit sample frame pointers	Decimal: T
number	Sample timing	Decimal: T
partition	Address range	Hexadecimal: H
expression	Expression	Hexadecimal: H
data-string	Collection of byte data	Hexadecimal: H

CHAPTER 8 EXPLANATION OF COMMANDS

This chapter explains in detail the commands available in the IE-78330-R in alphabetical order.

Conventions

- o ______: Indicates that the underlined item needs to be entered from the keyboard.
- o <cr>: Indicates that the return key (CR (0DH)) needs to be pressed.
- o <ESC>: Indicates that the escape key needs to be pressed.
- o ^: Indicates that the character on the right side of a caret (^) needs to be pressed while the control key is held down.
- o ■: Indicates the cursor.
- o R/O: Read only
- o R/W: Read/write
- o W/O: Write only
- o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.
- o In the explanation of each command in this chapter, the availability of the command for the individual prompts is indicated as follows:

Example:

ASM [word]	
Radix	word:H

trc:0>	×	emu:0>	×	brk:0>	○

Caution: The ASM command is available only when prompt brk:0 appears.

8.1 Line Assembly (ASM)

ASM [word]	
Radix	word:H

trc:0>	×	emu:0>	×	brk:0>	○

word: Start address for assembly

The ASM command enables the user to modify memory contents in mnemonics starting at the address specified in word.

When ASM <cr> is specified, the address next to the previously assembled locations is assumed as the start address.

See Chapter 10 for details on the assembler specifications.

Example: Changing memory contents starting at address 200H

```
brk:0>ASM 200H <cr>
 0200                  NOP                ← Displays mnemonic before change.
              = MOV [HL] ,A <cr>
        55
 0201                     NOP
              = MOVW AX,0FF23 <cr>
        Warning!(304)(*) Generate code? (Y/N) N <cr>  ← Causes warning because
              = MOVW AX,0FF23 <CR>                       Saddrp is performed for
        Warning!(304)(*) Generate code? (Y/N) Y <cr>     odd-numbered address.
        1C 23                             ← Generates code, but operation is
 0203                    NOP                unpredictable.
              = ORG 100H <cr>             ← Goes back to location before current
        Caution!(303)(*)                    location indicated by current location
 0100                    NOP                counter, so displays Caution.
              = MOV R1,FFH <cr>           ← Displays Error because of invalid
        Error!(302)(*)                      mnemonic.
              = MOV R1,#0FFH <cr>
        B9 FF
 0102                  NOP
              = BR 500H <cr>              ← Generic object code is generated.
        Caution!(303)(*)                    Displays Caution.
        2C 00 05
 0105                  NOP
              = END <cr>
brk:0>■
```

* The numbers in parentheses indicate the error message Nos.

8.2 Setting of Bus Event Detection Conditions (BRA)

BRA	{1, 2, 3, 4} [A=mask16][V=mask8][C=status][E=mask4]
Radix	mask16:H mask4:Y mask8:H

trc:0>	×	emu:0>	○	brk:0>	○

BRA 1: Specifies event detector 1.
BRA 2: Specifies event detector 2.
BRA 3: Specifies event detector 3.
BRA 4: Specifies event detector 4.
mask16: 16-bit detection address (up to two)
mask8: 8-bit detection data
status: Detection status
mask4: 4-bit external data

The BRA command sets bus event detection conditions.

There are four event detection conditions: address, data, status, and external data. They are ANDed when detected.

Four event detectors are provided for detecting a bus event. One of the four different detection conditions can be selected by specifying a number in the subcommand.

When BRA n <cr>(Note) is entered, the detection conditions set for a specified bus event detector are displayed, and an interactive setting mode is entered.

Note: n is a number 1 to 4.

Up to two addresses can be set as address conditions, separated from each other by a space. When two addresses are specified, they are ORed. If no address condition is specified, addresses are regarded insignificant.

Data condition is specified with 8-bit mask data. Because of the bus structure, low-order 8 bits of data are valid for addresses 0H to FDFFH. If the data condition is omitted, data are regarded insignificant.

The status condition is selected out of the following bus cycle attributes. If the status condition is omitted, the status is regarded insignificant.

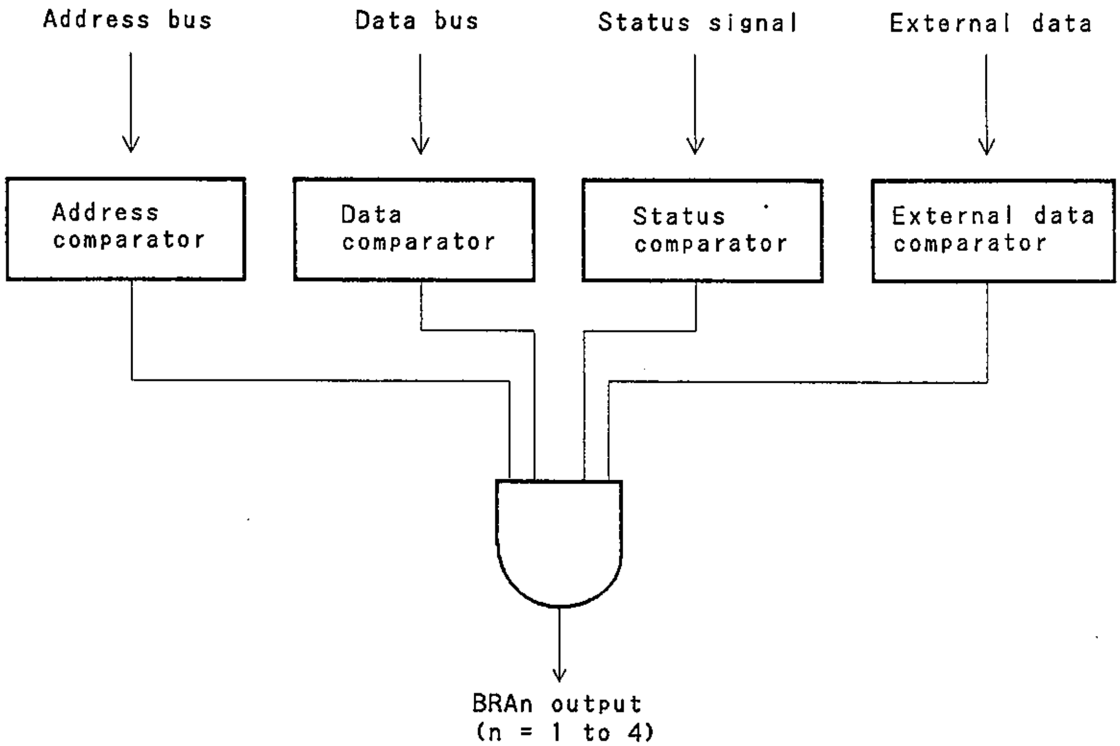
- OP: Op code fetch
- RW: Read/write
- R: Data read
- W: Data write
- RWP: Data read or write by program
- RP: Data read by program
- WP: Data write by program
- RWM: Data read or write by macro service
- RM: Data read by macro service
- WM: Data write by macro service
- NC: All fetch, read, or write operations

The external data condition is set with 4-bit mask data for the external data applied to external sense clips 5 to 8. If the external data condition is omitted, the external data is regarded insignificant.

Figure 8-1 shows the configuration of a bus event detector.

Fig. 8-1 Configuration of a Bus Event Detector

There are four event detectors BRA1 to BRA4, each made up of the following comparators. One of the four detection conditions can be selected by specifying a number in the subcommand.



Example 1: Displaying current conditions

```
brk:0>BRA <cr>

   (BRA1)       (BRA2)       (BRA3)       (BRA4)
 A 0XXXH        0FE40H,      0XXXXH       78XXH
     ---        1F00         ---          ---

 V 0XXH         0XXH         0XXH         0XXH


 C RW           RWP          NC           RW

 E 0101Y        0XXXXY       0XXXXY       1000Y
brk:0>■
```

Example 2: Setting break conditions interactively

```
brk:0>BRA 1 <cr>
 A   0XXXXH
 V   0XXH
 C   NC
 E   0XXXXY

 A = 0XXXH <cr>
 V = <cr>
 OP    (OPecode fetch)              RP    (Read by program)
 RW    (Read Write)                 WP    (Write by program)
 R     (Read)                       RWM   (Read Write by Macro service)
 W     (Write)                      RM    (Read by Macro service)
 RWP   (Read Write by program)      WM    (Write by Macro service)
                                    NC    (No condition)
 C = RW <cr>                              ← Sets C to RW.
 E = 100H <cr>
 Input data error.  (104)(Note)
 E = 100 <cr>                             ← Sets 4-bit mask data to 4H.
brk:0>■
```

Note: The numbers in parentheses indicate the error message Nos.

Example 3: Setting conditions in one line

```
brk:0>BRA 2 A=0FE40 1F00 C=RWP <cr>  ← When target program reads or
brk:0>■                                 writes data from/to memory
                                        location at address 0FE40H or
                                        1F00H, event is generated.
```

8.3 Setting of External Data Detection Conditions (BRD)

BRD [MASK4]	
Radix	MASK4:Y

trc:0>	×	emu:0>	○	brk:0>	○

mask4: Signal level on external sense clips 1 to 4

The BRD command sets the external input signal level to be detected.

When BRD <cr> is entered, the current detection level is displayed.

Example 1: Setting the detection level on external sense clips 1 to 4 to 3H

```
brk:0>BRD 3H <cr>
brk:0>■
```

Example 2: Displaying the set detection level

```
brk:0>BRD <cr>
 0011Y
brk:0>■
```

8.4 Trigger Condition Setting (BRM)

```
         ┌[BRA1][BRA2][BRA3][BRA4][BRD][BRS1][BRS2][BRS3][BRS4] ┐
BRM  [ ─┤                                                         ├─]
         └OFF                                                    ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

BR?(Note): Trigger condition

OFF: Cancels trigger conditions.

Note: BR? indicates BRA1, BRA2, BRA3, BRA4, BRD, BRS1, BRS2, BRS3, and BRS4 trigger conditions.

The BRM command selects one of the outputs of bus event detectors BRA1 to BRA4, a program execution detector, and external data detectors as the trigger signal. More than one trigger condition can also be specified, separated from each other by a space. These trigger conditions are ORed.

When BRM OFF <cr> is entered, no trigger signal is generated.

When BRM <cr> is entered, the current trigger conditions are displayed.

Example 1: Setting trigger conditions

```
brk:0>BRM BRD BRS2 <cr>     ← Sets BRD and BR2.
brk:0>■
```

Example 2: Displaying BRM trigger conditions

```
brk:0>BRM <cr>
 BRD BRS2                   ← Displays current conditions.
brk:0>■
```

8.5 Setting of Program Execution Detection Conditions (BRS)

BRS [1 / 2 / 3 / 4][A=word]	
Radix	word:H

trc:0>	×	emu:0>	○	brk:0>	○

BRS 1: Specifies event detector 1.

BRS 2: Specifies event detector 2.

BRS 3: Specifies event detector 3.

BRS 4: Specifies event detector 4.

word: Detection address

The BRS command sets an address for the program execution detector.

A detection address is set for each event detector. Addresses set in event detectors must be at least five bytes apart from each other.

If two address points are set at an interval of less than five bytes, the second point of them cannot sometimes be detected.

Example 1: Displaying current conditions

```
brk:0>BRS <cr>
    (BRS 1)    (BRS 2)  (BRS 3)  (BRS 4)
  A 0FD00H     0A000H    8000H    100H

brk:0>■
```

Example 2: Setting an address for BRS1

```
brk:0>BRS 1 A=7800    ← Generates event when instruction at address 7800H
brk:0>■                 is executed.
```

8.6 Setting of Checkpoint Conditions (CHK)

```
                  ┌REG          ┐
         ┌BR?  [─┤sfr           ├─] ┐
CHK  [─┤         └partition    ┘    ├─]
         └OFF                       ┘
```

Radix	partition:H

trc:0>	×	emu:0>	○	brk:0>	○

REG:	Specifies a register.
sfr:	Specifies up to five SFR names.
partition:	Specifies an internal RAM range.
BR? (Note):	Check condition
OFF:	Cancels the setting.

Note: Check conditions are BRA1, BRA2, BRA3, BRA4, BRD, BRS1, BRS2, BRS3, and BRS4. Two or more conditions can be specified at a time, separated from each other by a space.

The CHK command stops the emulation CPU temporarily when a specified event condition is detected, traces register, SFR, or internal RAM contents, then restart the emulation CPU.

Trace data for the checkpoint are displayed in the frame next to the checkpoint. Note that if the event condition is BRA, a slippage may occur.

When CHK OFF <cr> is entered, the set checkpoint is canceled.

Example 1: Specifying registers for the checkpoint

```
brk:0>CHK BRA1 BRS2 REG <cr>    ← Traces register contents when BRA1 or
brk:0>■                           BRS2 event occurs.
```

Example 2: Canceling the checkpoint setting.

```
brk:0>CHK OFF <cr>
brk:0>■
```

8.7 Clock Selection (CLK)

```
CLK [ I ]
    [ U ]
```

trc:0>	×	emu:0>	×	brk:0>	○

I: Clock in the emulator
U: User-defined clock

The CLK command selects a clock source of the clock signal to be supplied to the emulation CPU, and resets the emulation CPU. One of the following clock sources is selected:

(1) Clock in the emulator (CLK I)

When a 16-MHz crystal resonator is connected to the emulation CPU, the clock in the emulator must be specified (CLK I command).

(2) User-defined clock (CLK U)

When debugging is performed using a clock with a frequency of other than 16 MHz (16 MHz at maximum), the user-defined clock must be specified (CLK U command).

When the user-defined clock is specified, a crystal resonator having a frequency twice higher than the clock signal used must be connected to the socket (OPCK) on the emulation board in the IE-78330-R.

Example: To operate the emulation CPU at 12 MHz, a 24-MHz resonator must be connected.

Remarks 1. The clock in the emulator is used when the IE-78330-R is started.

2. When CLK <cr> is entered, the name of the current clock source is displayed.

3. Clock signals can only be supplied from the IE-78330-R. (The target system cannot supply clock signals.)

Example 1: Specifying a clock source

```
brk:0>CLK U <cr>    ← Specifies user-defined clock.
brk:0>■

brk:0>CLK I <cr>    ← Specifies clock in emulator (16 MHz).
brk:0>■
```

Example 2: Displaying the clock setting state

```
brk:0>CLK <cr>      ← Displays specified clock source.
 Internal           ← When clock in emulator is specified
brk:0>■

brk:0>CLK <cr>      ← When user-defined clock is specified
 User
brk:0>■
```

8.8 Display of Elapsed Execution Time and Number of Executed Instructions (CNT)

CNT

trc:0>	×	emu:0>	×	brk:0>	○

The CNT command measures and displays an elapsed execution time and the number of instructions executed between the ENB point and the DSB point or between the ENB point and the trigger point.

The results of measurement do not include the delay set with the DLY command.

The elapsed execution time can be measured in the range of 400 ns to 858 seconds (approximately 14 minutes), and instructions ranging from 1 to 65535 can be counted.

Example 1: Displaying elapsed execution time and the number of executed instructions

```
brk:0>CNT <cr>
 Emulation Time = 3min 12sec 500msec 000.0 usec  ← Displays elapsed
                                                   execution time.
 Instruction Stop = 28931T                       ← Displays number of
brk:0>■                                            executed instructions.
```

Example 2: Displaying a message when an overflow occurs

```
brk:0>CNT <cr>
 Emulation Timer overflow.  (200)(Note)          ← Timer overflows.
 Instruction counter overflow.  (201)(Note)      ← Counter overflows.
brk:0>■
```

Note: The numbers in parentheses indicate the error message Nos.

8.9 Creation of Command File (COM)

```
COM [ file  ]
      LST:
      CON:
```

trc:0>	O	emu:0>	O	brk:0>	O

file: File name
LST: Printer
CON: Console

The COM command opens a specified device to hold entered commands.

When COM file <cr> is entered, input commands are saved in the specified file.

When the extension of a file name is omitted, .STR is automatically added to the file name.

When COM LST: <cr> is entered, input commands are output to the printer.

To start command output, enter ^O.

To stop command output, enter ^O again.

After a file or printer is opened by a COM command, be sure to close the device by entering COM CON: <cr> when necessary data have been output to the device.

Example 1: Opening a file named SAMPLE.STR in drive B

```
brk:0>COM B:SAMPLE.STR <cr>
brk:0>■
```

Example 2: When there is a file having the same name as the file to be opened in drive B

```
brk:0>COM B:SAMPLE.STR <cr>
 File already exists.  (513)(Note)  B:SAMPLE.STR Delete? (Y or N):Y <cr>
brk:0>■
```

If there is a file named SAMPLE.STR with read and write attributes in drive B, the above message appears.

When Y <cr> is entered in response to the message, the existing file is deleted, and a new file is created.

Specification other than Y <cr> does not open a file.

Example 3: When a file cannot be opened

```
brk:0>COM B:SAMPLE.STR <cr>  ← Specifies name of file with R/O attribute.
 Read only file.  (507)(Note)  B:SAMPLE.STR
brk:0>■
```

When a file named SAMPLE.STR with the R/O attribute is present in drive B, the above message appears, and the command is ignored.

Example 4: When the printer is opened normally

```
brk:0>COM LST: <cr>
brk:0>■
```

Note: The numbers in parentheses indicate the error message Nos.

Example 5: When the printer cannot be opened normally

```
brk:0>COM LST: <cr>
  Printer used by other command.  (109)(Note)
brk:0>
```

If the printer is not available because it is being used by another command (process), the above message appears, and the command is ignored.

Note: The number in parentheses indicates the error message No.

Example 6: Command output started by entering ^O

```
brk:0>COM B:SAMPLE.STR <cr>    <- Opens file named SAMPLE.STR in drive B.
brk:0>RES <cr>
brk:0>^O CLK U <cr>            <- Enter ^O at beginning of command line.
brk:0>OUT ON <cr>
brk:0>
brk:0>^O                       <- Enter ^O when command input is prompted.

 -> Output to SAMPLE.STR
```

In the above example, the commands from the command (CLK U) following ^O input after the file is opened to the command (OUT ON) immediately before another ^O input are output to the file.

8.10 Display of CO Coverage Measurement Results (CVD)

<table><tr><td colspan="2">CVD [{D|K}[partition]]</td></tr><tr><td>Radix</td><td>partition : H</td></tr></table>

trc:0>	×	emu:0>	×	brk:0>	○

CVD D: Displays measurement results.
CVD K: Initializes measurement results.
partition: Display/initialization range

The CVD command displays or initializes the results of CO coverage measurement.

When CVD <cr> is entered, the results of CO coverage measurement are displayed.

(1) CVD D command

CVD [D[partition]]	
Radix	partition : H

trc:0>	×	emu:0>	×	brk:0>	○

CVD D: Displays measurement results.
partition: Display range

When a display range is specified, the percentage of executed instructions to the instructions in the specified range is indicated, and the range is displayed in 2-byte units.

When CVD D <cr> is entered, the percentage of executed instructions to the instructions in the mapped range is indicated, and the program execution map for the 64K-byte memory space is indicated in 64-byte units.

In a display of the C0 coverage measurement range, an asterisk (*) indicates that the instruction at a corresponding location has been executed. A period (.) indicates that a corresponding location is mapped. Blank parts are unmapped locations. A question mark indicates that a corresponding location is not mapped, but executed by the program.

Example 1: Displaying results of measurement

```
brk:0>CVD D <cr>
 addr   000 100 200 300  400 500 600 700  800 900 A00 B00  C00 D00 E00 F00
 0000  *************.** ****......****** **************** ****************
 1000  **************** **************** *********....... ................
 2000  ???????
 3000
 4000
 5000  ......
 6000  **************** ????
 7000
 8000  ................ ................ ................ ................
 9000  ................ ................ ................ ................
 A000  ................ ................ ................ ................
 B000  ................ ................ ................ ................
 C000  ................ ................ ................ ................
 D000  ................ ................ ................ ................
 E000  ................ ................ ................ ................
 F000  ................ ................ ................ ................
 Coverage 19.3%
brk:0>■
```

Example 2: Displaying results of measurement with range specification

```
brk:0>CVD D 300,4FF<cr>  ← Displays results of measurement for addresses 300H
                           to 4FFH.
 addr 00             1F 20             3F 40             5F 60             7F
 0300 **************** **************** **************** ****************
 0380 **************** **.............. ............**** ****************
 0400 **************** **************** **************** ****************
 0480 **************** **************** **************** ****************
 coverage 89.8%
brk:0>■
```

(2) CVD K command

CVD K [partition]

trc:0>	×	emu:0>	×	brk:0>	○

CVD K: Initializes measurement results.
partition: Initialization range

The CVD K command initializes the results of C0 coverage measurement.

When CVD K <cr> is entered, the entire space is initialized.

Example: Initializing results of C0 coverage measurement

```
brk:0>CVD K 300,4FF <cr>
brk:0>■
```

8.11 Manipulation of CO Coverage Measurement Range (CVM)

```
CVM [┬A   [partition]┬]
     ├[D][partition] ┤
     └K   [partition]┘
```

Radix	partition:H

trc:0>	×	emu:0>	×	brk:0>	○

CVM A: Specifies an additional measurement range.

CVM D: Displays a measurement range.

CVM K: Cancels measurement range specification.

Partition: CO coverage range

The CVM command adds, displays and cancels a CO coverage measurement range.

When CVM A partition <cr> is entered, the specified range is added to the current CO coverage measurement range.

When CVM D <cr> is entered, the entire CO coverage measurement range is displayed.

When CVM D partition <cr> is entered, the specified CO coverage range is displayed.

When CVM K <cr> is entered, the entire CO coverage measurement range is canceled, and the measurement results are initialized.

When CVM K partition <cr> is entered, the specified CO coverage range is canceled, and the measurement results are initialized.

When a coverage measurement range is displayed, parts marked with a period (.) are in the CO coverage measurement range, and blank parts are outside the range.

Loading object data with the LOD command automatically adds the loaded addresses to the coverage measurement range.

Example 1: Specifying a CO coverage measurement range

```
brk:0>CVM A 8000,0EFFF <cr>     ← Specifies measurement range from
brk:0>■                           addresses 8000H to 0EFFFH.
```

Example 2: Displaying a CO coverage measurement range

```
brk:0>CVM D <cr>
 addr  000 100 200 300   400 500 600 700   800 900 A00 B00   C00 D00 E00 F00
 0000  ................  ................  ................  ................
 1000  ................  ................  ................  ................
 2000    ┐
 3000    ├─ Not in the CO coverage measurement range
 4000    ┘
 5000  ....
 6000  ................
 7000
 8000  ................  ................  ................  ................
 9000  ................  ................  ................  ................
 A000  ................  ................  ................  ................
 B000  ................  ................  ................  ................
 C000  ................  ................  ................  ................
 D000  ................  ................  ................  ................
 E000  ................  ................  ................  ................
 F000
brk:0>■
```

Example 3: Canceling the current CO coverage measurement range

```
brk:0>CVM K <cr>
brk:0>■
```

8.12 Disassembly (DAS)

```
DAS [┬word     ┬]
    └partition┘
```

trc:0>	×	emu:0>	×	brk:0>	○

word: Disassemble start address

partition: Disassemble address range

The DAS command displays the contents of memory starting at a specified address in mnemonics.

When only a start address is specified, 11 lines of instructions stored in memory locations starting at the specified address are displayed.

When a start and end address are specified, the memory contents in the specified range are displayed.

When DAS <cr> is entered, 11 lines of instructions starting at the address next to the instructions displayed in the previous disassembly operation are displayed.

See Chapter 10 for the specifications of disassembler.

Example: Displaying disassembled code

```
brk:0>DAS 100 <cr>                      ← Displays 11 lines of memory
 Addr  Object       Mnemonic              contents starting at address 100H.
 0100  B8 00        MOV     R0,#0H
 0102  B9 01        MOV     R1,#1H
         .
         .
 0111  00           NOP
brk:0>■
```

8.13 Display of Directory (DIR)

DIR [file]

trc:0>	O	emu:0>	O	brk:0>	O

file: File name

The DIR command displays the names of directories and files.

When the file name is omitted, the names of all files in the current directory in the current drive are displayed.

Directory names are displayed enclosed in < >.

When a drive name is specified, the names of files on that drive are displayed. File names can also be displayed by specifying a directory path name or simply a file name.

Example 1: Displaying the names of all directories and files in the current drive

```
brk:0>DIR <cr>
 Directory = A:¥
  COMMAND COM PRINT SYS RSDRV   SYS CONFIG SYS  FORMAT EXE
  SPEED   COM SETUP STR IE78330 COM SYMDEB EXE <TEST      >
brk:0>■
```

Example 2: Displaying the names of directories and files for a specified path name

```
brk:0>DIR ¥TEST <cr>
 Directory = A:¥TEST
 <.          >  <..          > TEST1  HEX  TEST1  DBG
brk:0>■
```

8.14 Setting of Trigger Point (DLY)

```
DLY [ F ]
      M
      L
```

trc:0>	×	emu:0>	○	brk:0>	○

F: Sets the trigger point at the beginning of trace memory or sample memory.

M: Sets the trigger point at the middle of trace memory or sample memory.

L: Sets the trigger point at the end of trace memory or sample memory.

The DLY command places the trigger point set with the BRM command at the beginning, middle, or end of trace memory or sample memory. According to the trigger point position, the trace range for the real-time tracer or the internal RAM data sampler is selected.

The position of the trigger point is selected from the following three points:

. Beginning of trace memory or sample memory: F
(The real-time tracer or internal RAM data sampler traces data following the trigger point.)

. Middle of trace memory or sample memory: M
(The real-time tracer or internal RAM data sampler traces data before and behind the trigger point.)

. End of trace memory or sample memory: L
(The real-time tracer or internal RAM data sampler traces data preceding the trigger point.)

When DLY <cr> is entered, the current setting state is displayed.

Example 1: Placing the trigger point

```
brk:0>DLY F <cr>     ← Places trigger point at the beginning of trace
brk:0>■                memory or sample memory.

brk:0>DLY L <cr>     ← Places trigger point at the end of trace memory
brk:0>■                or sample memory.
```

Example 2: Displaying trigger point setting state

```
brk:0>DLY <cr>
 Trigger point     F   M   L
                   |   |   |
 Trace memory     ---------*->   ← Indicates current position of trigger
brk:0>■                            point with *.
```

8.15 Execution of Sub-process (DOS)

DOS

trc:0>	O	emu:0>	O	brk:0>	O

The DOS command executes a command processor as a sub-process. The command processor to be executed has a file name set in COMSPEC in the environment character string table which was specified with the shell command in CONFIG.SYS at the MS-DOS or PC DOS environment setup. By the DOS command, the control program is stopped temporarily.

If there is no file registered in COMSPEC, the path specified in environment setup is used to search and execute COMMAND.COM which is the standard command processor of the MS-DOS and PC DOS.

If the command processor cannot be searched, no sub-process is executed.

If COMMAND.COM which is the standard command processor of the MS-DOS and PC DOS is executed as a sub-process, EXIT <cr> must be entered to return control to the control program.

Example: Executing a sub-process

```
brk:0>DOS <cr>         ← Passes control to OS.
A>                     ← Prompt of OS appears.

A>EXIT <cr>            ← Passes control to control program.
 return from child
brk:0>■                ← Prompt of control program appears.
```

8.16 Setting of Disable Conditions (DSB)

```
        ┌[BRA1][BRA2][BRA3][BRA4][BRD][BRS1][BRS2][BRS3][BRS4] ┐
DSB  [─┤                                                        ├─]
        └OFF                                                    ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

BR? (Note): Disable condition

OFF: Cancels the disable conditions.

Note: BR? is BRA1, BRA2, BRA3, BRA4, BRD, BRS1, BRS2, BRS3, or BRS4 disable condition.

The DSB command sets disable conditions.

When a disable condition is satisfied, no trigger signal is output until the next enable condition is satisfied even when the trigger condition is satisfied.

When sequential enable conditions are set, all the conditions are cleared, and they are detected again starting from ENB1.

Multiple disable conditions can be specified, separated from each other by a space.

When DSB <cr> is entered, the current disable conditions are displayed.

When DSB OFF <cr> is entered, the current disable conditions are canceled.

Example 1: Setting disable conditions

```
brk:0>DSB BRA1 BRA2 <cr>    ← Sets BRA1 and BRA2 as disable conditions.
brk:0>■
```

Example 2: Displaying setting state

```
brk:0>DSB <cr>
 BRA1   BRA2                ← BRA1 and BRA2 are set.
brk:0>■
```

8.17 Setting of Enable Conditions (ENB)

```
        ┌1┐     ┌[BRA1][BRA2][BRA3][BRA4][BRD][BRS1][BRS2][BRS3][BRS4] ┐
ENB ────┤2├─[ ──┤                                                      ├─]
        └3┘     └ON                                                    ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

ENB 1: Specifies enable 1.

ENB 2: Specifies enable 2.

ENB 3: Specifies enable 3.

BR? (Note 1): Enable condition

ON: Specifies each enable condition to be passed.

The ENB command sets enable conditions.

If the enable condition is satisfied entering the enable state, a trigger signal is output when the trigger condition is satisfied.

Multiple enable conditions can be specified, separated from each other by a space.

When more than one enable point is specified, that is, sequential enable specification is made, a trigger becomes valid only when ENB1, ENB2, and ENB3 have been passed in that order.

If an event specified by the DSB command occurs while sequential event detection is being performed in order of ENB1, ENB2, and ENB3, the events detected by that time are invalidated, and the events must be detected again sequentially from the first.

When ENB n ON <cr>(Note 2) is entered, the enable condition is passed and the next enable condition becomes valid. That is, the enable condition is assumed to be satisfied irrespective of whether the enable points have been passed. When ENB1 ON <cr> and ENB2 ON <cr> are entered, for example, only the ENB3 condition needs to be satisfied to cause a trigger.

When ENB n <cr>(Note 2) is entered, a specified enable condition is displayed.

When ENB <cr> is entered, the set state is displayed.

Notes 1. BR? is BRA1, BRA2, BRA3, BRA4, BRD, BRS1, BRS2, BRS3, or BRS4 enable condition.

2. n is a number from 1 to 3.

Example 1: Setting enable conditions

```
brk:0>ENB 1 BRA1 BRS1 <cr>  ← Sets enable condition 1 to BRA1 and BRS1.
brk:0>■

brk:0>ENB 3 BRA4 <cr>       ← Sets enable condition 3 to BRA4.
brk:0>■
```

Example 2: Displaying the ENB1 set state

```
brk:0>ENB 1 <cr>
 1 BRA1 BRS1
brk:0>■
```

Example 3: Displaying all enable conditions

```
brk:0>ENB <cr>
 1 BRA1 BRS1
 2 ON
 3 BRA4
brk:0>■
```

8.18 Display of Event Detector Setting State (EVN)

EVN

trc:0>	O	emu:0>	O	brk:0>	O

The EVN command displays the event set state.

The event set state is displayed in two screens. On the first screen, "Next ? (Y/N)" appears to ask the user whether to shift to the next screen.

When Y <cr> is entered, the next screen appears.

When N <cr> is entered, the command terminates.

Example: Displaying the event state

```
brk:0>EVN <cr>

Enable1          ENB1 ON
Enable2          ENB2 ON
Enable3          ENB3 ON
Disable         DSB  OFF
Trigger         BRM  BRA1 BRS1
Check point     CHK  OFF
Qualify trace   TRX  OFF
Pass count      PAS  1T
Delay count     DLY  M
Trigger out     OUT  ON

External data   BRD  0XXXXY
Program execute BRS1 A=0XXXXH
                BRS2 A=0XXXXH
                BRS3 A=0XXXXH
                BRS4 A=0XXXXH
                                ←- Stopped here temporarily

Next ? (Y/N) Y <cr>             ←- Displays the next screen.

Bus detect      BRA1 A=0XXXXH
                     V=0XXH
                     C=NC
                     E=0XXXXY
                BRA2 A=0XXXXH
                     V=0XXH
                     C=NC
                     E=0XXXXY
                BRA3 A=0XXXXH
                     V=0XXH
                     C=NC
                     E=0XXXXY
                BRA4 A=0XXXXH
                     V=0XXH
                     C=NC
                     E=0XXXXY
```

8.19 Termination of Control Program (EXT)

EXT

trc:0>	×	emu:0>	×	brk:0>	○

The EXT command terminates the IE-78330-R control program and passes control to the OS. Open files are all closed.

Example: Terminating the control program

```
brk:0>EXT <cr>
A>■                 ← Prompt of OS appears
```

8.20 Display of Command History (HIS)

$\boxed{\text{HIS}}$

trc:0>	O	emu:0>	O	brk:0>	O

The HIS command displays up to 20 command lines most recently stored in history memory together with their command numbers in order of storage time.

Whenever a command is executed, it is recorded in history memory automatically.

To read a particular command line, !n <cr> is entered (n is a command number).

To read the latest command line, !! <cr> is entered.

To execute a read command, enter <cr> on that command line.

Example 1: Displaying commands recorded in history memory

```
brk:0>HIS <cr>
  1  LOD TEST
  2  CLK
     .
     .
 20 HIS
brk:0>■
```

Example 2: Reading a particular command by specifying its command number

```
brk:0>!2 <cr>       ← Reads second command in history memory.
 CLK <cr>           ← Enters <cr> to execute this command.
 Internal
brk:0>■
```

8.21 Display of Command Help Screen (HLP)

HLP [command]

trc:0>	○	emu:0>	○	brk:0>	○

command: Command body

The HLP command displays a list of commands or the usage of a particular command.

When a command body is specified, the usage of the command is displayed. When HLP <cr> is entered, a list of commands is displayed, then the system becomes ready for input of a command body.

To terminate the HLP command, <cr> is entered.

Example: Displaying a list of commands and command usage

```
brk:0>HLP <cr>
   Command Table
 ASM       BRA       BRD       BRS       BRM   ┐
 CHK       CVD       CVM       CLK       CNT   │
 COM       DAS       DLY       DSB       DIR   │
 DOS       ENB       EVN       EXT       HIS   │ ← Displays
 HLP       LOD       LST       MAP       MAT   │   command
 MEM       MOD       MOV       OUT       PAS   │   list.
 PGM       PSA       PSD       PST       REG   │
 RES       RUN       SAV       SFR       STP   │
 STR       SYM       TRD       TRG       TRF   │
 TRM       TRS       TRX       VRY       WRD   ┘

HLP>DIR <cr>     ← Specifies DIR command.
      ┌
      │ Explanation of DIR command
      └
HLP> <cr>        ← Terminates help command.
brk:0>■
```

8.22 Loading of Object, Symbols, and Debugging Environment (LOD)

```
LOD  file[module name¥][[D][C][S]][$V]
```

trc:0>	×	emu:0>	×	brk:0>	○

file:	File name
module name¥(Note):	Module name
D:	Specifies debugging environment.
C:	Specifies object code.
S:	Specifies symbols.
$V:	Specifies verification.

Note: When an IBM PC series personal computer is used as the host machine, a backslash (\) must be entered instead of ¥.

The LOD command loads a debugging environment, object code, and symbol table stored in specified files successively or separately.

A required module can be specified to load a symbol table.

When $V is specified, verification is performed in loading. An error detected during loading terminates the LOD command.

When the high-speed download mode is specified at the start of the IE-78330-R, data are loaded through channel 4. If the printer is used by another process, however, the following message appears, and data are loaded through channel 1:

Select serial interface

Example 1: Loading a debugging environment, object code, and symbols successively

```
brk:0>LOD SAMPLE <cr>(Note)
 Debug condition load (Y/N)? Y     ← Asks if debugging environment is
 debug condition load complete  ┐     loaded.  Enter Y.
 object load complete           │
 symbol table loading           │
 PUBLIC     load complete       │  ← Normal termination messages
 MOD01      load complete       │
 MOD02      load complete       ┘
brk:0>■
```

Example 2: Loading a debugging environment

```
brk:0>LOD SAMPLE.DBG D <cr>      ← Add switch D to end of command.
 debug condition load complete  ← Normal termination message
brk:0>■

brk:0>LOD SAMPLE D <cr>          ← When extension of file name is
 debug condition load complete     omitted, .DBG is set.
brk:0>■
```

Note: If neither extension nor switch is specified, a debugging environment, object code, and symbol table are loaded successively.

Example 3: Loading object code

```
brk:0>LOD SAMPLE.HEX C <cr>    ← Add switch C to end of command.
 object load complete         ← Normal termination message
brk:0>■

brk:0>LOD SAMPLE C <cr>        ← When extension of file name is omitted,
 object load complete            HEX is set.
brk:0>■
```

If an error is detected while object code is being loaded, one of the following messages appears:

```
Check sum error.  (516)(Note) ← A checksum error is detected.
Bad character.  (515)(Note) ← An invalid character is found in records.
Non map area access.  (107)(Note) ← An attempt was made to load data
                                     into a memory area not mapped.
```

Note: The numbers in parentheses indicate the error message Nos.

Example 4: Loading a symbol table

```
brk:0>LOD SAMPLE.SYM S <cr>     <- Add switch S to end of command line.
symbol table loading            <- Message telling start of symbol loading
  PUBLIC      load complete     ┐ <- Messages telling end of module
  MOD01       load complete     │    loading
  MOD02       load complete     ┘
brk:0>■

brk:0>LOD SAMPLE S <cr>         <- When extension of file name is omitted,
  symbol table loading             SYM is set.
  PUBLIC      load complete
  MOD01       load complete
  MOD02       load complete
brk:0>■

brk:0>LOD SAMPLE PUBLIC¥ MOD02¥ S <cr>  <- When a module is specified,
  symbol table loading                     only that module is loaded.
  PUBLIC      load complete
  MOD01       pass
  MOD02       load complete
brk:0>■
```

Example 5: When loading of a loaded symbol module is attempted

```
brk:0>LOD SAMPLE S <cr>
  symbol table loading
  PUBLIC      Pass              <- Not loaded
  MOD01       load complete
  MOD02       Pass              <- Not loaded
brk:0>■
```

Example 6: When an error is detected during loading

```
brk:0>LOD SAMPLE PUBLIC¥ MOD02¥ S <cr>
  symbol table loading
  PUBLIC      load complete
  MOD01       pass
  MOD02       Failed.  (518)(Note)  <- Message displayed when error is
brk:0>■                                detected during loading of module
                                       block
```

Note: The number in parentheses indicates the error message No.

8.23 Output of Results to File (LST)

```
         ┌file  ┐
LST   [ ─┤LST:  ├─ ]
         └CON:  ┘
```

trc:0>	O	emu:0>	O	brk:0>	O

file: File name

LST: Printer

CON: Console

The LST command opens a specified device to output the data displayed on the console to the device.

When LST file <cr> is entered, the displayed data are saved in the specified file. If the extension of the file name is omitted, .TXT is set.

When LST LST: <cr> is entered, the displayed data are output to the printer.

The output of data is started by entering ^P. Another input of ^P stops data output.

After a file or printer is opened with the LST command, be sure to enter LST CON: <cr> to close the device when the data have been output.

Example 1: Opening a file named SAMPLE.TXT in drive B

```
brk:0>LST B:SAMPLE.TXT <cr>
brk:0>■
```

Example 2: When a file with the same name as the file to be opened is present in drive B

```
brk:0>LST B:SAMPLE.TXT <cr>
 File already exists. (513)(Note)  B:SAMPLE.TXT Delete? (Y or N): Y <cr>
brk:0>■
```

If a file named SAMPLE.TXT with the R/W attribute is present in drive B, the above message appears.

If Y <cr> is entered in reply to the message, the existing file is deleted, and a new file is opened.

If other than Y <cr> is entered, no file is opened.

Example 3: When the file cannot be opened

```
brk:0>LST B:SAMPLE.TXT <cr>          ← Specifies file with R/O attribute.
 Read only file.  (507)(Note)    B:SAMPLE.
brk:0>■
```

If there is a file named SAMPLE.TXT with the R/O attribute in drive B, the above message appears, and the command is ignored.

Example 4: When the printer is opened normally

```
brk:0>LST LST: <cr>
brk:0>■
```

Example 5: When the printer cannot be opened normally

```
brk:1>LST LST: <cr>
 List device is used by other command.  (109)(Note)
brk:1>
```

If the printer is not available because it is used by another command (process), the above message appears, and the command is ignored.

Note: The numbers in parentheses indicate the error message Nos.

Example 6: Command output by entering ^P

```
  brk:0>LST B:SAMPLE.TXT <cr>  ← Opens file named SAMPLE.TXT in drive B.
  brk:0>RES <cr>
┌ brk:0>^P CLK U <cr>         ← ^P is entered at beginning of command line.
┤ brk:0>OUT ON <cr>
└ brk:0>■
  brk:0>^P                    ← ^P is entered when command input is prompted.

 → Output to SAMPLE.TXT
```

In the above example, commands starting with the command CLK U immediately after ^P and ending with the command OUT ON immediately before another input of ^P are output to an opened file.

8.24 Mapping (MAP)

```
MAP  I  [ partition ]
          0K
          8K
          16K
          24K
          32K
          40K
          48K
          56K

MAP [ T  [partition]]
      W
      R
      U
      K
```

Radix	partition:H

trc:0>	×	emu:0>	×	brk:0>	○

MAP I: Internal ROM
MAP T: Turbo-access-manager alternative memory
MAP W: Alternative memory
MAP R: Write-protected alternative memory
MAP U: User memory
MAP K: Cancels mapping (nonmapping)
partition: Mapping range

The MAP command specifies attributes of memory used by the target device.

For a memory area at 0H to 0DFFFH, the following memory attributes are specified in 8K-byte units.

- o Internal ROM
- o Turbo-access-manager alternative memory

- o Alternative memory
- o Write-protected alternative memory
- o User memory
- o Cancellation of mapping

For the 8K-byte memory area at 0E000H to 0FFFFH (excluding the internal RAM and SFR areas), the following memory attributes are specified:

- o User memory
- o Cancellation of mapping

Executing the MAP command resets the emulation CPU.

The range of the area mapped to internal ROM can be changed only with the MAP I command.

When only a subcommand is specified in command input, the mapping range specified by the subcommand is displayed.

When MAP K <cr> is entered, all mapped areas except internal ROM are canceled.

When MAP <cr> is entered, the current mapping state is displayed.

Example 1: Setting mapping

```
brk:0>MAP I 8K <cr>            ← Allocates addresses 0H to 1FFFH to
brk:0>■                          internal ROM.

brk:0>MAP R 2000,3FFF <cr>     ← Allocates addresses 2000H to 3FFFH to
brk:0>■                          write-protected alternative memory.
```

Example 2: Displaying the mapping state

```
brk:0>MAP <cr>
 0000-1FFF   Internal ROM
 2000-3FFF   R/O emulation
 4000-5FFF   R/W emulation
 6000-7FFF   User
 8000-DFFF   Turbo emulation
 E000-FC7F   Non map
 FD00-FEFF   <Internal RAM>  ← Indicates internal RAM space. This area
brk:0>■                        cannot be manipulated with MAP command.

brk:0>MAP R <cr>             ← Displays write-protected alternative
 2000-3FFF                     memory.
brk:0>■
```

Example 3: Canceling mapping

```
brk:0>MAP K 2000,5FFF <cr>   ← Cancels mapping of addresses 2000H to
brk:0>MAP K 0,1FF <cr>         5FFFH.
 Mapping error. (106)(Note)   ← Error occurs because internal ROM area
brk:0>■                         is specified.
```

Note: The number in parentheses indicates the error message No.

8.25 Math (MAT)

MAT expression	
Radix	expression:H

trc:0>	○	emu:0>	○	brk:0>	○

expression: Specifies an expression.

The MAT command evaluates an expression representation coded in an operand, and displays the results in hexadecimal, decimal, octal, and binary notations in this order.

The following operators can be coded in the expression. Up to 32 levels of parentheses are allowed.

( )	Highest
* /	↑
+ -	Priority
AND	↓
OR XOR	Lowest

All arithmetic and logical operations are performed with 16-bit integers. If an intermediate or final result of an operation is longer than 16 bits, the bits other than the low-order 16 bits are discarded.

Example: Operation

```
brk:0>MAT 5H+7H AND 17Q <cr>    ← Enters an expression.
 0CH,12T,14Q,1100Y              ← Displays results of operation.
brk:0>■
```

8.26 Memory Manipulation (MEM)

```
MEM C [word]

         ┌word      ┐
MEM D [─┤          ├─]
         └partition ┘

MEM E [partition]

     ┌ F ┐
MEM ─┤   ├─partition data-string
     └ G ┘

     ┌ M ┐
MEM ─┤ V ├─Partition word
     └ X ┘
```

Radix	word:H partition:H data-string:H

trc:0>	X	emu:0>	*	brk:0>	O

MEM C:	Changes memory contents.
MEM D:	Displays memory contents.
MEM E:	Checks memory.
MEM F:	Initializes memory contents.
word:	Start address
data-string:	Data string
MEM G:	Searches memory contents.
MEM M:	Copies memory contents.
MEM V:	Compares memory contents.
MEM X:	Exchanges memory contents.
partition:	Address range

The MEM command changes, displays, initializes, or searches memory contents in units of a memory length specified by the WRD command.

When MEM <cr> is entered, the contents of memory are displayed.

* When emu:0> appears, only MEM C and MEM D can be executed.

(1) MEM C command

MEM C [word]	
Radix	word:H

trc:0>	×	emu:0>	○	brk:0>	○

word: Start address for change

The MEM C command changes the contents of memory mapped at addresses 0H to 0FEFFH.

When the start address for change is specified in an operand, memory contents starting at the specified address are changed successively.

When the contents of a memory location is left unchanged, <cr> is entered. To terminate change operation, <cr> or <ESC> is entered.

When MEM C <cr> is entered, the address next to the memory locations for the previous change operation is assumed as the start address for memory change.

Caution: When a command to change memory contents is entered during trace (trc:0> mode) or emulation (emu:0> mode), execution of the emulation CPU is temporarily suspended.

Example 1: Changing memory contents in bytes
(When B is specified in the WRD command)

```
brk:0>MEM C 100 <cr>    ← Changes memory contents starting at
 0100  00 11 <cr>          address 100H.
 0101  11 22 <cr>
 0102  22 .<cr>         ← Terminates change of memory contents.
brk:0>■
```

Example 2: Changing memory contents in words
(When W is specified in the WRD command)

```
brk:0>MEM C 100 <cr>    ← Changes memory contents starting at
 0100  1100 2211 <cr>      address 100H.
 0102  3322 4433 <cr>
         .
         .

         .
         .
 0108  9988 .<cr>       ← Terminates change of memory contents.
brk:0>■
```

(2) MEM D command

MEM D [{ word | partition }]	
Radix	word:H partition:H

trc:0>	×	emu:0>	○	brk:0>	○

word: Start address for display
partition: Display address range

The MEM D command displays the contents of mapped memory locations at addresses 0H to 0FEFFH.

When the start memory address for display is specified in an operand, 11 lines of memory contents starting at the specified address are displayed. When a memory range for display is specified, the contents of the range are displayed.

When MEM D <cr> is entered, 11 lines of memory contents are displayed, starting at the address next to the memory locations for the previous change operation.

On a byte basis, memory contents are displayed in hexadecimal notation and ASCII characters.

On a word basis, memory contents are displayed in hexadecimal notation only.

Table 8-1 ASCII Character Data Displayed in Bytes

		Low-order 4 bits															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
High-order 4 bits	2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	.	.

Example 1: Displaying memory contents in bytes
(When B is specified in the WRD command)

```
brk:0>MEM D 100,17F <cr>  ← Displays memory contents at addresses 100H to 17FH.
 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    !"#$%&'()*+,-./

 Address                Data (16 bytes)                     ASCII characters
brk:0>■
```

Example 2: Displaying memory contents at addresses 100H to 12FH in words (When W is specified in the WRD command)

```
brk:0>MEM D 100, 12F <cr>  ← Displays memory contents at addresses 100H to 12FH.
 0100   0100 0302 0504 0706 0908 0B0A 0D0C 0F0E
 0110   3130 3332 3534 3736 3938 3B3A 3D3C 3F3E
 0120   4140 4342 4544 4746 4948 4B4A 4C4D 4F4E
brk:0>■
```

(3) MEM E command

MEM E [partition]	
Radix	partition:H

trc:0>	×	emu:0>	×	brk:0>	○

partition: Address range to be checked

The MEM E command checks the mapped memory area at addresses 0H to 0FE7FH.

A particular memory area can be checked when a range to be checked is specified in an operand, or whole mapped memory can be checked when MEM E <cr> is entered.

Example: Checking memory

```
brk:0>MEM E 0XXX <cr>         ← Checks memory at 0000H to 0FFFH.
 complete                     ← Normal termination message
brk:0>■

brk:0>MEM E 1000,1FFF <cr>    ← Checks memory at 1000H to 1FFFH.
 1021                         ← Indicates address at which error
brk:0>■                         is detected in memory check.
```

If an error is detected, the subsequent locations are not checked.

(4) MEM F command

MEM F partition data-string	
Radix	partition:H data-string:H

trc:0>	×	emu:0>	×	brk:0>	○

partition: Address range for initialization
data-string: Initialization data

The MEM F command initializes the contents of the mapped memory area at addresses 0H to 0FE7FH with given data.

Up to 10 data strings can be specified. Mask data can be coded as initialization data. When mask data are specified, the masked bits are left unchanged.

Example 1: Initializing memory contents with data strings in bytes
(When B is specified in the WRD command)

```
brk:0>MEM F 100,1FF 1,2,3,4,5,6,7,8,9,0 <cr>  ← Initializes memory contents at
brk:0>■                                          addresses 100H to 1FFH to byte
                                                 data strings of 1, 2, 3, 4, 5,
                                                 6, 7, 8, 9, 0.
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>■
```

Example 2: Initializing memory contents with mask data

```
brk:0>MEM F 100,11F 3X <cr>
brk:0>■
```

Example 3: Displaying memory contents after initialization

```
brk:0>MEM D 100,13F <cr>
 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>■
```

Remark: When memory contents are initialized with mask data 3X, the low-order four bits are not initialized, as shown in the above example.

Example 4: Initializing memory contents with data strings in words
(When W is specified in the WRD command)

```
brk:0>MEM F 100,17F 1,2,3,4,5,6,7,8,9,0 <cr>  ← Initializes memory contents at
brk:0>■                                          addresses 100H to 17FH to word
                                                 data strings of 1, 2, 3, 4, 5,
                                                 6, 7, 8, 9, 0.

brk:0>MEM D 100, 17F <cr>
 0100    0001 0002 0003 0004 0005 0006 0007 0008    .............
 0110    0009 0000 0001 0002 0003 0004 0005 0006    .............
 0120    0007 0008 0009 0000 0001 0002 0003 0004    .............
 0130    0005 0006 0007 0008 0009 0000 0001 0002    .............
                           .
                           .

                           .
                           .
 0170    0007 0008 0009 0000 0001 0002 0003 0004    .............
brk:0>■
```

Example 5: Initializing memory contents with mask data

```
brk:0>MEM F 100,11F 808X <cr>

brk:0>MEM D 100, 13F <cr>
 0100   8081 8082 8083 8084 8085 8086 8087 8088    .............
 0110   8089 8080 8081 8082 8083 8084 8085 8086    .............
 0120   0007 0008 0009 0000 0001 0002 0003 0004    .............
 0130   0005 0006 0007 0008 0009 0000 0001 0002    .............
brk:0>■
```

Remark: When memory contents are initialized with mask data 808X, the low-order four bits are not initialized, as shown in the above example.

(5) MEM G command

MEM G partition data-string	
Radix	partition:H data-string:H

trc:0>	×	emu:0>	×	brk:0>	○

partition: Address range to be searched
data string: Data string to be searched for

The MEM G command searches the mapped memory locations at addresses 0H to 0FE7FH for specified data strings.

Up to 10 data strings can be specified as the data to be searched for. Mask data can be coded as the search data.

Example 1: Searching memory contents in bytes
(When B is specified in the WRD command)

```
brk:0>MEM G 100,11F 30,31,32 <cr> ← Searches memory at addresses 100H to 11FH
 0109                               for consecutive data strings 30, 31, and 32.
 0113     Addresses at which data strings are found
 011D
brk:0>■
```

```
brk:0>MEM D 100,11F <cr>
 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
brk:0>■
```

Example 2: Searching memory contents in words
(When W is specified in the WRD command)

```
brk:0>MEM G 100,12F 12,13,14 <cr> ← Searches memory at addresses 100H to 12FH
 0124                               for consecutive data strings 12, 13, and 14.
brk:0>■
```

```
brk:0>MEM D 100,12F <cr>
 0100   0000 0001 0002 0003 0004 0005 0006 0007    ................
 0110   0008 0009 000A 000B 000C 000D 000E 000F    ................
 0120   0010 0011 0012 0013 0014 0015 0016 0017    ................
brk:0>■
```

(6) MEM M command

MEM M partition word	
Radix	partition:H word:H

trc:0>	×	emu:0>	×	brk:0>	○

word: Start address of copy destination
partition: Address range of copy source

The MEM M command copies memory contents in the mapped area at addresses 0H to 0FE7FH.

The memory contents of a specified copy range are copied to the copy destination starting at the specified addresses.

Example: Copying memory contents

```
brk:0>MEM M 100,10F 120 <cr>  ← Copies memory contents at addresses 100H to
brk:0>■                         10FH to memory locations starting at address
                                120H.
brk:0>MEM D 100,13F <cr>      ← Displays memory contents after copy operation.
 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  31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36   1234567890123456
 0130  09 00 01 02 03 04 05 06 07 08 09 00 01 02 03 04   ................
brk:0>■
```

(7) MEM V command

MEM V partition word	
Radix	partition:H word:H

trc:0>	×	emu:0>	×	brk:0>	○

partition: Address range of data with which the data specified with the start address are compared

word: Start address of compared data

The MEM V command compares data in the mapped memory area at addresses 0H to 0FE7FH.

The data in a specified memory range are compared with the data at a specified start address and subsequent addresses.

If the data do not match, discrepant data and their addresses are displayed.

Example: Comparing memory contents

```
brk:0>MEM V 100,10F 110 <cr>  ← Compares memory contents at addresses 100H to
 Source   destination             10FH with memory contents starting at address
 0100 01-0110 30                  110H.
 0101 02-0111 41
 0105 06-0115 00              ← Displays discrepant data and their addresses.
 010D 04-011D 01
brk:0>■
```

```
brk:0>MEM D 100,11F <cr>
 0100 01 02 03 04 05 06 07 08 09 00 01 02 03 04 05 06  .........
 0110 30 41 03 04 05 00 07 08 09 00 01 02 03 01 05 06  .........
brk:0>■
```

(8) MEM X command

MEM X partition word	
Radix	partition:H word:H

trc:0>	×	emu:0>	×	brk:0>	○

partition: Address range of the other data to be exchanged

word: Start address of data to be exchanged

The MEM X command exchanges data in the mapped memory area at addresses 0H to 0FE7FH.

The memory contents in a specified range and the data at a specified start address and subsequent addresses are exchanged.

Example: Exchanging memory contents

```
brk:0>MEM D 100,12F <cr>
 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   ................
brk:0>■

brk:0>MEM X 100,10F 120 <cr>  ← Exchanges memory contents at addresses 100H to
brk:0>■                          10FH and memory contents starting at address
                                 120H.
brk:0>MEM D 100,12F <cr>  ← Displays memory contents after exchange operation.
 0100  03 04 05 06 07 08 09 00 01 02 03 04 05 06 07 08   ................
 0110  37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32   7890123456789012
 0120  31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36   1234567890123456
brk:0>■
```

8.27 Setting the Channel 2 Mode (MOD)

```
             ┌CHAR┐             ┌19200┐            ┌7┐          ┌NON ┐          ┌1┐
MOD [MODE=───┤    ├─][BAUD= ────┤ 9600├─][LONG= ───┤ ├─][PAR=───┤EVEN├─][STOP= ─┤ ├─]
             └FLOW┘             │ 4800│            └8┘          └ODD ┘          └2┘
                                │ 2400│
                                │ 1200│
                                │  600│
                                └  300┘
```

trc:0>	×	emu:0>	○	brk:0>	○

MODE: Handshaking mode

BAUD: Baud rate

LONG: Character length

PAR: Parity bit

STOP: Stop bit

The MOD command sets the communication mode for channel 2. The communication mode can be set in the interactive format or by specifying the operands of the MOD command on a line.

When MOD <cr> is entered, the communication mode is set in the interactive format.

Example 1: To set the communication mode by specifying the operands on a line

```
brk:0>MOD MODE=CHAR BAUD=4800 LONG=8 PAR=NON STOP=2 <cr>   ← Sets on a
brk:0>■                                                      line.
```

Example 2: To set the communication mode in the interactive format

```
brk:0>MOD <cr>           ← Sets in the interactive format.
 Mode   CHAR = FLOW <cr> ← The mode is changed to flow control.
 Baud   4800 = 9600 <cr> ← The baud rate is changed to 9600 baud.
 Long      8 = <cr>      ← The character length is not changed.
 Par     NON = EVEN <cr> ← The parity bit is changed to even.
 Stop      2 = 1 <cr>    ← The stop bit is changed to 1.
brk:0>■
```

8.28 Moving Data between Alternate Memory and User Memory (MOV)

MOV [U / I] partition word [$V]	
Radix	word:H partition:H

trc:0>	×	emu:0>	×	brk:0>	○

MOV U:	From alternate memory to user memory
MOV I:	From user memory to alternate memory
partition:	Address range of the source
word:	Start address of the destination
$V:	Verify specification

The MOV command moves the contents of a memory location addressed by partition to a memory area starting with word. Data in alternate memory, turbo access manager alternate memory, or write-protected alternate memory in the IE-78330-R is moved to/from memory in a target system.

The MOV U command moves data from alternate memory, turbo access manager alternate memory, or write-protected alternate memory in the IE-78330-R to memory in a target system.

The MOV I command moves data from memory in a target system to alternate memory, turbo access manager alternate memory, or write-protected alternate memory in the IE-78330-R.

The address range of destination memory must be mapped, but the source memory need not be mapped.

When $V is specified at the end of a command line, the verify function is added.

Example 1: To move the contents of alternate memory to user memory with the verify function

```
brk:0>MOV U 4000,7FFF 4000 $V <cr>   ← This sample example moves the
brk:0>■                                 data of alternate memory from
                                        4000H to 7FFFH to RAM at
                                        addresses 4000H and later in the
                                        target system with the verify
                                        function.
```

Example 2: To move the contents of user memory to alternate memory

```
brk:0>MAP <cr>
 0000 - 3FFF  Internal ROM
 4000 - 7FFF  R/W emulation        ← The destination memory is mapped to
 8000 - FC7F  Non map                alternate memory.
 FC80 - FEFF <Internal RAM>
brk:0>
brk:0>MOV I 4000,7FFF 4000 <cr>    ← Moves the data of target memory from
brk:0>■                              4000H to 7FFFH to addresses 4000H
                                     and later of alternate memory.
```

Example 3: When data could not be moved correctly

```
brk:0>MOV U 4000,7FFF 4000 $V <cr>
 6200H              ┐             ← The addresses where data could not
 6300H              │               be moved correctly are displayed.
 7000H              ┘
brk:0>■
```

8.29 Specifying the External Output of a Trigger Signal (OUT)

```
        ┌OFF┐
OUT  [──┤   ├──]
        └ON ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

OFF: External sense clip No. 1 is used for trace signal input.

ON: External sense clip No. 1 is used for trigger signal output.

The OUT command specifies whether external sense clip No. 1 is to be used for trace signal input or output.

When OUT <cr> is entered, the current state (ON or OFF) is displayed.

When external sense clip No. 1 is used for trigger signal output (when OUT ON is entered), disconnect the clip from the signal output line of the target system. Otherwise, the target system or the IE-78330-R may be damaged.

Example 1: Output specification

```
brk:0>OUT ON <cr>   ← Sets trigger signal output.
brk:0>■
```

Example 2: When No. 1 is connected to the output line

```
brk:0>OUT ON <cr>
 External trigger line short.  (204)(Note)  ← Because signals from the
brk:0>■                                        output line collide with
                                               signals from clip No. 1,
                                               the command is rejected.
```

Note: The number in parentheses indicates the error message No.

Example 3: To display the current state

```
brk:0>OUT <cr>  ← Displays the current state.
 OFF            ← Trace signal input specification
brk:0>■
```

8.30 Setting a Pass Condition (PAS)

PAS [pass8]	
Radix	pass8 : T

trc:0>	×	emu:0>	○	brk:0>	○

pass8: Pass condition (pass count)

The PAS command sets the number of events to be caused.

When PAS <cr> is entered, the set pass condition is displayed.

Example 1: To set a pass condition

```
brk:0>PAS 7 <cr>   ← Sets the pass count to 7.
brk:0>■
```

Example 2: To display the set pass condition

```
brk:0>PAS <cr>
 7T                ← 7 is now set.
brk:0>■
```

8.31 Controlling a PROM Programmer, Changing Some Control Characters or Deleting the Control Characters (PGM)

PGM [C]

trc:0>	×	emu:0>	×	brk:0>	○

C: Specification for changing some control characters or deleting the control characters

The PGM command sets the IE-78330-R in the terminal mode for remote operations of an NEC PROM programmer (PG-1500 or PG-2000) which is connected to channel 2.

To set the IE-78330-R in the terminal mode, enter PGM <cr>.

To terminate the terminal mode, enter ^Z.

Cautions 1. When an NEC PROM programmer other than PG-1500 and PG-2000 is connected to channel 2 and the IE-78330-R is used as a terminal, the following control characters cannot be used.

^A ^B ^C ^D ^E ^F ^H ^I ^J ^K ^L ^M
^N ^Q ^S ^W ^Z

When one or more of the above control characters are to be used, enter PGM C <cr> to release the restrictions on the use of the control characters or to change some of the control characters.

Cautions 2. Control characters are changed interactively. The following 16 control characters can be used.

^A ^B ^E ^F ^G ^N ^O ^P ^R ^T ^U ^V
^W ^X ^Y ^Z

The same control character cannot be specified more than once.

When the DEL key is pressed or ^H is entered, the initial values of the control characters are displayed.

When the escape key is pressed during change of control characters, the changed control characters are invalidated.

3. When an attempt is made to change the restrictions on the use of the control characters, the system automatically enters the terminal mode.

Example 1: When a PG-1500 is connected to channel 2 and the IE-78330-R is used as a terminal

```
brk:0>PGM <cr>
Beginning of PGM mode    ← Message indicating the start of a PGM command
PG>                      ← Prompt output from the PG

PG>^Z                        ← Terminates the terminal mode.
 Exit PGM mode (Y/N) Y       ← Enter Y in response to the confirmation
                               message for the end of the terminal mode.
 Termination of PGM mode     ← Message indicating the end of the terminal
brk:0>■                        mode
```

Example 2: To make all the control characters valid

```
brk:0>PGM C <cr>
 Put through all control character (Y/N) Y <cr>  ← All the control characters
 Beginning of PGM mode                              are made valid.

 ^Z                         ← When ^Z is entered, the following message is output.
 EXIT PGM mode (Y/N) Y <cr>  ← ^Z is sent to CH2 and the terminal mode does not
                               end.

 ^Z
 EXIT PGM mode (Y/N) Y <cr>  ← Enter Y to terminate the terminal mode.
  Termination of PGM mode     ← Message indicating the end of the terminal mode
 brk:0>■
```

Example 3: To change some of the control characters

```
brk:0>PGM C <cr>
 Put through all control character (Y/N) N <cr>

 Termination of "PGM"      ...^Z
 Beginning   of "HEX LOAD" ...^A
 Beginning   of "HEX SAVE" ...^E
 Beginning   of "SYM LOAD" ...^N     ← The initial values are displayed.
 Termination of "LOAD"     ...^B
 Termination of "SAVE"     ...^F
 Break       of "LOAD/SAVE"...^W

 Termination of "PGM"      ...^Z ■  ← The system displays the current value and
                                       waits for entry.
```

. Enter any characters.

```
 Termination of "PGM"      ...^A <cr>   ← ^Z is changed to ^A.
 Beginning   of "HEX LOAD" ...^Z <cr>   ← ^A is changed to ^Z.
 Beginning   of "HEX SAVE" ...^N <cr>   ← ^E is changed to ^N.
 Beginning   of "SYM LOAD" ...^E <cr>   ← ^N is changed to ^E.
 Termination of "LOAD"     ...^B <cr>   ← No change
 Termination of "SAVE"     ...^X <cr>   ← ^F is changed to ^X.
 Break       of "LOAD/SAVE"...^G <cr>   ← ^W is changed to ^G.
```

. After the changes, the system displays the following message and enters the PGM mode.

```
 Beginning of PGM mode
```

8.32 Setting Sampling Addresses (PSA)

PSA [word][word][word]

trc:0>	×	emu:0>	○	brk:0>	○

word: Sample address

The PSA command specifies the address of an internal RAM to be sampled.

Up to three sample addresses can be set from 0FE00H to 0FEFFH. When an odd address is set, the odd address and the even address immediately before the odd address are used as a pair. This is because data are sampled in words.

When PSA <cr> is entered, the set addresses are displayed.

When no sample address is set, -- is displayed.

Example 1: To set three parameter addresses

```
brk:0>PSA 0FE40 0FE42 0FE44 <cr>   ← Three sample addresses are set.
brk:0>■

brk:0>PSA <cr>
 0FE40 0FE42 0FE44                 ← The odd addresses after even
brk:0>■                              addresses are also set.
```

Example 2: To display the current set state

```
brk:0>PSA <cr>                     ← Displays the current set state.
 --
brk:0>■
```

8.33 Displaying Sample Data (PSD)

```
          ┌ALL  ┐
          │F    │   ┌$W┐
PSD  [─┤T    ├─][─┤$B├─]
          │L    │   └$Y┘
          └point┘
```

Radix	point:T

trc:0>	✕	emu:0>	○	brk:0>	○

ALL: Displays all sample data.

F: Displays data from the beginning of sample data.

T: Displays data at the event detection point together with the preceding five lines and the following five lines.

L: Displays data from the end of sample data.

point: Displays data from a specified sample frame No.

$W: Displays data in units of words.

$B: Displays data in units of bytes.

$Y: Displays data in units of bits.

The PSD command displays sample data which has been written into sampler memory. Sample data is obtained by sampling internal RAM data using the three addresses preset by the SPA command. In this case, data are sampled at intervals set by the PST command during real-time emulation.

The sample data displayed first is determined according to the position of the current sample pointer or a specified sample pointer.

When emulation terminates, the sample pointer is placed at the last frame.

The sample pointer can be specified in either of the following ways:

. Specify the pointer in the prompt mode after sample data are displayed.

. Specify the pointer by specifying the operands in the SPD command.

The table below shows the relationship between the sample pointer specification and frames to be displayed.

	Prompting command	Function
Absolute address	frame No.	Displays a specified frame and 10 lines following the frame.
+/cr/- movement	+/cr	Displays 11 lines following a frame that has just been displayed.
	-	Displays 11 lines preceding a frame that has just been displayed.
Sample pointer setting	F	Displays the first 11 lines.
	L	Displays the last 11 lines.
Trigger retrieval	T	Displays the trigger frame together with the preceding five frame and the following five lines.

External data are displayed in units of bits irrespective of the display specification.

Cautions 1. After the sample address is set by the PSA command, sample data remain undefined until a write operation is performed.

2. Sample data are cleared when real-time execution is performed again.

3. The internal RAM data sampler is stopped during single-step execution or procedure execution.

Entering PSD <cr> displays the frame at which the pointer is currently placed and 10 lines immediately following the frame.

When PSD <cr> is entered on completion of data sampling, the same data as with the PSD T command are displayed.

The default for $W, $B, or $Y of the PSD command is the memory length set with the WRD command.

Example 1: To display data that have just been sampled in units of bytes
(When B is specified in the WRD command)

```
brk:0>PSD <cr>
 frame    FE40   FE41   FE42   FE43   FE44   FE45
  0051     00     23     CF     23     A0     CF
  0052     00     23     CA     27     A0     CF
  0053     00     23     C5     2C     A0     CF
  0054     00     23     C0     33     A0     CF
  0055     00     23     B8     30     A0     CF
 T0056     00     23     B4     2F     A0     CF
  0057     00     23     A6     35     A0     CF
  Total frame = 0057   (F/L/T/+/cr/-/Frame No./.) ? F <cr>
  0000     45     23     AA     23     A0     CF
  0001     45     23     AA     27     A0     CF
  0002     45     23     AA     2C     A0     CF
  0003     45     23     AA     33     A0     CF
  0004     45     23     AA     30     A0     CF
  0005     00     23     AA     23     A0     CF
  0006     00     23     CA     27     A0     CF
  0007     00     23     C5     2C     A0     CF
  0008     00     23     C0     33     A0     CF
  0009     00     23     B8     30     A0     CF
  0010     00     23     B8     30     A0     CF
  Total frame = 0057   (F/L/T/+/cr/-/Frame No./.) ? <ESC>
brk:0>■
```

Example 2: To display data that have just been sampled in units of words
(When W is specified in the WRD command)

```
brk:0>PSD <cr>
 frame   FE40    FE42    FE44
  0051   2300    23CF    CFA0
  0052   2300    27CA    CFA0
  0053   2300    2CC5    CFA0
  0054   2300    33C0    CFA0
  0055   2300    30B8    CFA0
 T0056   2300    2FB4    CFA0
  0057   2300    35A6    CFA0
  Total frame = 0057   (F/L/T/+/cr/-/Frame No./.) ? <ESC>
brk:0>■
```

Example 3: When $W is specified in the PSD command

```
brk:0>PSD 5 $W <cr>
 frame   FE42    FE44
  0005   23CF    CFA0
  0006   2567    CFA0
  0007   2568    CFA0
  0008   2569    CFA0
  Total frame = 0008   (F/L/T/+/cr/-/Frame No./.) ? <ESC>
brk:0>■
```

Example 4: When $B is specified in the PSD command

```
brk:0>PSD 5 $B <cr>
 frame   FE42   FE43   FE44   FE45
  0005   CF     23     A0     CF
  0006   67     25     A0     CF
  0007   68     25     A0     CF
  0008   69     25     A0     CF
  Total frame = 0008   (F/L/T/+/cr/-/Frame No./.) ? <ESC>
brk:0>■
```

Example 5: When $Y is specified in the PSD command

```
brk:0>PSD 5 $Y <cr>
 frame   FE42      FE43      FE44      FE45
  0005   11001111  00100011  10100000  11001111
  0006   01100111  00100101  10100000  11001111
  0007   01101000  00100101  10100000  11001111
  0008   01101001  00100101  10100000  11001111
  Total frame = 0008   (F/L/T/+/cr/-/Frame No./.) ? <ESC>
brk:0>■
```

Example 6: When only one sample address is set by the PSA command

```
brk:0>PSD 51 <cr>
 frame   FE40
  0051   2300
  0052   2300
  0053   2300
  0054   2300
  0055   2300
 T0056   2300
  0057   2300
  Total frame = 0057   (F/L/T/+/cr/-/Frame No./.) ? <ESC>
brk:0>■
```

Example 7: When no sample data is found

```
brk:0>PSD T <cr>
 No sampled data.  (206)(Note)
brk:0>■
```

Example 8: When no address is specified by the PSA command

```
brk:0>PSD T <cr>
 No PSA address.  (205)(Note)
brk:0>■
```

Note: The numbers in parentheses indicate the error message Nos.

8.34 Setting the Sample Timing (PST)

```
        ┌number┐
PST [ ─┤ .4    ├─ ]
        │ .6    │
        └ .8   ┘
```

Radix	number : T

trc:0>	×	emu:0>	○	brk:0>	○

number:	Sample timing
.4:	0.4 usec
.6:	0.6 usec
.8:	0.8 usec

The PST command sets the sample timing of internal RAM data.

The specifiable timing values are 0.4 us, 0.6 us, 0.8 us, and 1 to 10000 us in steps of microseconds.

Example 1: To set the sample timing to 0.8 us

```
brk:0>PST .8 <cr>
brk:0>■
```

Example 2: To display the current setting

```
brk:0>PST <cr>
 0.8 usec
brk:0>■
```

8.35 Manipulating Registers (REG)

```
               ┌PSW flag name ┐
REG  C  [-┤               ├-]
               └register name ┘

               ┌ALL           ┐
REG  D  [-┤ PSW flag name ├-]
               └register name ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

ALL:	All registers in all register banks
PSW flag name:	UF, RBS2, RBS1, RBS0, S, Z, RSS, AC, IE, P/V, LT, CY
register name:	PC, SP, PSW, R0, R1, R2, R3, R4, R5, R6, R7, RP4, RP5, RP6, RP7, X, A, B, C, VP, UP, DE, HL

The REG command displays or changes general registers, PC, SP, and PSW.

The program status word (PSW) can be displayed or changed in units of flags.

When REG <cr> is entered, the system displays the contents of the registers in the current register bank and the contents of the PC, SP, and PSW.

Caution: When data are read from or written into a register during trace (trc:0> mode) or emulation (emu:0> mode), execution of the emulation CPU is temporarily suspended.

(1) REG C command

```
                ┌PSW flag name ┐
REG  C  [─┤                        ├─]
                └register name ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

PSW flag name: UF, RBS2, RBS1, RBS0, S, Z, RSS, AC, IE, P/V, LT, CY

register name: PC, SP, PSW, R0, R1, R2, R3, R4, R5, R6, R7, RP4, RP5, RP6, RP7, X, A, B, C, VP, UP, DE, HL

The REG C command changes the contents of a specified register and later registers sequentially.

When PSW is specified, UF to CY can be changed in units of bits in that order. When a PSW flag name is specified, the user can change the specified flag and later flags sequentially.

When REG C <cr> is entered, the user can change the general registers (R0 to R7 and RP4 to RP7), PC, and SP sequentially.

When a register is not to be changed, press the return key only. To terminate the change processing, press the escape key.

Example 1: To change the contents of the general registers in the current register bank and the contents of the PC and SP

```
brk:0>REG C <cr>
 R0(X)       00 = 10 <cr>
 R1(A)       10 = 20 <cr>
              .
              .
 SP        9000 =  <cr>        ← No change
brk:0>■
```

Example 2: To change all the PSW flags sequentially

```
brk:0>REG C PSW <cr>
 UF            0 = 1 <cr>
 RBS2          1 = 0 <cr>
               .
               .
 CY            0 = 1 <cr>
brk:0>■
```

Example 3: To specify a PSW flag and change the contents of the flag and later flags sequentially

```
brk:0>REG C RBS2 <cr>
 RBS2          1 = 0 <cr>
 RBS1          1 = <cr>
 RBS0          0 = 1 <cr>
 S             1 = <ESC>      ← Terminates the change processing.
brk:0>■
```

(2) REG D command

```
REG D [ ALL           ]
        PSW flag name
        register name
```

trc:0>	×	emu:0>	○	brk:0>	○

ALL:	All registers in all register banks
PSW flag name:	UF, RBS2, RBS1, RBS0, S, Z, RSS, AC, IE, P/V, LT, CY
register name:	PC, SP, PSW, R0, R1, R2, R3, R4, R5, R6, R7, RP4, RP5, RP6, RP7, X, A, B, C, VP, UP, DE, HL

When ALL is specified, the system displays the contents of all the registers in all register banks (banks 0 to 7).

When PSW is specified, the system displays the contents of all the PSW flags.

When REG D <cr> is entered, the system displays the contents of all the registers in the current bank and the contents of the PC, SP, and PSW.

Example 1: To display the contents of a specified register

```
brk:0>REG D PC <cr>   ← Displays the PC contents.
 PC        1000
 brk:0>■
```

Example 2: To display all the registers in all register banks

```
brk:0>REG D ALL <cr>
 PC     SP   PSW:  UF   RBS2 RBS1 RBS0   S  Z  RSS   AC  IE  P/V  LT  CY
1000   7000         0    0    1    1     0  1   0     0   0   1    0   1

         R0   R1   R2   R3   R4   R5   R6   R7      RP4    RP5    RP6    RP7
 BANK0   01   02   03   04   05   06   07   08      0909   0A0A   0B0B   0C0C
 BANK1   11   12   13   14   15   16   17   18      1919   1A1A   1B1B   1C1C
 BANK2   21   22   23   24   25   26   27   28      2929   2A2A   2B2B   2C2C
 BANK3   31   32   33   34   35   36   37   38      3939   3A3A   3B3B   3C3C
 BANK4   98   76   54   32   10   AA   BB   CC      DEF0   5562   1F20   3434
 BANK5   00   00   00   00   22   3F   1C   52      0006   AEFC   7000   1020
 BANK6   11   22   33   44   55   66   77   88      9999   AAAA   BBBB   CCCC
 BANK7   71   72   73   74   75   76   77   78      7979   7A7A   7B7B   7C7C
brk:0>■
```

Example 3: To display the contents of all the registers in the current bank

```
brk:0>REG D <cr>
 PC      SP   PSW:  UF   RBS2 RBS1 RBS0   S  Z  RSS   AC  IE  P/V  LT  CY
 1000   7000         0    0    1    1     0  1   0     0   0   1    0   1
 R0      R1    R2    R3   R4   R5   R6     R7     RP4    RP5   RP6    RP7
  X       A     C     B                           VP     UP    DE     HL
 31      32    33    34   35   36   37     38     3939   3A3A  3B3B   3C3C
brk:0>■
```

8.36 Resetting IE-78330-R and Emulation Device (RES)

RES [H]

trc:0>	O	emu:0>	O	brk:0>	O

H: IE-78330-R resetting

The RES command can reset the IE-78330-R and emulation device.

. Emulation device resetting: RES <cr>
. IE-78330-R resetting: RES H <cr>

When RES <cr> is entered during trace (in the trc:0> prompt mode) or emulation (in the emu:0> prompt mode), execution stops and a break occurs (the system enters the brk:0> prompt mode).

Example 1: To reset the emulation device

```
brk:0>RES <cr>
brk:0>■
```

Example 2: To reset the IE-78330-R

```
brk:0>RES H <cr>
  [  IE-78330-R start message
brk:0>■
```

8.37 Execution (RUN)

```
RUN  N  [word]

RUN  B  [word]
                  ┌(*1)   ┐
RUN  T  [word][, ─┤       ├─][REG][TRD]
                  └step16─┘
                  ┌(*1)   ┐
RUN  T  [word][, ─┤       ├─][REG][TRD]PRC
                  └step16─┘

        (*1)                        (*2)
  ┌register      ┌ =  ┐       ┌mask8  ┐ ┐
  │ name         │ >  │      ─┤       ├─│
 ─┤             ─┤ <  ├─      └mask16─┘ ├─
  │              │ => │                 │
  │              │ >= │                 │
  │              │ =< │                 │
  │              │ <= │                 │
  │              │ >< │                 │
  │              └ <> ┘                 │
  └PSW flag name = bit                 ─┘
```

Radix	word:H	step16:T	mask8:H	mask16:H	bit:Y

trc:n>	×	emu:n>	×	brk:n>	○

*1 To be substituted
*2 For macros, only =, <>, and >< are valid.

RUN N:	Nonbreak real-time execution
RUN B:	Real-time execution with breaks
RUN T:	Step execution
RUN T PRC:	Procedure execution
word:	Execution start address
PRC:	Procedure execution specification
register name:	PC, SP, R0, R1, R2, R3, R4, R5, R6, R7, RP4, RP5, RP6, RP7, X, A, B, C, VP, UP, DE, HL
PSW flag name:	UF, RBS2, RBS1, RBS0, S, Z, RSS, AC, IE, P/V, LT, CY
REG:	Register display specification
TRD:	Trace display specification
bit:	Single-bit numeric value
step16:	Number of 16-bit steps
mask8:	8-bit mask data
mask16:	16-bit mask data

The RUN command starts execution of a target program by the emulation CPU.

There are four types of execution functions depending upon the specification of subcommands or operands.

An execution start address is specified in the operand.

A specified start address must not exceed 0FDFFH.

(1) RUN N command

RUN N [word]	
Radix	word:H

trc:0>	×	emu:0>	×	brk:0>	○

word: Execution start address

The RUN N command executes a target program from a specified address, and stops the real-time tracer or internal RAM data sampler when the delay conditions are satisfied.

A target program being executed can be terminated only when the STP command is executed or a forced break (fail-safe break) is caused.

The default for the execution start addresses is the current PC value.

The table on the next page shows how each piece of hardware operates after the RUN N command is executed.

Caution: When a command to manipulate contents of a register, memory, or SFR is entered during trace (trc:0> mode) or emulation (emu:0> mode), execution of the emulation CPU is temporarily suspended.

Fig. 8-2 Operation of Hardware during Real-time Execution without Breaks

Hardware \ Event				Execution start RUN N ↓	Enable condition ENB ↓	Qualified trace condition TRX ↓	Trigger condition BRM ↓	Check point condition CHK ↓		Disable condition DSB ↓	Pass condition PAS ↓	Delay condition DLY ↓	Analyzer start TRG ↓	Analyzer stop STP T ↓	Emulation termination STP ↓
Emulation CPU		Operation	Stop	Real-time execution	→	→	→	Stop	Real-time execution	→	→	→	→	→	Stop
Emulation CPU		Interrupt	Held	Accepted	→	→	→	Held	Accepted	→	→	→	→	→	Held
Prompt			brk:0>	trc:0>	→	→	→	→	→	→	→	emu:0>	trc:0>	emu:0>	brk:0>
Event detector			Stop	Detected	→	→	→	Stop	Detected	→	Stop	→	Detected	Stop	→
Analyzer	Real-time tracer	All trace mode	Stop	Trace	→	→	→	Stop	Trace	→	→	Stop	Trace	Stop	→
Analyzer	Real-time tracer	Section trace mode	Stop	→	Trace	→	→	Stop	Trace	Stop	→	→	→	→	→
Analyzer	Real-time tracer	Qualified trace mode	Stop	→	→	Trace / Stop	→	→	→	→	→	→	→	→	→
Analyzer	Real-time tracer	Check point trace mode	Stop	→	→	→	→	Trace	Stop	→	→	→	→	→	→
Analyzer	Internal RAM data sampler	Sampler memory	Stop	Writing at sample timing	→	→	→	Stop	Writing at sample timing	→	→	Stop	Writing at sample timing	Stop	→
Analyzer	Performance evaluation	Execution time measurement	Stop	→	Measure	→	→	Stop	Measure	Stop	→	→	→	→	→
Analyzer	Performance evaluation	Executed instruction count	Stop	→	Measure	→	→	Stop	Measure	Stop	→	→	→	→	→
Analyzer	CO coverage		Stop	Measure	→	→	→	Stop	Measure	→	→	→	→	→	Stop
External trigger output			Low	→	→	→	→	→	→	→	High	Low	→	→	→

Remark: The arrows in the table indicate that the operation on the left continues.

Example 1: When only the analyzer is stopped by satisfying the delay conditions

```
brk:0>RUN N 200 <cr>  ← Starts executing a target program at 200H.
 User-system Vcc-ON     Emulation start at 0200
trc:0>
emu:0>■
```

Example 2: When the analyzer is stopped by an STP T command

```
brk:0>RUN N 200 <cr>
 User-system Vcc-ON     Emulation start at 0200
trc:0>STP T <cr>            ← Stops the analyzer.
emu:0>■
```

Example 3: When the analyzer and execution of the emulation CPU are stopped by an STP command

```
brk:0>RUN N 200 <cr>
 User-system Vcc-ON     Emulation start at 0200
trc:0>STP <cr>              ← Stops executing the emulation CPU.
 Escape break terminated
  PC    SP   PSW: UF  RBS2 RBS1 RBS0  S   Z  RSS  AC  IE  P/V  LT  CY
 0109  7000        0    0    1    1   0   1   0    0   0   1    0   1
  R0    R1   R2   R3   R4   R5   R6   R7    RP4    RP5   RP6    RP7
   X     A    C    B                         VP     UP    DE     HL
  30    10   20   30   20   20   60   70   1000   2000  3000   4000
brk:0>■
```

When the analyzer and execution of the emulation CPU stop with the STP command, the system displays the contents of the registers in the current register bank.

(2) RUN B command

RUN B [word]	
Radix	word:H

trc:0>	×	emu:0>	×	brk:0>	○

word: Execution start address

The RUN B command starts executing a target program from a specified address, and stops the emulation CPU, real-time tracer, and internal RAM data sampler when the delay conditions are satisfied.

The default for the execution start addresses is the current PC value.

To forcibly stop the analyzer or suspend execution of the emulation CPU, enter an STP command.

When the analyzer and emulation CPU are stopped by satisfying the delay conditions, the system automatically enters the one-step execution mode or procedure execution mode.

Pressing the return key in either of the above modes executes the next instruction.

When / <cr> is entered, the system enters the procedure execution mode.

To suspend the one-step execution or procedure execution, press the escape key.

The table on the next page shows how each piece of hardware operates after the RUN B command is executed.

Fig. 8-3 Operation of Hardware during Real-time Execution with Breaks

Hardware		Event		Execution start RUN B ↓	Enable condition ENB ↓	Qualified trace condition TRX ↓	Trigger condition BRM ↓	Check point condition CHK ↓	Disable condition DSB ↓	Pass condition PAS ↓	Delay condition DLY ↓	Re-execution <cr>or/<cr> ↓		Emulation terminations <ESC> ↓
Emulation CPU		Operation	Stop	Real-time execution	→	→	→	Stop	Real-time execution	→	→	Stop	Step execution Procedure execution	→	Stop
Emulation CPU		Interrupt	Held	Accepted	→	→	→	Held	Accepted	→	→	Held	→	→	→
Prompt			brk:0>	trc:0>	→	→	→	→	→	→	→	None	→	→	brk:0>
Event detector			Stop	Detected	→	→	→	Stop	Detected	→	Stop	→	→	→	→
Analyzer	Real-time tracer	All trace mode	Stop	Trace	→	→	→	Stop	Trace	→	→	Stop	All trace All trace for main routine	→	Stop
Analyzer	Real-time tracer	Section trace mode	Stop	→	Trace	→	→	Stop	Trace	Stop	→	→	All trace All trace for main routine	→	Stop
Analyzer	Real-time tracer	Qualified trace mode	Stop	→	→	Trace / Stop	→	→	→	→	→	→	All trace All trace for main routine	→	Stop
Analyzer	Real-time tracer	Check point trace mode	Stop	→	→	→	→	Trace	Stop	→	→	→	→	→	→
Analyzer	Internal RAM data sampler	Sampler memory	Stop	Writing at sample timing	→	→	→	Stop	Writing at sample timing	→	→	Stop	→	→	→
Analyzer	Performance evaluation	Execution time measurement	Stop	→	Measure	→	→	Stop	Measure	Stop	→	→	→	→	→
Analyzer	Performance evaluation	Executed instruction count	Stop	→	Measure	→	→	Stop	Measure	Stop	→	→	→	→	→
Analyzer	C0 coverage		Stop	Measure	→	→	→	Stop	Measure	→	→	→	Measure	→	Stop
External trigger output			Low	→	→	→	→	→	→	→	High	Low	→	→	→

Remarks 1. The arrows in the table indicate that the operation on the left continues.
2. <cr> indicates the point at which the return key is pressed.

Example: When the analyzer and execution of the emulation CPU are stopped by a trigger

```
brk:0>RUN B 100 <cr>               ← Starts executing a target program at 100H.
 User-system Vcc-ON    Emulation start at 0100
trc:0>
 Execution break terminated         ← The execution stopped by a trigger.
  PC     SP   PSW: UF     RBS2 RBS1 RBSO  S     Z    RSS   AC   IE   P/V   LT    CY
 0109   7000        0      0    1    1    0     1     0     0    0    1     0     1
  RO     R1    R2   R3     R4   R5   R6   R7          RP4    RP5     RP6       RP7
   X     A     C    B                                  VP     UP      DE        HL
  30     10    20   30     20   20   60   70          1000   2000    3000      4000
 One step emulation standby <cr>   ← Starts the one-step execution mode.
 Frame Status Address Data  Label  Mnemonic
 0022         010A                 MOV      R1,#3H
  PC     SP   PSW: UF     RBS2 RBS1 RBSO  S     Z    RSS   AC   IE   P/V   LT    CY
 010A   7000        0      0    1    1    0     1     0     0    0    1     0     1
  RO     R1    R2   R3     R4   R5   R6   R7          RP4    RP5     RP6       RP7
   X     A     C    B                                  VP     UP      DE        HL
  30     03    20   30     20   20   60   70          1000   2000    3000      4000
 One step emulation standby <ESC>  ← Terminates the one-step execution mode.
brk:0>■
```

(3) RUN T command

RUN T [word][、 { (*1) / step16 }][REG][TRD]					
Radix	word:H	step16:T	mask8:H	mask16:H	bit:Y

trc:0>	×	emu:0>	×	brk:0>	○

word: Execution start address
REG: Register display specification
TRD: Trace display specification
step16: Number of 16-bit steps
mask8: 8-bit mask data
mask16: 16-bit mask data
register name: PC, SP, RO, R1, R2, R3, R4, R5, R6, R7, RP4, RP5, RP6, RP7, X, A, B, C, VP, UP, DE, HL
PSW flag name: UF, RBS2, RBS1, RBSO, S, Z, RSS, AC, IE, P/V, LT, CY
bit: Single bit numeric value

*1 To be substituted

```
                              (*2)
┌register    ┌  =  ┐    ┌mask8  ┐   ┐
│name        │  >  │   ─┤       ├─  │
┤            ┤  <  ├    └mask16 ┘   ├
│            │  => │                │
│            │  >= │                │
│            │  =< │                │
│            │  <= │                │
│            │  >< │                │
│            └  <> ┘                │
└PSW flag name = bit                ┘
```

*2 When mask data are used for mask8 and mask16, only =, <>, and >< are valid.

The RUN T command executes each instruction of a target program sequentially from a specified address, and traces the execution results.

The default for the execution start addresses is the current PC value.

When REG is specified, the contents of the current register are displayed for each instruction.

When TRD is specified, the result of executing each instruction is disassembled and displayed.

Step execution continues intermittently until the register conditions are satisfied and the specified number of instructions are executed.

When step execution terminates, the system enters the one-step execution mode or procedure execution mode.

Pressing the return key in either of the above modes executes the next instruction.

When / <cr> is entered, the system enters the procedure execution mode.

To suspend the one-step execution or procedure execution, press the escape key.

The table on the next page shows how each piece of hardware operates after the RUN T command is executed.

Caution: All interrupts during the step execution are held.

Fig. 8-4 Operation of Hardware during Step Execution

Hardware			Event	Command input RUN T ↓	Re-execution <cr>or/<cr> ↓		Emulation termination <ESC> ↓
Emulation CPU	Operation		Stop	Step execution	Step execution Procedure execution	→	Stop
	Interrupt		Held	→	→	→	→
Prompt			brk:0>	None	→	→	brk:0>
Event detector			Stop	→	→	→	→
Analyzer	Real-time tracer	All-trace mode	Stop	Trace	All trace Main routine trace	→	Stop
		Section trace mode	Stop	Trace	All trace Main routine trace	→	Stop
		Qualified trace mode	Stop	Trace	All trace Main routine trace	→	Stop
		Check point trace mode	Stop	→	→	→	→
	Internal RAM data sampler	Sampler memory	Stop	→	→	→	→
	Performance evaluation	Execution time measurement	Stop	→	→	→	→
		Executed instruction count	Stop	→	→	→	→
	C0 coverage		Stop	Measure	→	→	Stop
External trigger output			Low	→	→	→	→

Remark: The arrows in the table indicate that the operation on the left continues.

Example: Step execution

```
brk:0>RUN T 100,R1=1 TRD <cr>       <- Starts executing a target program at 100H.
                                       Stops execution with R1=1.
 User-system Vcc-ON    Emulation start at 0100
 Frame Status Address Data             Label  Mnemonic
 0000          0100                           MOVW     SP,#0FE00H
 0006          0104                           MOV      R0,#0H
 0008          0106                           MOV      R1,#1H
  terminated
  PC     SP   PSW: UF    RBS2 RBS1 RBS0  S    Z    RSS   AC    IE  P/V  LT   CY
 0108   FE00        0     0    1    1    0    1     0     0     0   1    0    1
  R0     R1    R2   R3    R4   R5   R6   R7         RP4      RP5     RP6     RP7
   X      A     C    B                               VP       UP      DE      HL
  03     01    20   30    20   20   60   70        1000     2000    3000    4000
 One step emulation standby <cr>    <- Starts the one-step execution mode.
 Frame Status Address Data             Label  Mnemonic                  EX
 0000          0108                           MOV      R2,#2H           00
  PC     SP   PSW: UF    RBS2 RBS1 RBS0  S    Z    RSS   AC   IE  P/V  LT   CY
 0108   FE00        0     0    1    1    0    1     0     0    0   1    0    1
  R0     R1    R2   R3    R4   R5   R6   R7        RP4      RP5     RP6     RP7
   X      A     C    B                              VP       UP      DE      HL
  03     01    02   30    20   20   60   70       1000     2000    3000    4000
 One step emulation standby <ESC>   <- Terminates the one-step execution mode.
brk:0>■
```

(4) RUN T command with PRC

RUN T [word][, { (*1) / step16 }][REG][TRD] PRC					
Radix	word:H	step16:T	mask8:H	mask16:H	bit:Y

trc:0>	×	emu:0>	×	brk:0>	○

word:	Execution start address
REG:	Register display specification
TRD:	Trace display specification
step16:	Number of 16-bit steps
mask8:	8-bit mask data
mask16:	16-bit mask data
register name:	PC, SP, R0, R1, R2, R3, R4, R5, R6, R7, RP4, RP5, RP6, RP7, X, A, B, C, VP, UP, DE, HL

PSW flag name: UF, RBS2, RBS1, RBS0, S, Z, RSS, AC, IE, P/V, LT, CY

PRC: Procedure execution specification

bit: Single bit numeric value

*1 To be substituted

```
                                  (*2)
 ┌register    ┌ =  ┐      ┌mask8  ┐  ┐
 │name        │ >  │     ─┤       ├─ │
─┤           ─┤ <  ├─     └mask16 ┘  ├─
 │            │ => │                 │
 │            │ >= │                 │
 │            │ =< │                 │
 │            │ <= │                 │
 │            │ >< │                 │
 │            └ <> ┘                 │
 └PSW flag name = bit                ┘
```

*2 When mask data are used for mask8 and mask16, only =, <>, and >< are valid.

The RUN T command with PRC executes each instruction of a target program sequentially from a specified address, and traces the results of execution. For a routine whose nesting level is deeper than that of the routine from which the procedure execution started (current routine), such as a routine or context routine enclosed by the CALL and RET instructions, the target program is executed step by step, but tracing is not performed.

The default for the execution start addresses is the current PC value.

When REG is specified, the contents of the current register are displayed for each instruction.

When TRD is specified, the result of executing each instruction is disassembled and displayed.

Step execution continues intermittently until the register conditions are satisfied and a specified number of instructions are executed.

When the procedure execution terminates, the system enters the one-step execution mode or procedure execution mode.

Pressing the return key in either of the above modes executes the next instruction.

When / <cr> is entered, the system enters the procedure execution mode.

To suspend the one-step execution or procedure execution, press the escape key.

The table on the next page shows how each piece of hardware operates after the RUN T command with PRC is executed.

Caution: All interrupts during the procedure execution are held.

Fig. 8-5 Operation of Hardware during Procedure Execution

Hardware \ Event				Command input RUN T ↓	Re-execution <cr>or/<cr> ↓		Emulation termination <ESC> ↓
Emulation CPU		Operation	Stop	Procedure execution	Step execution Procedure execution	→	Stop
Emulation CPU		Interrupt	Held	→	→	→	→
Prompt			brk:0>	None	→	→	brk:0>
Event detector			Stop	→	→	→	→
Analyzer	Real-time tracer	All-trace mode	Stop	All trace of main routine	All trace Main routine trace	→	Stop
Analyzer	Real-time tracer	Section trace mode	Stop	All trace of main routine	All trace Main routine trace	→	Stop
Analyzer	Real-time tracer	Qualified trace mode	Stop	All trace of main routine	All trace Main routine trace	→	Stop
Analyzer	Real-time tracer	Check point trace mode	Stop	→	→	→	→
Analyzer	Internal RAM data sampler	Sampler memory	Stop	→	→	→	→
Analyzer	Performance evaluation	Execution time measurement	Stop	→	→	→	→
Analyzer	Performance evaluation	Executed instruction count	Stop	→	→	→	→
Analyzer	C0 coverage		Stop	Measure	→	→	Stop
External trigger output			Low	→	→	→	→

Remark: The arrows in the table indicate that the operation on the left continues.

Example: Procedure execution

```
brk:0>RUN T 100,R0=3H TRD PRC <cr>(Note)
 User-system Vcc-ON      Emulation start at 100
 Frame Status Address Data  Label  Mnemonic
 0000          0100                MOVW     SP,#0FE00H
 0006          0104                MOV      R1,#0H
 0010          0106                CALL     !NEST1
 0015   WR     FDFE     01
 0016   WR     FDFF     09
  terminated
  PC    SP  PSW: UF    RBS2 RBS1 RBS0  S    Z    RSS  AC   IE  P/V  LT   CY
 0109  FE00       0     0    0    0    0    0    0    0    0    0    0    0
   R0    R1    R2    R3    R4    R5    R6    R7      RP4      RP5     RP6      RP7
    X     A     C     B                              VP       UP      DR       HL
   03    00    00    00    00    00    00    00     0000     0000    0000     0000
 One step emulation standby <cr>   ← Starts the one-step execution mode.
 Frame Status Address Data  Label  Mnemonic
 0000          0109                MOV      R3,#3H
  PC    SP  PSW: UF    RBS2 RBS1 RBS0  S    Z    RSS  AC   IE  P/V  LT   CY
 0109  FE00       0     0    0    0    0    0    0    0    0    0    0    0
   R0    R1    R2    R3    R4    R5    R6    R7      RP4      RP5     RP6      RP7
    X     A     C     B                              VP       UP      DE       HL
   03    00    00    00    00    00    00    00     0000     0000    0000     0000
 One step emulation standby <ESC>  ← Terminates the one-step execution mode.
brk:0>■
```

Note: Execution of a target program starts at 100H and the execution stops with R0=3.

8.38 Saving Object and Debugging Environment (SAV)

SAV	file[partition][C][D][$V]
Radix	partition:H

trc:0>	×	emu:0>	×	brk:0>	○

file:	File name
partition:	Save address range
C:	Object specification
D:	Debugging environment specification
$V:	Verify specification

The SAV command saves the object code and debugging environment to a file specified by the host machine.

Up to five save address ranges, which must be delimited by a space, can be specified.

When no save address range is specified, the system saves all the contents of mapped memory and the addresses 0FE00H to 0FE7FH of the internal RAM.

When C is specified, the object code is saved. When D is specified, the debugging environment is saved. When neither C nor D is specified, the object code is saved first, and then the debugging environment is saved.

When the object code and debugging environment are to be saved, no extension can be specified for a file name. In this case, an extension, HEX (object file) or DBG (debugging environment file), is automatically added.

When $V is specified at the end of a command line, the contents of the file and memory are verified after the object code is saved.

The following debugging environmental elements are saved.

- o Internal ROM size
- o Internal RAM size
- o Setting of the uPD78330 + TAM emulation
- o Download mode
- o Settings of the following commands concerning the debugging environment

BRA	BRD	BRS	BRM	CHK	ENB	DSB	CLK
CVM	PAS	DLY	MAP	MOD	PGM	PSA	PST
TRM	TRF	TRS	TRX	WRD			

Example 1: To save the object and debugging environment

```
brk:0>SAV B:SAMPLE <cr>
object save complete           ] <- Message output when processing
debug condition save complete  ]    terminated normally
brk:0>■
```

Example 2: To specify save address ranges and save the object code with the verify function

```
brk:0>SAV B:SAMPLE.HEX 0XXX 2000,27FF C $V <cr>(Note)
object save complete           ] <- Message output when processing
object verify complete         ]    terminated normally
brk:0>■
```

Note: Addresses from 0 to 0FFF and 2000 to 27FF are saved with the verify function.

Example 3: To save the debugging environment

```
brk:0>SAV B:SAMPLE.DBG D <cr>
debug condition save complete  <- Message output when processing
brk:0>■                           terminated normally
```

Example 4: When a file named SAMPLE.HEX already exits in drive B

```
brk:0>SAV B:SAMPLE.HEX C <cr>
 File already exists.  (513)(Note) B:SAMPLE.HEX Delete ? (Y or N):Y <cr>
brk:0>■
```

The above message is output when a file named SAMPLE.HEX with the read/write attribute already exists.

When Y <cr> is entered, the existing file (SAMPLE.HEX) is deleted and a new SAMPLE.HEX file is created.

Otherwise a new file is not created.

Example 5: When a file named SAMPLE.HEX in drive B cannot be opened

```
brk:0>SAV B:SAMPLE.HEX C <cr>  ← The name of a file whose attribute is
                                  R/O is specified.
     Read only file.  (507)(Note)  B:SAMPLE.HEX
    brk:0>■
```

The system displays the above message when a file named SAMPLE.HEX whose attribute is R/O (read-only) already exists in drive B, then rejects the command.

Note: The numbers in parentheses indicate the error message Nos.

8.39 Manipulating SFRs (SFR)

```
         ┌ C ┐
SFR [ ───┤   ├─[SFR name]]
         └ D ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

SFR C: SFR change

SFR D: SFR display

SFR name: P0,P1,(P2),P3,P4,P5,(P7),(P8),P9,TLA,(TM2),
(CT00),(CT01),(CT02),(CT10),
<PM0>,<PM1>,<PM3>,<PM5>,<PM9>,(TM0),(TM1),(TM3),
<PMC0>,<PMC1>,<PMC3>,BRG,RTP,RTPR,PRDC,RTPS,
PWMC,PWM0,ADM,PWM1,CM11,CM12,CM20,CM21,CM30,
CSIM,SBIC,SIO,ASIM,(ASIS),(RXB),<TXS>,CMX0,
CM01R,CM02R,CM03R,CM04R,CC00R,CC01R,(ADCR0),
(ADCR1),(ADCR2),(ADCR3),(ADCR4),(ADCR5),(ADCR6),
(ADCR7),TMC0,BRGM,TMC1,TUM0,TUM1,TOC0,TOC1,PPOS,
STBC,CCW,WDM,MM,PWC,FCC,IF0,IF1,MK0,MK1,PB0,
PB1,ISM0,ISM1,CSE0,CSE1,INTM0,INTM1,(ISPR),PRSL
EXTSFR0,EXTSFR1,EXTSFR2,EXTSFR3,EXTSFR4,EXTSFR5,
EXTSFR6,EXTSFR7,EXTSFR8,EXTSFR9,EXTSFR10,
EXTSFR11,EXTSFR12,EXTSFR13,EXTSFR14,EXTSFR15

The SFR command changes or displays SFRs.

When SFR <cr> is entered, SFRs are displayed.

Cautions 1. SFRs in parentheses are used for reading only and SFRs in angle brackets are used for writing only.

2. When SFRs are to be manipulated in the emu:0> prompt mode, reading or writing of SFRs is stopped temporarily.

(1) SFR C command

SFR C [SFR name]

trc:0>	×	emu:0>	○	brk:0>	○

SFR name: P0,P1,(P2),P3,P4,P5,(P7),(P8),P9,TLA,(TM2), (CT00),(CT01),(CT02),(CT10),<PM0>,<PM1>, <PM3>,<PM5>,<PM9>,(TM0),(TM1),(TM3),<PMC0>, <PMC1>,<PMC3>,BRG,RTP,RTPR,PRDC,RTPS,PWMC, PWM0,ADM,PWM1,CM11,CM12,CM20,CM21,CM30, CSIM,SBIC,SIO,ASIM,(ASIS),(RXB),<TXS>,CMX0, CM01R,CM02R,CM03R,CM04R,CC00R,CC01R, (ADCR0),(ADCR1),(ADCR2),(ADCR3),(ADCR4), (ADCR5),(ADCR6),(ADCR7),TMC0,BRGM,TMC1, TUM0,TUM1,TOC0,TOC1,PPOS,STBC,CCW,WDM,MM, PWC,FCC,IF0,IF1,MK0,MK1,PB0,PB1,ISM0,ISM1, CSE0,CSE1,INTM0,INTM1,(ISPR),PRSL EXTSFR0,EXTSFR1,EXTSFR2,EXTSFR3,EXTSFR4, EXTSFR5,EXTSFR6,EXTSFR7,EXTSFR8,EXTSFR9, EXTSFR10,EXTSFR11,EXTSFR12,EXTSFR13, EXTSFR14,EXTSFR15

The SFR C command changes SFRs.

When SFR <cr> is entered, registers P0 to PRSL are changed in the ascending order of mapping addresses.

When an SFR name is specified, that SFR and later SFRs are changed sequentially.

When an SFR is not to be changed, press the return key only.

To stop changing SFRs halfway, press the escape key.

Caution: SFRs in parentheses are used for reading only and SFRs in angle brackets are used for writing only.

Example 1: To change the contents of all the SFRs sequentially

```
brk:0>SFR C <cr>               ← Changes all writable SFRs.
 P0          77 = 88 <cr>
 P1          88 = <cr>         ← No change
          .
          .
 PRSL        00 = 02 <cr>
brk:0>■
```

Example 2: To change SFRs sequentially from a specified SFR

```
brk:0>SFR C ASIM <cr>          ← Changes SFRs from register ASIM.
 ASIM        03 = 04 <cr>
 TXS         -- = 0AA <cr>     ← -- is displayed for a write-only
          .                      register.
          .
 TMC0        20 = <ESC>        ← Terminates the change processing.
brk:0>■
```

Example 3: When a read-only SFR is specified

```
brk:0>SFR C ASIS <cr>          ← A read-only SFR is specified.
 Read only                     ← A message indicating that the SFR
brk:0>■                          is a read-only one
```

(2) SFR D command

SFR D [SFR name]

trc:0>	×	emu:0>	○	brk:0>	○

SFR name: P0,P1,(P2),P3,P4,P5,(P7),(P8),P9,TLA,(TM2),(CT00),(CT01),(CT02),(CT10),<PM0>,<PM1>,<PM3>,<PM5>,<PM9>,(TM0),(TM1),(TM3),<PMC0>,<PMC1>,<PMC3>,BRG,RTP,RTPR,PRDC,RTPS,PWMC,PWM0,ADM,PWM1,CM11,CM12,CM20,CM21,CM30,CSIM,SBIC,SIO,ASIM,(ASIS),(RXB),<TXS>,CMX0,CM01R,CM02R,CM03R,CM04R,CC00R,CC01R,(ADCR0),(ADCR1),(ADCR2),(ADCR3),(ADCR4),(ADCR5),(ADCR6),(ADCR7),TMC0,BRGM,TMC1,TUM0,TUM1,TOC0,TOC1,PPOS,STBC,CCW,WDM,MM,PWC,FCC,IF0,IF1,MK0,MK1,PB0,PB1,ISM0,ISM1,CSE0,CSE1,INTM0,INTM1,(ISPR),PRSL
EXTSFR0,EXTSFR1,EXTSFR2,EXTSFR3,EXTSFR4,EXTSFR5,EXTSFR6,EXTSFR7,EXTSFR8,EXTSFR9,EXTSFR10,EXTSFR11,EXTSFR12,EXTSFR13,EXTSFR14,EXTSFR15

The SFR D command displays the contents of readable SFRs.

When an SFR name is specified, the specified SFR is displayed. If a specified SFR is a write-only one, -- is displayed.

When SFR D <cr> is entered, registers P0 to PRSL are displayed in the ascending order of addresses.

When the system cannot display all SFRs on a screen at a time, the system displays the message "Next ? (Y/N)" and waits for entry. When Y <cr> is entered, the next screen is displayed. Otherwise the system terminates the command.

Caution: SFRs in parentheses are used for reading only and SFRs in angle brackets are used for writing only.

Example 1: To display the contents of all the SFRs

```
brk:0>SFR D <cr>        ← Displays all readable SFRs.
 P0     P1     P2     P3     P4     P5     P7     P8     P9     TLA    TM2
 00     00     00     00     00     00     00     00     00     00     0000
                        .
                        .
 ISM1   CSE0   CSE1   INTM0  INTM1  PRSL
 2828   9876   5432   10     76     32

 Next ? (Y/N) Y <cr>  ← Displays the next screen.

 EXTSFR0  EXTSFR1  EXTSFR2  EXTSFR3  EXTSFR4  EXTSFR5  EXTSFR6  EXTSFR7
 00       00       00       00       00       00       00       00

 EXTSFR8  EXTSFR9  EXTSFR10 EXTSFR11 EXTSFR12 EXTSFR13 EXTSFR14 EXTSFR15
 00       00       00       00       00       00       00       00
brk:0>■
```

Example 2: When an SFR is specified

```
brk:0>SFR D P5 <cr>       ← Displays register P5.
 P5         01
brk:0>■

brk:0>SFR D TxS <cr>      ← A write-only SFR is specified.
 TxS        --            ← -- is displayed for a write-only SFR.
brk:0>■
```

8.40 Execution Stop (STP)

STP

trc:0>	O	emu:0>	O	brk:0>	×

STP T

trc:0>	O	emu:0>	×	brk:0>	×

T: Stopping the analyzer only

The STP command has the following functions:

. Stopping the emulation CPU and analyzer: STP <cr>
. Stopping the analyzer only: STP T <cr>

8.41 Command Input from a File (STR)

STR file[parameter]

trc:0>	O	emu:0>	O	brk:0>	O

file: File name
parameter: Real parameter

The STR command inputs commands or data from a specified file.

Specify a file which is created with a COM command or editor as the input file.

To stop input of data or commands from a file, enter ^L.

To restart input, enter ^L again.

To terminate input of data or commands, enter ^K.

Formal parameters in an input file(Note) can be replaced by real parameters. Up to four real parameters can be specified.

Note: Use an editor to create a file containing formal parameters.

Use $0, $1, $2, and $3 to describe formal parameters in an input file.

Use $$ for relative addresses of the assembler to distinguish such addresses from formal parameters.

Example 1: To input commands from a file

```
brk:0>STR B:SAMPLE.STR <cr>   ← A SAMPLE.STR file in drive B is
                                specified as the input file.
brk:0>CLK I                   ← First command input
brk:0>RES
brk:0>OUT OFF
brk:0>■
```

. The following shows the contents of the file in Example 1.

```
CLK I
RES
OUT OFF
```

Example 2: To input commands from a file containing formal parameters

```
brk:0>STR SAMPLE.STR U ON <cr>   ← Real parameters U and ON are specified.
brk:0>CLK U                      ← First command input
brk:0>RES
brk:0>OUT ON
brk:0>■
```

. The following shows the contents of the file in Example 2.

```
CLK $0
RES
OUT $1
```

Example 3: When a specified file is missing

```
brk:0>STR B:SAMPLE.STR <cr>
     File not found.  (509)(Note)  B:SAMPLE.STR  ← Message output when
brk:0>■                                             the specified file
                                                    (SAMPLE.STR) is
                                                    missing
```

Note: The number in parentheses indicates the error message Nos.

8.42 Manipulating Symbols (SYM)

```
SYM  ┌ A ┐
    ─┤   ├─symbol word
     └ C ┘

SYM  D [module name¥]

SYM  E [symbol]

     ┌ K ┐
SYM ─┤ L ├─
     │ S │
     └ M ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

SYM A:	Registering IESYMBOL
SYM C:	Changing IESYMBOL
SYM D:	Displaying IESYMBOL or preload symbols
SYM E:	Deleting IESYMBOL
SYM K:	Deleting IESYMBOL or preload symbols
SYM L:	Loading IESYMBOL
SYM S:	Saving IESYMBOL
SYM M:	Current module specification
symbol:	Symbol name
word:	Symbol value
module name¥:	Module name

(1) SYM A command

SYM A symbol word	
Radix	word:H

trc:0>	×	emu:0>	○	brk:0>	○

symbol: Symbol name
word: Symbol value

The SYM A command registers an IE symbol.

The name of a symbol to be registered must be different from the name of an existing IE symbol and reserved words.

IESYMBOL (module name) is assigned to a symbol registered with an SYM A command.

The symbol is of a code type.

Example: Registering an IE symbol

```
brk:0>SYM A SYMBOL01 1000 <cr>   ← Registers SYMBOL01 whose symbol
brk:0>■                            value is 1000H.
```

(2) SYM C command

SYM C symbol word	
Radix	word:H

trc:0>	×	emu:0>	○	brk:0>	○

symbol: Symbol name
word: Symbol value

The SYM C command changes the symbol value of a specified IE symbol.

Example: Changing the symbol value of an IE symbol

brk:0>SYM C SYMBOL01 2000 <cr> ← Changes the symbol value of the IE symbol named SYMBOL01 to 2000H.
brk:0>■

(3) SYM D command

SYM D [module name¥]

trc:0>	×	emu:0>	○	brk:0>	○

module name¥(Note 1): Module name

The SYM D command displays registered IE symbols or preload symbols.

When a module name is specified, all the symbols in the specified module are displayed.

When PUBLIC¥ is specified, all the symbols in the IESYMBOL and PUBLIC modules are displayed(Note 1).

When SYM D <cr> is entered, all registered IE symbols or preload symbols are displayed for each module.

When no symbol is registered, the following message is displayed.

No symbol. (404)(Note 2)

When a specified module is missing, the following message is displayed.

Module not found. (519)(Note 2)

Notes 1. When the host machine is an IBM PC series, ¥ is replaced by \.

2. The numbers in parentheses indicate the error message Nos.

Example 1: To display all registered IE symbols or preload symbols

```
brk:0>SYM D <cr>
 module : IESYMBOL
    1000 INT1          2000 INT2
 module : PUBLIC
    0100 START         0600 DATA       01FF FINISH
 module : SUBPRG1
    0250 START         0280 FINISH
brk:0>■
```

Example 2: To display public symbols only

```
brk:0>SYM D PUBLIC¥ <cr>
 module : IESYMBOL
    1000 INT1          2000 INT2
 module : PUBLIC
    0100 START         0600 DATA       01FF FINISH
brk:0>■
```

Example 3: To display symbols in a specified module

```
brk:0>SYM D SUBPRG1¥ <cr>
 module : SUBPRG1
    0250 START          0280 FINISH
brk:0>■
```

(4) SYM E command

SYM E symbol

trc:0>	×	emu:0>	○	brk:0>	○

symbol: Symbol name

The SYM E command deletes a specified IE symbol.

When no symbol is specified, all symbols defined with an SYM A command are deleted.

Example 1: To delete a specified IE symbol

```
brk:0>SYM E INT1 <cr>   ← Deletes INT1.
brk:0>■
```

Example 2: To delete all IE symbols

```
brk:0>SYM E  <cr>
brk:0>■
```

(5) SYM K command

SYM K

trc:0>	×	emu:0>	○	brk:0>	○

The SYM K command deletes all registered IE symbols or preload symbols.

Example: Deleting all symbols

```
brk:0>SYM K <cr>
brk:0>■
```

(6) SYM L command

SYM L

trc:0>	×	emu:0>	○	brk:0>	○

The SYM L command loads IESYMBOL from an IE symbol file. An IE symbol file to be loaded resides in the current drive and the name of the file is IE78330.SYM.

Example 1: Loading an IE symbol file

```
brk:0>SYM L <cr>
brk:0>■
```

Example 2: When no IE symbol file is found

```
brk:0>SYM L <cr>
 IE78330.SYM file not found.  (400)(Note)   ← Message output when
brk:0>■                                        no IE symbol file is
                                               found for emulation
                                               of a uPD78330 series
```

Note: The number in parentheses indicates the error message No.

(7) SYM S command

SYM S

trc:0>	×	emu:0>	○	brk:0>	○

The SYM S command saves IESYMBOL into an IE symbol file.

The system automatically assigns IE78330.SYM to the name of an IE symbol file.

The system selects the current directory on the current drive for the drive unit.

Example 1: Saving IESYMBOL to an IE symbol file

```
brk:0>SYM S <cr>
brk:0>■
```

Example 2: When IESYMBOL is missing

```
brk:0>SYM S <cr>
 No symbol.  (404)(Note)      ← Message output when no IESYMBOL is
brk:0>■                          found
```

Example 3: When a specified IE symbol file already exists

```
brk:0>SYM S <cr>
 File already exists.  (513)(Note)   IE78330.SYM Delete ? (Y/N) : Y <cr>
brk:0>■
```

When an IE78330.SYM file with the read/write attribute already exists, the above message is output. When Y <cr> is entered, the existing file (IE78330.SYM) is deleted and a new IE78330.SYM file is created.

Otherwise the command is rejected.

Example 4: When an IE symbol file cannot be created

```
brk:0>SYM S <cr>
 Read only file.  (507)(Note)  IE78330.SYM
brk:0>■
```

The system displays the above message when the attribute of the file is R/O (read-only), and then rejects the command.

Note: The number in parentheses indicates the error message No.

(8) SYM M command

SYM M

trc:0>	×	emu:0>	○	brk:0>	○

The SYM M command displays or changes the current module name.

When the current module is set, symbols in the current module can be described without specifying the module name.

When a symbol in the current module has the same name as the symbol in the public symbol module, the symbol in the current module is treated as the public symbol (see Figure 8-6).

To describe a local symbol belonging to another module, the module name must be specified before the local symbol name.

Example 1: To display the current module name

```
brk:0>SYM M <cr>
 MOD01 ¥ = ■          ← The current module name is displayed.

brk:0>SYM M <cr>
 MOD01 ¥ = <cr>       ← Enter <cr> not to change the current module
brk:0>■                  name.
```

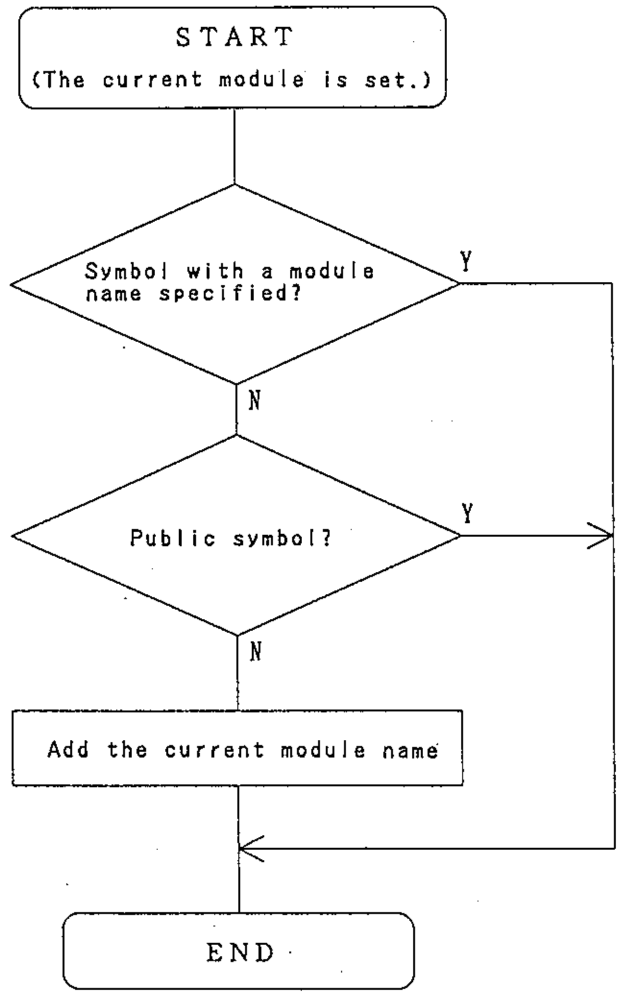
Example 2: To change the current module name

```
brk:0>SYM M <cr>
 MOD01 ¥ = MOD02¥ <cr>  ← Change the current module name to MOD02.
brk:0>■

brk:0>SYM M <cr>
 MOD02 ¥ = <cr>         ← The current module name is MOD02.
brk:0>■
```

Remark: When the host machine is an IBM PC series, ¥ is replaced by \.

Fig. 8-6 Processing Flow of Symbols when the Current Module is Set



8.43 Displaying Trace Data (TRD)

```
              ┌ F ┐          ┌ $F ┐
TRD          -┤   ├-[ALL][-  │ $J │  -]
              └ I ┘          │ $Q │
                             └ $C ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

TRD F: Frame display

TRD I: Instruction display

ALL: Displays all trace data or all data that satisfy the retrieval conditions from the first frame of the current trace block.

$F: Displays the frame which satisfies the retrieval conditions together with ten frames (five immediately before the frame and five immediately after the frame).

$J: Displays the frames concerning branches of the program.

$Q: Displays the frames which satisfy the retrieval conditions.

$C: Displays the frames at a check point.

The TRD command displays traced data in the current trace block, or retrieves and displays the current trace block.

Trace data can be displayed in eight formats according to the combination of a subcommand and operands.

The following four terms concerning the TRD command are explained.

- o Trace block
- o Trace pointer setting
- o Trace data retrieval display
- o Marking information when trace data is displayed

(a) Trace block

Whenever data are written into trace memory, the data are added to trace data. When the RES H command is entered, however, trace memory is cleared.

Data traced by the emulation CPU by executing a target program is called a trace block. In this case, the target program is executed after an emulation start command (RUN B, RUN N, RUN T, or TRG) is input and before the next emulation start command is input.

The trace data display command (TRD) displays trace data in the current trace block, or retrieves the current trace block and displays data satisfying the retrieval conditions.

(b) Trace pointer setting

For trace data display or trace data retrieval display, the trace data that are displayed first are determined according to the position of the current trace pointer or a specified trace pointer.

When emulation terminates, the trace pointer is placed at the last frame of the current trace block.

To change the trace display area, specify the trace pointer as necessary in the prompting mode after trace data are displayed, or enter ALL in the TRD command.

Table 8-2 shows the relationship between the trace pointer specification and the frames to be displayed.

Table 8-2 Relationship between the Trace Pointer Specification and Frames to be Displayed

<table>
<tr><th colspan="2">Trace pointer specification</th><th>Prompting command</th><th>Function</th><th>Inside/outside of the current trace block</th></tr>
<tr><td colspan="2">Absolute address</td><td>frame No.</td><td>Displays a specified frame and 10 lines following the frame.</td><td>Inside/outside</td></tr>
<tr><td rowspan="2">Current trace block change</td><td>Previous</td><td>P</td><td>Displays the last 11 lines of the previous trace block.</td><td rowspan="2">Outside</td></tr>
<tr><td>Next</td><td>N</td><td>Displays the first 11 lines of the next trace block.</td></tr>
<tr><td colspan="2" rowspan="2">+/cr/- movement</td><td>+/cr</td><td>Displays 11 lines following a frame that has just been displayed.</td><td rowspan="5">Internal</td></tr>
<tr><td>-</td><td>Displays 11 lines preceding a frame that has just been displayed.</td></tr>
<tr><td colspan="2" rowspan="2">Operation in the current trace block</td><td>F</td><td>Displays the first 11 lines.</td></tr>
<tr><td>L</td><td>Displays the last 11 lines.</td></tr>
<tr><td colspan="2">Trigger retrieval</td><td>T</td><td>Displays the trigger frame together with the preceding five lines and the following five lines.</td></tr>
</table>

To change the trace block to be displayed, enter the trace block change command (P or N), or set the trace pointer by entering the absolute address command.

(c) Trace data retrieval display

A function to retrieve trace frames of the current trace block and display the frames satisfying the retrieval conditions is called the trace data retrieval display.

To specify the retrieval conditions, enter one of the $ operands in the TRD command, or enter the trigger retrieval command in the prompting mode after trace data are displayed.

The TRD command operands and their functions are listed below.

Table 8-3 TRD Command Operands and Their Functions

Operand	Function
$F	Displays the frame that satisfies the retrieval conditions together with the preceding five lines and the following five lines.
$J	Displays a frame concerning program branches.
$Q	Displays a frame that satisfies the retrieval conditions.
$C	Displays a frame at the check point.

The retrieval conditions required for the $F or $Q operand must be preset by the retrieval condition setting command (TRF).

(d) Trace data marking information

When trace data is displayed, the following data items are automatically appended:

- Operation status of the emulation CPU when the trace frame was written into trace memory (Execution mode)
- Trace mode
- Marking information to identify special frames.

The emulation CPU execution modes and marking information for the trace mode are listed below.

Table 8-4 Execution Modes and Marking Information for the Trace Mode

Execution mode	Trace mode	Marking information
Real-time emulation	ALL	[ALL trace mode terminated.]
	SEC	[SEC trace mode terminated.]
	TRX	[TRX trace mode terminated.]
Single-step execution	(All trace of execution steps)	[one step emulation terminated.]
Procedure execution	(All trace only for the main routine)	[procedure emulation terminated.]

Marking information for the single-step execution and procedure execution are displayed only when the execution mode is changed.

Marking information for special frames is shown below.

Table 8-5 Marking Information for Special Frames

Emulation CPU/tracer status	Marking status
Occurrence of interrupt vector	<INTxx>
Trigger frame	T
Check routine	<CHK> Written data

Remark: In the section trace mode or qualified trace mode, the trigger point occurs between the trigger display frame and the immediately preceding frame.

Trace data of the real-time tracer, trace blocks, and marking information are listed below.

Table 8-6 Real-time Tracer Operations

Command	Trace mode (TRM)	Time tag selection	Trace data of real-time tracer	Trace block	Marking information
RUN_B <cr>	All trace ↑ ↑ ↑ ↑ ↑ ↑ ↑	EXT ↑ ↑ ↑ ↑ ↑ ↑ ↑	T D	①	-[All trace mode terminated.] -[one step emulation terminated.]
RUN_B <cr>	Qualified trace ↑ ↑ ↑	EXT ↑ ↑ ↑	T	②	-[TRX trace mode terminated.] -[one step emulation terminated.]
RUN_N	Section trace ↑ ↑ ↑	TIME ↑ ↑ ↑	T D	③	-[SEC trace mode terminated.]
TRG	Section trace ↑ ↑ ↑ ↑	TIME ↑ ↑ ↑ ↑	T D	④	-[SEC trace mode terminated.]
RUN_T_PRC <cr> <cr> <cr>	Procedure trace ↑ ↑ ↑ ↑ ↑	TIME ↑ ↑ ↑ ↑ ↑		⑤	-[procedure emulation terminated.] -[one step emulation terminated.]

. T: Point at which the pass conditions are satisfied.
. D: Point at which the delay conditions are satisfied.

(1) TRD F command

```
                  ┌$F┐
TRD  F  [ALL] [ ─┤$J├─ ]
                  │$Q│
                  └$C┘
```

trc:0>	×	emu:0>	○	brk:0>	○

ALL: Displays all trace data or all data that satisfy the retrieval conditions from the first frame of the current trace block.

$F: Displays the frame which satisfies the retrieval conditions together with ten frames (five immediately before the frame and five immediately after the frame).

$J: Displays the frames concerning branches of the program.

$Q: Displays the frames which satisfy the retrieval conditions.

$C: Displays the frames at a check point.

The TRD F command displays trace frames of the current trace block in the frame mode, or retrieves and displays trace data of the current trace block in the frame mode.

A trace frame line consists of the following data items:

Table 8-7 Data Items of a Trace Frame Line (TRD F)

Label	Description
Frame	The frame No. of trace memory, which is assigned sequentially from the first frame that is written into trace memory. The frame Nos. which can be displayed are values from 0000 to 8191.
Status	The bus cycle status of a trace frame . OP: Op-code fetch . RD: Read access . WR: Write access . INTRD: Read cycle by interrupt . INTWR: Write cycle by interrupt . MSRD: Read cycle by macro service . MSWR: Write cycle by macro service . BROP: First fetch after branch . WST(Note): Write activation cycle of BCU bus (Internal write cycle)
Address	Fetch or read/write address
Data	Fetch or read/write data. Data enclosed in parentheses are invalid fetch data.
Exu	Exu operation information of the emulation CPU . M1: Start of execution of an instruction
8--EX--1	External sense data. This data item is displayed when tracing of external sense data are selected by the TRS E command.
Clock	Time tag (the number of system clocks between frames). This data item is displayed when tracing of time tags is selected by the TRS T command.

Caution: If the internal RAM (0FE00H to 0FEFFH) is addressed in a mode other than the short direct addressing mode, trace data become unpredictable. In this case, instructions of the target program are executed normally, however.

Note: The WST cycle is a write cycle in a device. Cycles actually written into memory are WR cycles following the WST cycle.

When $F, $J, $Q, or $C is specified, trace data are retrieved and data satisfying the retrieval conditions are displayed.

```
                   ┌ $F ┐
                   │ $J │
.  When   TRD F [ ─┤    ├─ ] <cr>   is entered,
                   │ $Q │
                   └ $C ┘
```

the system retrieves data in the current trace block from the current trace pointer in the forward direction. When data satisfying the retrieval conditions are detected, the system displays the data, then enters the prompt mode.

```
                       ┌ $F ┐
                       │ $J │
.  When   TRD F ALL [ ─┤    ├─ ] <cr>   is entered,
                       │ $Q │
                       └ $C ┘
```

the system retrieves data in the current trace block from the first frame in that block in the forward direction. When data satisfying the retrieval conditions are detected, the system displays the data, then enters the prompt mode.

Cautions 1. xxx is entered for the time tag of the first frame because the value is undefined.

2. When a time tag value is 255 system clocks or greater, xxx is entered.

Example 1: To display all trace data in units of frames

```
emu:0>TRD F ALL <cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0000     OP     0100     65        00000000
  0001     OP     0101     00    M1  00000000
                   .
                   .
  2047     OP     010A     C8        00000000
emu:0>■
```

Example 2: To display the frame satisfying the retrieval conditions together with ten frames (five immediately before the frame and five immediately after the frame)

```
emu:0>TRD F $F <cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0029     OP     0100     65        00000000
  0030     OP     0101     00    M1  00000000
                   .
                   .
 !0033     OP     0134     C8        00000000
                   .
                   .
  0039     OP     0158     42        00000000
  Total frame = 2048  (F/L/T/P/N/+/cr/-/Frame No./.) ? ■
```

Example 3: To display all frames that satisfy the retrieval conditions

```
emu:0>TRD F $Q <cr>
  Frame  Status  Address  Data  Exu  8--EX--1
 !0012     WR     8001     32        00000000
 !0065     WR     8001     32        00000000
                   .
                   .
 !0143     WR     8001     32        00000000
emu:0>■
```

Example 4: To display frames concerning program branches

```
emu:0>TRD F $J <cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0000    BROP    0100     65         00000000
                           .
                           .
                           .
 < INTCM10 >
  0063    INTRD   001E     00         00000000
  0064    INTRD   001F     40         00000000
  0065    INTWR   FDFE     08         00000000
  0066    INTWR   FDFF     07         00000000
  Total frame = 1074   (F/L/T/P/N/+/cr/-/Frame No./.) ? <ESC>
emu:0>■
```

Example 5: To display the last 11 lines of the previous trace block

```
emu:0>TRD F <cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0027    OP      0100     65         00000000
  0030    OP      0101     00    M1   00000000
                           .
                           .
  0033    OP      0134     C8         00000000
                           .
                           .
  0037    OP      0158     42         00000000
  Total frame = 2048   (F/L/T/P/N/+/cr/-/Frame No./.) ? P<cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0018    OP      0100     65         00000000
  0020    OP      0101     00    M1   00000000
                           .
                           .
  0024    OP      0120     A0         00000000
                           .
                           .
  0028    OP      0111     30         00000000
  Total frame = 2048   (F/L/T/P/N/+/cr/-/Frame No./.) ? <ESC>
emu:0>■
```

Example 6: To display the first 11 lines of the next trace block

```
emu:0>TRD F <cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0015     OP      0100    65        00000000
  0016     OP      0101    00    M1  00000000
                           .
                           .
  0020     OP      0120    A0        00000000
                           .
                           .
  0025     OP      0111    30        00000000
  Total frame = 2048   (F/L/T/P/N/+/cr/-/Frame No./.) ? N<cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0026     OP      0100    65        00000000
  0030     OP      0101    00    M1  00000000
                           .
                           .
  0033     OP      0134    C8        00000000
                           .
                           .
  0036     OP      0158    42        00000000
  Total frame = 2048   (F/L/T/P/N/+/cr/-/Frame No./.) ? <ESC>
emu:0>█
```

Example 7: To display the 1024th frame and following 10 lines by entering 1024 for the frame No.

```
emu:0>TRD F <cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0029     OP      0100    65        00000000
  0030     OP      0101    00    M1  00000000
                           .
                           .
  0033     OP      0134    C8        00000000
                           .
                           .
  0039     OP      0158    42        00000000
  Total frame = 2048   (F/L/T/P/N/+/cr/-/Frame No./.) ? 1024<cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  1024     RD      FE10    65        00000000
  1025     OP      0101    00    M1  00000000
                           .
                           .
  1030     OP      0134    C8        00000000
                           .
                           .
  1034     WR      FE12    42        00000000
  Total frame = 2048   (F/L/T/P/N/+/cr/-/Frame No./.) ? <ESC>
emu:0>█
```

Example 8: To display marking information on trace data

```
emu:0>TRD F ALL <cr>
  Frame  Status  Address  Data  Exu  8--EX--1
  0000                     00   M1  00000000
  0001     RD      0100    00       00000000
  0002     OP      0101    12       00000000
  0003     OP      0010    35       00000000
  0004                     45   M1  00000000
  0005     OP      0124    0B       00000000
  0006     OP      0125    0C       00000000
<CHK>
 FE00      01
  Frame  Status  Address  Data  Exu  8--EX--1
  0009                     00   M1  00000000
  0010     OP      0125    00       00000000
  0011     WR      FF34    15       00000000
 T0012    BROP     0230    45       00000000
  0013                     BB   M1  00000000
  0014     OP      4508    30       00000000
  0015     OP      4411    02       00000000
[All trace mode terminated.]
  0016                     00   M1  00000000
  0017     OP      0012    55       00000000
  0018     OP      0013    12       00000000
[one step emulation terminated.]
  0019                     00   M1  00000000
  0020     OP      0012    55       00000000
  0021     OP      0013    12       00000000
[procedure emulation terminated.]
emu:0>■
```

(2) TRD I command

```
TRD  I  [ALL] [-┌$F┐-]
                │$J│
                │$Q│
                └$C┘
```

trc:0>	×	emu:0>	○	brk:0>	○

ALL: Displays the current trace data or all data that satisfy the retrieval conditions.

$F: Displays the frame which satisfies the retrieval conditions together with ten frames (five immediately before the frame and five immediately after the frame).

$J: Displays the frames concerning branches of the program.

$Q: Displays the frames which satisfy the retrieval conditions.

$C: Displays the frames at a check point.

The TRD I command displays the trace frames in the current trace block in the instruction mode.

Alternatively, this command retrieves trace data in the current trace block, and displays data satisfying the retrieval conditions in the instruction mode.

A trace frame line consists of the following data items:

Table 8-8 Data Items of a Trace Frame Line (TRD I)

Label	Description
Frame	The frame No. of trace memory, which is assigned sequentially from the first frame that is written into trace memory. The frame Nos. which can be displayed are values from 0000 to 8191.
Status	The bus cycle status of a trace frame . OP: Op-code fetch . RD: Read access . WR: Write access . INTRD: Read cycle by interrupt . INTWR: Write cycle by interrupt . MSRD: Read cycle by macro service . MSWR: Write cycle by macro service . BROP: First fetch after branch
Address	Fetch or read/write address
Data	Fetch or read/write data. Data enclosed in parentheses are invalid fetch data.
Label	Label symbol
Mnemonic	Instruction mnemonic
EX	External sense data. This data item is displayed when tracing of external sense data are selected by the TRS E command.
Clock	Time tag (the number of system clocks between frames). This data item is displayed when tracing of time tags is selected by the TRS T command.

Caution: If the internal RAM (OFE00H to OFEFFH) is addressed in a mode other than the short direct addressing mode, trace data become unpredictable. In this case, instructions of the target program are executed normally, however.

When $F, $J, $Q, or $C is specified, trace data are retrieved and data satisfying the retrieval conditions are displayed.

. When TRD I [$\begin{bmatrix} \$F \\ \$J \\ \$Q \\ \$C \end{bmatrix}$] <cr> is entered,

the system retrieves data in the current trace block from the current trace pointer in the forward direction. When data satisfying the retrieval conditions are detected, the system displays the data, then enters the prompt mode.

. When TRD I ALL [$\begin{bmatrix} \$F \\ \$J \\ \$Q \\ \$C \end{bmatrix}$] <cr> is entered,

the system retrieves data in the current trace block from the first frame in that block in the forward direction. When data satisfying the retrieval conditions are detected, the system displays the data, then enters the prompt mode.

Cautions 1. xxx is entered for the time tag of the first frame, because the value is undefined.

2. When a time tag value is 255 system clocks or greater, xxx is entered for the time tag.

3. Since read/write accesses to instructions are displayed, the order of displayed trace data may differ from that of the actual operation of the emulation CPU.

Cautions 4. Data traced in the qualified trace mode (TRM TRX) are displayed in units of frames, even if instruction trace display is specified by entering the TRD I command.

5. The first and last frames in data that are traced in the section trace mode (TRM SEC) may become undefined when instruction trace display is specified by entering the TRD I command.

Example 1: To display all trace data by instructions

```
emu:0>TRD I ALL <cr>
  Frame Status Address Data             Label  Mnemonic
                                        PUBLIC¥EXTACSS::
  0000          0100    80                     MOVW      RP0,#1234H
  0003          0103    80                     MOVW      !4000H,RP0
                        .
                        .
  1019          0109    81                     BR        $109H
emu:0>■
```

Example 2: To display the frame satisfying the retrieval conditions together with ten frames (five immediately before the frame and five immediately after the frame) by instructions

```
emu:0>TRD I $F <cr>
  Frame Status Address Data             Label  Mnemonic
                                        PUBLIC¥EXTACSS::
  0000          0100    80                     MOVW      RP0,#1234H
  0003          0103    80                     MOVW      !4000H,RP0
                        .
                        .
 !0007(Note)    0107    81                     NOP
                        .
                        .
  0014    WR    4001    12 81
  Total frame = 1020    (F/L/T/P/N/+/cr/-/Frame No./.) ? ■
```

Note: The line starting with ! is the frame which satisfies the retrieval conditions.

Example 3: To display all frames that satisfy the retrieval conditions by instructions

```
emu:0>TRD I $Q <cr>
  Frame Status Address Data             Label  Mnemonic
                                        PUBLIC¥EXTACSS::
 !0007          0107    81                     NOP
 !0028          0107    81                     NOP
                        .
                        .
 !1003          0107    81                     NOP
  Total frame = 1020     (F/L/T/P/N/+/cr/-/Frame No./.) ? ■
```

Example 4: To display frames concerning program branches by instructions

```
emu:0>TRD I $J <cr>
  Frame Status Address Data             Label  Mnemonic
  0053          012F                           BR        $12FH
  0057          012F                           BR        $12FH

  0063   INTRD  001E   00
 T0063   INTRD  001E   00  ← Frame which detected an event
  0064   INTRD  001F   10
  0065   INTWR  FDFE   08
  0066   INTWR  FDFF   00
  0068   INTWR  FDFC   2F
  Total frame = 1074   (F/L/T/P/N/+/cr/-/Frame No./.)? <ESC>
emu:0>■
```

Example 5: To display frames concerning section tracing by instructions

```
brk:0>TRD I ALL <cr>
 Frame Status Address Data     Label  Mnemonic              EX
  8180          0180   2C2003         BR       !320H

 T8185   BROP   0320   14                                   2F
  8185          0320   14FE           BR       $322H

  8188          0322   00             NOP
  8191    OP    0321   FE                                   3F
[SEC trace mode terminated.]
brk:0>■
```

Example 6: To display frames traced in the qualified trace mode

```
brk:0>TRD | ALL <cr>
 Frame Status   Address  Data  Exu  7--EX--0
  8178    OP     0164     1B         00000011
  8179    OP     0165    (00)        00000011
 <INTCM10>
  8180   BROP    0180    m2C         00000011
  8181                    20    M1   00000011
  8182    OP     0181     20         00000011
  8183    OP     0182     03         00000011
  8184    OP     0183    (00)        00000011
 <CHK>
   P5       10
 Frame Status   Address  Data  Exu  7--EX--0
 T8185   BROP    0320    m14         00101111
  8186                    FE    M1   00111111
  8187    OP     0321     FE         00111111
  8188    OP     0322    (00)        00111111
 [TRX trace mode terminated.]
brk:0>
```

8.44 Setting the Trace Data Retrieval Conditions (TRF)

```
        ┌word       ┐
TRF  [A=┤           ├][V=mask8][C=status][E=mask8]
        └partition  ┘
```

Radix	word:H partition:H mask8:H

trc:0>	×	emu:0>	○	brk:0>	○

word:	Retrieval address
partition:	Retrieval address range
status:	Retrieval status
mask8:	8-bit mask data

The TRF command sets the retrieval conditions of trace data.

The retrieval conditions can be set in the interactive mode or by specifying the operands of the TRF command on a line.

To set the retrieval conditions in the interactive mode, first display the current retrieval conditions and set each condition sequentially.

When TRF <cr> is entered, the system displays the current retrieval conditions and enters the interactive mode.

Up to five retrieval addresses can be set, separated from each other by a space. When no address is set, all address areas are set.

Retrieval data are set in 8-bit mask data. When no retrieval data are set, all data are set.

Selects one of the following bus cycle attributes for the retrieval status. When no attribute is set, NC is set.

- . BROP: First fetch after branch
- . OP: Op-code fetch
- . RWI: Read/write by interrupt
- . RI: Read by interrupt
- . WI: Write by interrupt
- . RW: Data read/write
- . R: Data read
- . W: Data write
- . RWP: Data read/write by program
- . RP: Data read by program
- . WP: Data write by program
- . RWM: Data read/write by macro service
- . RM: Data read by macro service
- . WM: Data write by macro service
- . NC: All read/write operations including op-code fetch

External data to be retrieved, which are input from the external sense clips (Nos. 1. to 8), are set in 8-bit mask data.

When no external data are set, all external data are set.

Example 1: To set the retrieval conditions on a line

```
brk:0>TRF A=300X 0FE00,0FE7F V=55H C=WP E=0XXH <cr> (Note)
brk:0>
```

Note: Trace data are retrieved with 55H program-written at addresses 3000H to 300FH and 0FE00H to 0FE7FH.

Example 2: To set the retrieval conditions in the interactive mode

```
brk:0>TRF <cr>
 A=3000H,300FH,FE00H,FE7FH ┐
 V=55H                     │ ← The current retrieval conditions are displayed.
 C=WP                      │
 E=0XXH                    ┘

 A=4000 <cr>                          ← Changed to 4000H.
 V=0XXH <cr>                          ← Changed to 0XXH.
  BROP (BRanch OPecode fetch)       R     (Read)
  OP   (OPecode fetch)              W     (Write)
  RWI  (Read Write by Interrupt)    RWP   (Read Write by Program)
  RI   (Read by Interrupt)          RP    (Read by program)
  WI   (Write by Interrupt)         WP    (Write by Program)
  NC   (No Condition)               RWM   (Read Write by Macro service)
  RW   (Read Write)                 RM    (Read by Macro service)
                                    WM    (Write by Macro service)

 C=BROP <cr>                          ← Changed to BROP.
 E=1H <cr>                            ← Changed to 1H.
brk:0>■
```

8.45 Restarting the Analyzer (TRG)

TRG

trc:0>	×	emu:0>	○	brk:0>	×

The TRG command restarts the analyzer, real-time tracer, and internal RAM data sampler.

When the analyzer is restarted, the emulation CPU must be operating and the analyzer must be stopped (in the emu:0> prompt mode).

Example: To restart the analyzer

```
emu:0>TRG <cr>
Analyzer start
trc:0>■
```

8.46 Setting the Trace Mode (TRM)

```
TRM [ ALL ]
    [ TRX ]
    [ SEC ]
```

trc:0>	×	emu:0>	○	brk:0>	○

ALL: All trace
TRX: Qualified trace
SEC: Section trace

The TRM command sets the trace mode.

When TRM ALL <cr> is entered, tracing starts after a RUN B or RUN N command is entered.

When TRM TRX <cr> is entered, tracing is performed only when an event specified in the TRX command occurs.

When TRM SEC <cr> is entered, only the section from ENB to DSB is traced.

When TRM <cr> is entered, the current trace mode is displayed.

Example 1: To set the trace mode to qualified trace

```
brk:0>TRM TRX <cr>
brk:0>■
```

Example 2: To display the current trace source

```
brk:0>TRM <cr>
 TRX              <- Qualified trace is set.
brk:0>■
```

Example 3: To set the trace mode to section trace

```
brk:0>TRM SEC <cr>
brk:0>■
```

8.47 Selecting Trace Data (TRS)

```
TRS [ E ]
      T
```

trc:0>	×	emu:0>	○	brk:0>	○

E: External data

T: Time tag

The TRS command specifies whether external data or time tags are traced.

When TRS E <cr> is entered, external data are selected.

When TRS T <cr> is entered, time tags are selected.

When TRS <cr> is entered, the current selection is displayed.

Example 1: To select external data for trace data

```
brk:0>TRS E <cr>
brk:0>
```

Example 2: To select time tags for trace data

```
brk:0>TRS T <cr>
brk:0>■
```

Example 3: To display the current setting

```
brk:0>TRS <cr>        ← Displays the current setting.
 Time_tag             ← Time_tag is set.
brk:0>■
```

8.48 Setting Trace Qualify (TRX)

```
        ┌[BRA1][BRA2][BRA3][BRA4]┐
TRX  [ ─┤                        ├─ ]
        └OFF                     ┘
```

trc:0>	×	emu:0>	○	brk:0>	○

BR? (Note): Qualify conditions
OFF: Release of qualify

Note: BR? indicates the qualify conditions BRA1, BRA2, BRA3, and BRA4.

The TRX command specifies event trigger sources in the form of trace qualify conditions. Up to four qualify conditions can be specified.

When TRX OFF <cr> is entered, the specification of trace qualify is released.

When TRX <cr> is entered, the set qualify conditions are displayed.

The TRX command does not function when TRX is not selected in the TRM command.

Example 1: To set BRA1 and BRA2 as the qualify condition

```
brk:0>TRX BRA1 BRA2 <cr>
brk:0>■
```

Example 2: To display the set qualify conditions

```
brk:0>TRX <cr>
 BRA1 BRA2
brk:0>■
```

8.49 Verifying an Object File and Memory Contents (VRY)

VRY file

trc:0>	×	emu:0>	×	brk:0>	○

file: File name

The VRY command verifies the object code saved in a file for the host machine and the memory contents.

The verification of memory depends upon the mapping status.

When the contents of the object file differs from the memory contents, the address, object data, and memory contents are displayed.

Example 1: To verify data in a SAMPLE.HEX file and the memory contents

```
brk:0>VRY SAMPLE.HEX <cr>    ← Verifies the SAMPLE.HEX file and memory contents.
 object verify complete      ← Message output for normal termination
brk:0>■
```

Example 2: When an invalid verification result is obtained

```
brk:0>VRY SAMPLE.HEX <cr>
 object verify
 Address  File  Memory     ← Message output when an invalid verification result
  0123     00     01          is obtained
  1234     FF     FE
brk:0>■
```

Example 3: When the specified file is missing

```
brk:0>VRY SAMPLE.HEX <cr>
 File not found.  (509)(Note)  SAMPLE.HEX  ← Message output when the specified
brk:0>■                                      file (SAMPLE.HEX) is missing
```

Note: The number in parentheses indicates the error message No.

8.50 Setting the Memory Word Length (WRD)

```
        ┌B┐
WRD  [──┤ ├──]
        └W┘
```

trc:0>	emu:0>	brk:0>
×	○	○

B: Byte specification

W: Word specification

The WRD command sets the data length (memory length) when data are displayed or changed in memory manipulation operations.

Select B (byte) or W (word) for the data length.

When WRD <cr> is entered, the current data length is displayed.

Example 1: To change the memory length from word to byte

```
brk:0>WRD B <cr>
brk:0>■
```

Example 2: To display the current memory length

```
brk:0>WRD <cr>
Word                    ← Word is set.
brk:0>■
```

CHAPTER 9 ERROR MESSAGES

This chapter explains the error messages of the IE-78330-R. The error messages are listed in numeric order by error code. Table 9-1 explains the type of error according to the error code.

Table 9-1 Error Message Codes and Types

Code	Type
000-002	System error
100-112	Error common to emulator
200-210	Analyzer error
300-304	Online assembler error
400-409	Symbol error
500-520	File error (load, save, PGM)

Conventions

o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

9.1 System Error (Codes 000 to 002)

Error code: (000)
Message: Communication error. (000)
Explanation: Communication between the IE-78330-R and the host machine could not be established normally.

Error code: (001)
Message: Illegal hardware. Enter RES command. (001)
Explanation: A hardware failure occurred.

Error code: (002)
Message: Illegal hardware. Enter RES H command. (002)
Explanation: A hardware failure occurred.

9.2 Error Common to Emulator (Codes 100 to 112)

Error code: (100)
Message: Command/Data too long. (100)
Explanation: The entered command or data contains 128 characters or more.

Error code: (101)
Message: Unrecognized command. (101)
Explanation: An invalid command was entered.

Error code: (102)
Message: Command format error. (102)
Explanation: The command name is valid, but the operand is invalid.

Error code: (103)
Message: Unexecutable command. (103)
Explanation: The entered command cannot be executed in the current mode.

Error code: (104)
Message: Input data error. (104)
Explanation: Invalid data were entered.

Error code: (105)
Message: aborted. (105)
Explanation: Processing was aborted.

Error code: (106)
Message: Mapping error. (106)
Explanation: Some memory areas are not mapped within the specified range of addresses.

Error code: (107)
Message: Non map area access. (107)
Explanation: An attempt was made to access unmapped memory during command execution.

Error code: (108)
Message: Can not test. (108)
Explanation: There is no memory which can be tested.

Error code: (109)
Message: List device is used by other command. (109)
Explanation: The printer is used by other commands.

Error code: (110)
Message: Warning double define:module-name (110)
Explanation: The same module name was specified more than once in the LOD command.

Error code: (111)
Message: Can not execute HLP command! (111)
Explanation: No help file was found on the same directory as IE78330.EXE when the emulator was started.

Error code: (112)
Message: Keyword Error. (112)
Explanation: An invalid command name was specified in the HLP command.

9.3 Analyzer Error (Codes 200 to 210)

Error code: (200)
Message: Emulation Timer overflow. (200)
Explanation: The emulation timer overflowed.

Error code: (201)
Message: Instruction counter overflow. (201)
Explanation: The instruction counter overflowed.

Error code: (204)
Message: External trigger line short. (204)
Explanation: External sense clip 1 is connected to an output pin.

Error code: (205)
Message: No PSA address. (205)
Explanation: PSA (sample address) is not set.

Error code: (206)
Message: No sampled data. (206)
Explanation: No sampled data were found.

Error code: (207)
Message: No traced data. (207)
Explanation: No traced data were found.

Error code: (208)
Message: Not found. (208)
Explanation: The target data were not found.

Error code: (209)
Message: Trace block not found. (209)
Explanation: No trace block was found.

Error code: (210)
Message: Trigger frame not found. (210)
Explanation: No trigger frame was found in trace data.

9.4 Online Assembler Error (Codes 300 to 304)

Error code: (300)
Message: Assemble area over! (300)
Explanation: The accessible range of memory was exceeded in the ASM command.

Error code: (301)
Message: Disassemble area over! (301)
Explanation: The accessible range of memory was exceeded in the DAS command.

Error code: (302)
Message: Error! (302)
Explanation: No object code can be generated. Or, it is clear that an error occurred.

Error code: (303)
Message: Caution! (303)
Explanation: A generic object code was generated. Or, caution needs be exercised.

Error code: (304)
Message: Warning! (304)
Explanation: An object code can be generated, but normal operation cannot be performed.

9.5 Symbol Error (Codes 400 to 409)

Error code: (400)
Message: IE78330.SYM file not found. (400)
Explanation: The IE78330.SYM file was not found on the current disk when the SYM L command was executed.

Error code: (401)
Message: Illegal IESYMBOL file. (401)
Explanation: The append symbol file is invalid in format in the SYM L command.

Error code: (402)
Message: Double define symbol. (402) symbol-name
Explanation: An attempt was made to register a registered symbol.

Error code: (403)
Message: Double define module name. (403) module-name
Explanation: The module name is already registered.

Error code: (404)
Message: No symbol. (404)
Explanation: No symbol was found.

Error code: (405)
Message: Reserved symbol. (405)
Explanation: A reserved word was defined as a symbol in the SYM A command.

Error code: (406)
Message: IESYMBOL table full. (406)
Explanation: No free IE symbol save area was found when the SYM A or SYM L command was executed.

Error code: (407)

Message: Symbol not found. (407)

Explanation: The symbol specified in the SYM C or SYM E command was not found.

Error code: (408)

Message: Symbol module table full. (408)

Explanation: The number of modules which can be entered was exceeded in the LQD command.

Error code: (409)

Message: Symbol table full. (409)

Explanation: No free symbol save area was found when the LOD command was executed.

9.6 File Error (Load, Save, PGM) (Codes 500 to 520)

Error code: (500)
Message: Can not close (500) file-name
Explanation: The file could not be closed normally.

Error code: (501)
Message: Can not close file-name.Cancel XXX command. (501)
Explanation: The file could not be closed normally while the XXX command was executed (XXX: STR, LST, or COM command).

Error code: (502)
Message: Can not open. (502) file-name
Explanation: The specified file could not be opened.

Error code: (503)
Message: Disk read error. (503) file-name
Explanation: A read error occurred in the file.

Error code: (504)
Message: Disk read error file-name.Cancel STR command. (504)
Explanation: A read error occurred in the file while the STR command was executed.

Error code: (505)
Message: Disk write error. (505) file-name
Explanation: A write error occurred in the file.

Error code: (506)
Message: Disk write error.file-name.Cancel XXX command. (506)
Explanation: A write error occurred in the file while the XXX command was executed (XXX: LST or COM command).

Error code: (507)
Message: Read only file. (507) file-name
Explanation: An attempt was made to create a file whose name is the same as that of the file with an R/O attribute.

Error code: (508)
Message: File make error. (508) file-name
Explanation: The file could not be created.

Error code: (509)
Message: File not found. (509)
Explanation: The specified file name was not found.

Error code: (510)
Message: HELP file not found. (510)
Explanation: No help file was found on the same directory as IE78330.EXE when the HLP command was executed.

Error code: (511)
Message: Bad file entry. (511)
Explanation: A file name was invalid in format.

Error code: (512)
Message: File already opened. (512) file-name
Explanation: The name of an opened file was specified.

Error code: (513)
Message: File already exists. (513)
Explanation: The specified file already exists.

Error code: (514)
Message: Reserved file name. (514)
Explanation: The reserved file name to be used by system software was specified.

Error code: (515)
Message: Bad character. (515)
Explanation: An invalid character was detected while an object code was loaded or saved.

Error code: (516)
Message: Check sum error. (516)
Explanation: A checksum error was detected while an object code was loaded or saved.

Error code: (517)
Message: Illegal record. (517)
Explanation: The symbol table file is invalid in record format in the LOD command.

Error code: (518)
Message: Load failed. (518)
Explanation: An error occurred while a symbol or an object code was loaded with the LOD command.

Error code: (519)
Message: Module not found. (519)
Explanation: The module name specified in the LOD command was not found in a symbol file.

Error code:	(520)
Message:	Multi define. (520)
Explanation:	The same character was set in the PGM command when control characters were changed.

CHAPTER 10 ONLINE ASSEMBLER AND DISASSEMBLER SPECIFICATIONS

This chapter explains the instruction set and special function registers (SFRs) of the target device (uPD78330, uPD78334, or uPD78P334) and the online assembler and disassembler specifications.

Conventions

o <cr>: Indicates that the return key (CR (0DH)) needs to be pressed.

o R/O: Read only

o R/W: Read/write

o W/O: Write only

o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

10.1 Instruction Set of the Target Device

The instruction set of the target device (uPD78330, uPD78334, or uPD78P334) is divided functionally into 19 instruction types:

- o 8-bit data transfer
- o 16-bit data transfer
- o 8-bit arithmetic/logical
- o 16-bit arithmetic/logical
- o Multiply/divide
- o Signed multiply/divide
- o Increment/decrement
- o Shift/rotate
- o BCD correction
- o Data conversion
- o Bit manipulation
- o Call/return
- o Stack manipulation
- o Special
- o Unconditional branch
- o Conditional branch
- o Context switching
- o String
- o CPU control

o Operand notation and coding format

Notation	Coding
r r1 r2	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15 R0, R1, R2, R3, R4, R5, R6, R7 C, B
rp rp1 rp2	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7 RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7 DE, HL, VP, UP
sfr sfrp	Special function register abbreviation(*1) Special function register abbreviation (16-bit manipulation register)(*2)
post	RP0, RP1, RP2, RP3, RP4, RP5/PSW, RP6, RP7 (Can be coded more than once. However, RP5 can only be used in a PUSH or POP instruction, and PSW can only be used in a PUSHU or POPU instruction.)
mem	[DE], [HL], [DE+], [HL+], [DE-], [HL-], [VP], [UP]: Register indirect mode [DE+A], [HL+A], [DE+B], [HL+B], [VP+DE], [VP+HL]: Based indexed mode [DE+byte], [HL+byte], [VP+byte], [UP+byte], [SP+byte]: Based mode word[A], word[B], word[DE], word[HL]: Indexed mode
saddr saddrp	0FE20H-0FF1FH Immediate data or label 0FE20H-0FF1EH Immediate data (bit 0 = 0, however) or label (for 16-bit manipulation)
$addr16 !addr16 addr11 addr5	0000H-0FDFFH Immediate data or label: Relative addressing 0000H-0FDFFH Immediate data or label: Immediate addressing (Data up to 0FFFFH can be coded in an MOV instruction.) 800H-0FFFH Immediate data or label 40H-7EH Immediate data (bit 0 = 0, however)(*3) or label
word byte bit n	16-bit immediate data or label 8-bit immediate data or label 3-bit immediate data or label 3-bit immediate data (0 to 7)

*1 See Table 10-1.

*2 See Table 10-2.

*3 Do not attempt to access word data at odd address (bit 0 = 1).

Remarks 1. The same register name can be specified in rp and rp1, but different codes are generated. Refer to the uPD78334 User's Manual (IEU-729) for details.

2. Functional names (X, A, C, B, E, D, L, H, AX, BC, DE, HL, VP, UP) can be specified in r, r1, rp, rp1, and post, as well as absolute names (R0 to R15, RP0 to RP7).

3. Immediate addressing is effective for all address spaces. Relative addressing is effective for the locations within a displacement range of -128 to +127 from the starting address of the next instruction.

4. Conventions for explaining operations

Symbol	Explanation
A	Register A; 8-bit accumulator
X	Register X
B	Register B
C	Register C
D	Register D
E	Register E
H	Register H
L	Register L
R0-R15	Register 0 to register 15 (absolute names)
AX	Register pair (AX); 16-bit accumulator
BC	Register pair (BC)
DE	Register pair (DE)
HL	Register pair (HL)
RP0-RP7	Register pair 0 to register pair 7 (absolute names)
PC	Program counter
SP	Stack pointer
UP	User stack pointer
PSW	Program status word
CY	Carry flag
AC	Auxiliary carry flag
Z	Zero flag
P/V	Parity/overflow flag
T	Sign flag
TPF	Table position flag

(to be continued)

(Cont'd)

Symbol	Explanation
RBS	Register bank select flag
RSS	Register set select flag
IE	Interrupt enable flag
STBC	Standby control register
WDM	Watchdog timer mode register
( )	Contents of memory indicated by the address or the register contents enclosed in parentheses. (+) or (-) indicates that the contents in parentheses are incremented or decremented by one after execution of the instruction.
(( ))	Contents of memory indicated by the contents of memory indicated by the address enclosed in nested parentheses
xxH	Hexadecimal number
x_H, x_L	8 high-order bits and 8 low-order bits of a 16-bit register
!xx	Address indicated through direct addressing
$xx	Address indicated through relative addressing

5. Symbols in flag field

Symbol	Explanation
(blank)	No change
0	Cleared to zero.
1	Set to 1.
X	Set or reset according to the result.
P	The P/V flag operates as a parity flag.
V	The P/V flag operates as an overflow flag.
R	The saved value is restored.

Instruction type	Mnemonic	Operand	Operation	Flag S	Z	AC	P/V	CY
8-bit data transfer	**MOV**	r1,#byte	r1 ← byte					
		saddr,#byte	(saddr) ← byte					
		sfr(*),#byte	sfr ← byte					
		r,r1	r ← r1					
		A,r1	A ← r1					
		A,saddr	A ← (saddr)					
		saddr,A	(saddr) ← A					
		saddr,saddr	(saddr) ← (saddr)					
		A,sfr	A ← sfr					
		sfr,A	sfr ← A					
		A,mem	A ← (mem)					
		mem,A	(mem) ← A					
		A,[saddrp]	A ← ((saddrp))					
		[saddrp],A	((saddrp)) ← A					
		A,!addr16	A ← (!addr16)					
		!addr16,A	(!addr16) ← A					
		PSWL,#byte	PSW_L ← byte	×	×	×	×	×
		PSWH,#byte	PSW_H ← byte					
		PSWL,A	PSW_L ← A	×	×	×	×	×
		PSWH,A	PSW_H ← A					
		A,PSWL	A ← PSW_L					
		A,PSWH	A ← PSW_H					
	XCH	A,r1	A ←→ r1					
		r,r1	r ←→ r1					
		A,mem	A ←→ (mem)					
		A,saddr	A ←→ (saddr)					
		A,sfr	A ←→ sfr					
		A,[saddrp]	A ←→ ((saddrp))					
		saddr,saddr	(saddr)←→(saddr)					
16-bit data transfer	**MOVW**	rp1,#word	rp1 ← word					
		saddrp,#word	(saddrp) ← word					
		sfrp,#word	sfrp ← word					
		rp,rp1	rp ← rp1					
		AX,saddrp	AX ←(saddrp)					
		saddrp,AX	(saddrp) ← AX					
		saddrp,saddrp	(saddrp) ← (saddrp)					
		AX,sfrp	AX ← sfrp					
		sfrp,AX	sfrp ← AX					
		rp1,!addr16	rp1 ← (addr16)					
		!addr16,rp1	(addr16) ← rp1					
		AX,mem	AX ← (mem)					
		mem,AX	(mem) ← AX					

(to be continued)

* If STBC or WDM is coded in sfr, the instruction is used as a special instruction, and both differ in the number of bytes.

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	Flag				
				S	Z	AC	P/V	CY
(*)	XCHW	AX,saddrp	AX ←→ (saddrp)					
		AX,sfrp	AX ←→ sfrp					
		saddrp,saddrp	(saddrp) ←→ (saddrp)					
		rp,rp1	rp ←→ rp1					
8-bit arithmetic/logical	ADD	A,#byte	A,CY ← A+byte	×	×	×	V	×
		saddr,#byte	(saddr),CY ← (saddr)+byte	×	×	×	V	×
		sfr,#byte	Sfr, CY ← sfr+byte	×	×	×	V	×
		r,r1	r,CY ← r+r1	×	×	×	V	×
		A,saddr	A,CY ← A+(saddr)	×	×	×	V	×
		A,sfr	A,CY ← A+sfr	×	×	×	V	×
		saddr,saddr	(saddr),CY←(saddr)+(saddr)	×	×	×	V	×
		A,mem	A,CY ← A+(mem)	×	×	×	V	×
		mem,A	(mem),CY ← (mem)+A	×	×	×	V	×
	ADDC	A,#byte	A,CY ← A+byte+CY	×	×	×	V	×
		saddr,#byte	(saddr),CY←(saddr)+byte+CY	×	×	×	V	×
		sfr,#byte	sfr,CY ← sfr+byte+CY	×	×	×	V	×
		r,r1	r,CY ← r+r1+CY	×	×	×	V	×
		A,saddr	A,CY←A+(saddr)+CY	×	×	×	V	×
		A,sfr	A,CY ← A+sfr+CY	×	×	×	V	×
		saddr,saddr	(saddr),CY←(saddr)+(saddr)+CY	×	×	×	V	×
		A,mem	A,CY←A+(mem)+CY	×	×	×	V	×
		mem,A	(mem),CY ← (mem)+A+CY	×	×	×	V	×
	SUB	A,#byte	A,CY ← A-byte			×	×	×
		saddr,#byte	(saddr),CY ← (saddr)-byte	×	×	×	V	×
		sfr,#byte	sfr,CY ← sfr-byte	×	×	×	V	×
		r,r1	r,CY ← r-r1	×	×	×	V	×
		A,saddr	A,CY ← A-(saddr)	×	×	×	V	×
		A,sfr	A,CY ← A-sfr	×	×	×	V	×
		saddr,saddr	(saddr),CY←(saddr)-(saddr)	×	×	×	V	×
		A,mem	A,CY ← A-(mem)	×	×	×	V	×
		mem,A	(mem),CY←(mem)-A	×	×	×	V	×
	SUBC	A,#byte	A,CY←A-byte-CY	×	×	×	V	×
		saddr,#byte	(saddr),CY←(saddr)-byte-CY	×	×	×	V	×
		sfr,#byte	sfr,CY ← sfr-byte-CY	×	×	×	V	×
		r,r1	r,CY ← r-r1-CY	×	×	×	V	×
		A,saddr	A,CY ← A-(saddr)-CY	×	×	×	V	×
		A,sfr	A,CY ← A-sfr-CY	×	×	×	V	×
		saddr,saddr	(saddr),CY←(saddr)-(saddr)-CY	×	×	×	V	×
		A,mem	A,CY ← A-(mem)-CY	×	×	×	V	×
		mem,A	(mem),CY ← (mem)-A-CY	×	×	×	V	×

(to be continued)

* 16-bit data transfer

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	Flag				
				S	Z	AC	P/V	CY
8-bit arithmetic/logical	AND	A,#byte	A ← A ∧ byte	×	×		P	
		saddr,#byte	(saddr) ← (saddr) ∧ byte	×	×		P	
		sfr,#byte	sfr ← sfr ∧ byte	×	×		P	
		r,r1	r ← r ∧ r1	×	×		P	
		A,saddr	A ← A ∧ (saddr)	×	×		P	
		A,sfr	A ← A ∧ sfr	×	×		P	
		saddr,saddr	(saddr)←(saddr) ∧ (saddr)	×	×		P	
		A,mem	A ← A ∧ (mem)	×	×		P	
		mem,A	(mem) ← (mem) ∧ A	×	×		P	
	OR	A,#byte	A ← A ∨ byte	×	×		P	
		saddr,#byte	(saddr) ← (saddr) ∨ byte	×	×		P	
		sfr,#byte	sfr ← sfr ∨ byte	×	×		P	
		r,r1	r ← r ∨ r1	×	×		P	
		A,saddr	A ← A ∨ (saddr)	×	×		P	
		A,sfr	A ← A ∨ sfr	×	×		P	
		saddr,saddr	(saddr) ← (saddr)∨ (saddr)	×	×		P	
		A,mem	A ← A ∨ (mem)	×	×		P	
		mem,A	(mem) ← (mem) ∨ A	×	×		P	
	XOR	A,#byte	A ← A ∀ byte	×	×		P	
		saddr,#byte	(saddr) ← (saddr) ∀ byte	×	×		P	
		sfr,#byte	sfr ← sfr ∀ byte	×	×		P	
		r,r1	r ← r ∀ r1	×	×		P	
		A,saddr	A ← A ∀ (saddr)	×	×		P	
		A,sfr	A ← A ∀ sfr	×	×		P	
		saddr,saddr	(saddr)←(saddr) ∀ (saddr)	×	×		P	
		A,mem	A ← A ∀ (mem)	×	×		P	
		mem,A	(mem) ← (mem) ∀ A	×	×		P	
	CMP	A,#byte	A-byte	×	×	×	V	×
		saddr,#byte	(saddr)-byte	×	×	×	V	×
		sfr,#byte	sfr-byte	×	×	×	V	×
		r,r1	r-r1	×	×	×	V	×
		A,saddr	A-(saddr)	×	×	×	V	×
		A,sfr	A-sfr	×	×	×	V	×
		saddr,saddr	(saddr)-(saddr)	×	×	×	V	×
		A,mem	A-(mem)	×	×	×	V	×
		mem,A	(mem)-A	×	×	×	V	×

(to be continued)

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	S	Z	AC	P/V	CY
16-bit arithmetic/logical	ADDW	AX,#word	AX,CY ← AX+word	×	×	×	V	×
		saddrp,#word	(saddrp),CY ← (saddrp)+word	×	×	×	V	×
		sfrp,#word	sfrp,CY ← sfrp+word	×	×	×	V	×
		rp,rp1	rp,CY ← rp+rp1	×	×	×	V	×
		AX,saddrp	AX,CY ← AX+(saddrp)	×	×	×	V	×
		AX,sfrp	AX,CY ← AX+sfrp	×	×	×	V	×
		saddrp,saddrp	(saddrp),CY ← (saddrp)+(saddrp)	×	×	×	V	×
	SUBW	AX,#word	AX,CY ← AX-word	×	×	×	V	×
		saddrp,#word	(saddrp),CY ← (saddrp)-word	×	×	×	V	×
		sfrp,#word	sfrp,CY ← sfrp-word	×	×	×	V	×
		rp,rp1	rp,CY ← rp-rp1	×	×	×	V	×
		AX,saddrp	AX,CY ← AX-(saddrp)	×	×	×	V	×
		AX,sfrp	AX,CY ← AX-sfrp	×	×	×	V	×
		saddrp,saddrp	(saddrp),CY ← (saddrp)-(saddrp)	×	×	×	V	×
	CMPW	AX,#word	AX-word	×	×	×	V	×
		saddrp,#word	(saddrp)-word	×	×	×	V	×
		sfrp,#word	sfrp-word	×	×	×	V	×
		rp,rp1	rp-rp1	×	×	×	V	×
		AX,saddrp	AX-(saddrp)	×	×	×	V	×
		AX,sfrp	AX-sfrp	×	×	×	V	×
		saddrp,saddrp	(saddrp)-(saddrp)	×	×	×	V	×
Multiply/divide	MULU	r1	AX ← AX×r1					
	DIVUW	r1	AX (quotient), r1 (remainder) ← AX÷r1					
	MULUW	rp1	AX (high-order 16 bits), rp1 (low-order 16 bits) ← AX×rp					
	DIVUX	rp1	AXDE (quotient), rp1 (remainder) ← AXDE÷rp1					
(*)	MULW	rp1	AX (high-order 16 bits), rp1(low-order 16 bits) ← AX×rp1					
Increment/decrement	INC	r1	r1 ← r1+1	×	×	×	V	
		saddr	(saddr) ← (saddr)+1	×	×	×	V	
	DEC	r1	r1 ← r1-1	×	×	×	V	
		saddr	(saddr) ← (saddr)-1	×	×	×	V	
	INCW	rp2	rp2 ← rp2+1					
		saddrp	(saddrp) ← (saddrp)+1					
	DECW	rp2	rp2 ← rp2-1					
		saddrp	(saddr) ← (saddrp)-1					

(to be continued)

* Signed multiply

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	S	Z	AC	P/V	CY
Shift/rotate	ROR	rl,n	$(CY, rl_7 \leftarrow rl_0, rl_{m-1} \leftarrow rl_m) \times n$ times				P	×
	ROL	rl,n	$(CY, rl_0 \leftarrow rl_7, rl_{m+1} \leftarrow rl_m) \times n$ times				P	×
	RORC	rl,n	$(CY \leftarrow rl_0, rl_7 \leftarrow CY, rl_{m-1} \leftarrow rl_m) \times n$ times				P	×
	ROLC	rl,n	$(CY \leftarrow rl_7, rl_0 \leftarrow CY, rl_{m+1} \leftarrow rl_m) \times n$ times				P	×
	SHR	rl,n	$(CY \leftarrow rl_0, rl_7 \leftarrow 0, rl_{m-1} \leftarrow rl_m) \times n$ times	×	×	0	P	×
	SHL	rl,n	$(CY \leftarrow rl_7, rl_0 \leftarrow 0, rl_{m+1} \leftarrow rl_m) \times n$ times	×	×	0	P	×
	SHRW	rpl,n	$(CY \leftarrow rpl_0, rpl_{15} \leftarrow 0, rpl_{m-1} \leftarrow rpl_m) \times n$ times	×	×	0	P	×
	SHLW	rpl,n	$(CY \leftarrow rpl_{15}, rpl_0 \leftarrow 0, rpl_{m+1} \leftarrow rpl_m) \times n$ times	×	×	0	P	×
	ROR4	[rpl]	$A_{3-0} \leftarrow (rpl)_{3-0}$, $(rpl)_{7-4} \leftarrow A_{3-0}$, $(rpl)_{3-0} \leftarrow (rpl)_{7-4}$					
	ROL4	[rpl]	$A_{3-0} \leftarrow (rpl)_{7-4}$, $(rpl)_{3-0} \leftarrow A_{3-0}$, $(rpl)_{7-4} \leftarrow (rpl)_{3-0}$					

(to be continued)

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	Flag S	Z	AC	P/V	CY
(*1)	ADJBA		Decimal Adjust Accumulator	×	×	×	P	×
	ADJBS							
(*2)	CVTBW		When A_7=0 X ← A, A ← 00H When A_7=1 X ← A, A ← FFH					
Bit manipulation	MOV1	CY,saddr.bit	CY ← (saddr.bit)					×
		CY,sfr.bit	CY ← sfr.bit					×
		CY,A.bit	CY ← A.bit					×
		CY,X.bit	CY ← X.bit					×
		CY,PSWH.bit	CY ← PSW_H.bit					×
		CY,PSWL.bit	CY ← PSW_L.bit					×
		saddr.bit,CY	(saddr.bit) ← CY					
		sfr.bit,CY	sfr.bit ← CY					
		A.bit,CY	A.bit ← CY					
		X.bit,CY	X.bit ← CY					
		PSWH.bit,CY	PSW_H.bit ← CY					
		PSWL.bit,CY	PSW_L.bit ← CY					
	AND1	CY,saddr.bit	CY ← CY ∧ (saddr.bit)					×
		CY,/saddr.bit	CY ← CY ∧ $\overline{\text{(saddr.bit)}}$					×
		CY,sfr.bit	CY ← CY ∧ sfr.bit					×
		CY,/sfr.bit	CY ← CY ∧ $\overline{\text{sfr.bit}}$					×
		CY,A.bit	CY ← CY ∧ A.bit					×
		CY,/A.bit	CY ← CY ∧ $\overline{\text{A.bit}}$					×
		CY,X.bit	CY ← CY ∧ X.bit					×
		CY,/X.bit	CY ← CY ∧ $\overline{\text{X.bit}}$					×
		CY,PSWH.bit	CY ← CY ∧ PSW_H.bit					×
		CY,/PSWH.bit	CY ← CY ∧ $\overline{PSW_H\text{.bit}}$					×
		CY,PSWL.bit	CY ← CY ∧ PSW_L.bit					×
		CY,/PSWL.bit	CY ← CY ∧ $\overline{PSW_L\text{.bit}}$					×
	OR1	CY,saddr.bit	CY ← CY ∨ (saddr.bit)					×
		CY,/saddr.bit	CY ← CY ∨ $\overline{\text{(saddr.bit)}}$					×
		CY,sfr.bit	CY ← CY ∨ sfr.bit					×
		CY,/sfr.bit	CY ← CY ∨ $\overline{\text{sfr.bit}}$					×
		CY,A.bit	CY ← CY ∨ A.bit					×
		CY,/A.bit	CY ← CY ∨ $\overline{\text{A.bit}}$					×
		CY,X.bit	CY ← CY ∨ X.bit					×
		CY,/X.bit	CY ← CY ∨ $\overline{\text{X.bit}}$					×
		CY,PSWH.bit	CY ← CY ∨ PSW_H.bit					×
		CY,/PSWH.bit	CY ← CY ∨ $\overline{PSW_H\text{.bit}}$					×
		CY,PSWL.bit	CY ← CY ∨ PSW_L.bit					×
		CY,/PSWL.bit	CY ← CY ∨ $\overline{PSW_L\text{.bit}}$					×

(to be continued)

*1 BCD correction *2 Data conversion

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	Flag				
				S	Z	AC	P/V	CY
Bit manipulation	**XOR1**	CY,saddr.bit	CY ← CY $\forall$ (saddr.bit)					×
		CY,sfr.bit	CY ← CY $\forall$ sfr.bit					×
		CY,A.bit	CY ← CY $\forall$ A.bit					×
		CY,X.bit	CY ← CY $\forall$ X.bit					×
		CY,PSWH.bit	CY ← CY $\forall$ PSW_H.bit					×
		CY,PSWL.bit	CY ← CY $\forall$ PSW_L.bit					×
	SET1	saddr.bit	(saddr.bit) ← 1					
		sfr.bit	sfr.bit ← 1					
		A.bit	A.bit ← 1					
		X.bit	X.bit ← 1					
		PSWH.bit	PSW_H.bit ← 1					
		PSWL.bit	PSW_L.bit ← 1	×	×	×	×	×
	CLR1	saddr.bit	(saddr.bit) ← 0					
		sfr.bit	sfr.bit ← 0					
		A.bit	A.bit ← 0					
		X.bit	X.bit ← 0					
		PSWH.bit	PSW_H.bit ← 0	×	×	×	×	×
		PSWL.bit	PSW_L.bit ← 0					
	NOT1	saddr.bit	(saddr.bit) ← $\overline{\text{(saddr.bit)}}$					
		sfr.bit	sfr.bit ← $\overline{\text{sfr.bit}}$					
		A.bit	A.bit ← $\overline{\text{A.bit}}$					
		X.bit	X.bit ← $\overline{\text{X.bit}}$					
		PSWH.bit	PSW_H.bit ← $\overline{PSW_H\text{.bit}}$					
		PSWL.bit	PSW_L.bit ← $\overline{PSW_L\text{.bit}}$	×	×	×	×	×
	SET1	CY	CY ← 1					1
	CLR1	CY	CY ← 0					0
	NOT1	CY	CY ← $\overline{\text{CY}}$					×

(to be continued)

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	Flag S Z AC P/V CY
Call/return	CALL	!addr16	(SP-1) ← $(PC+3)_H$, (SP-2) ← $(PC+3)_L$, PC ← !addr16, SP ← SP-2	
		rp1	(SP-1) ← $(PC+2)_H$, (SP-2) ← $(PC+2)_L$, PC_H←$rp1_H$, PC_L←$rp1_L$, SP←SP-2	
		[rp1]	(SP-1) ← $(PC+2)_H$, (SP-2) ← $(PC+2)_L$, PC_H ← (rp1+1), PC_L ← (rp1), SP ← SP-2	
	CALLF	!addr11	(SP-1) ← $(PC+2)_H$, (SP-2) ← $(PC+2)_L$, PC_{15-11} ← 00001, PC_{10-0} ← !addr11, SP ← SP-2	
	CALLT	[addr5]	(SP-1) ← $(PC+1)_H$, (SP-2) ← $(PC+1)_L$, PC_H←(TPF,00000000,addr5+1), PC_L←(TPF,00000000,addr5), SP ← SP-2	
	BRK		(SP-1) ← PSW_H, (SP-2) ← PSW_L, (SP-3) ← $(PC+1)_H$, (SP-4) ← $(PC+1)_L$, PC_L ← (003EH), PC_H ← (003FH), SP ← SP-4, IE ← 0	
	RET		PC_L ← (SP), PC_H ← (SP+1), SP ← SP+2	
	RETB		PC_L ← (SP), PC_H ← (SP+1), PSW_L←(SP+2), PSW_H←(SP+3) SP ← SP+4	R R R R R
	RETI		PC_L ← (SP), PC_H ← (SP+1), PSW_L←(SP+2), PSW_H←(SP+3) SP ← SP+4	R R R R R

(to be continued)

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	S	Z	AC	P/V	CY
Stack manipulation	PUSH	sfrp	(SP-1)←sfr_H,(SP-2)←sfr_L SP ← SP-2					
		post	{(SP-1) ← $post_H$,(SP-2) ← $post_L$,SP←SP-2}×n times(*1)					
		PSW	(SP-1) ← PSW_H,(SP-2)←PSW_L SP ← SP-2					
	PUSHU	post	{(UP-1) ← $post_H$,(UP-2) ← $post_L$,UP←UP-2}×n times(*1)					
	POP	sfrp	sfr_L ← (SP),sfr_H ← (SP+1) SP ← SP+2					
		post	{$post_L$←(SP),$post_H$←(SP+1), SP←SP+2}×n times(*1)					
		PSW	PSW_L←(SP),PSW_H ←(SP+1), SP ← SP+2	R	R	R	R	R
	POPU	post	{$post_L$←(UP),$post_H$←(UP+1), UP←UP+2}×n times(*1)					
	MOVW	SP,#word	SP ← word					
		SP,AX	SP ← AX					
		AX,SP	AX ← SP					
	INCW	SP	SP ← SP+1					
	DECW	SP	SP ← SP-1					
Special	CHKL	sfr	(Pin level)∀(signal level before buffer output)	×	×		P	
	CHKLA	sfr	A←{(Pin level)∀(signal level before buffer output)}	×	×		P	
(*2)	BR	!addr16	PC ← !addr16					
		rp1	PC_H ← $rp1_H$,PC_L ← $rp1_L$					
		[rp1]	PC_H ← (rp1+1),PC_L ← (rp1)					
		$addr16	PC ← $addr16					

(to be continued)

*1 n indicates the number of registers specified in post.

*2 Unconditional branch

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	Flag S	Z	AC	P/V	CY
Conditional branch	BC BL	$addr16	PC ← $addr16 if CY=1					
	BNC BNL	$addr16	PC ← $addr16 if CY=0					
	BZ BE	$addr16	PC ← $addr16 if Z=1					
	BNZ BNE	$addr16	PC ← $addr16 if Z=0					
	BV BPE	$addr16	PC ← $addr16 if P/V=1					
	BNV BPO	$addr16	PC ← $addr16 if P/V=0					
	BN	$addr16	PC ← $addr16 if S=1					
	BP	$addr16	PC ← $addr16 if S=0					
	BGT	$addr16	PC←$addr16 if (P/V∀S)∨Z=0					
	BGE	$addr16	PC ← $addr16 if P/V ∀ S=0					
	BLT	$addr16	PC ← $addr16 if P/V ∀ S=1					
	BLE	$addr16	PC←$addr16 if (P/V∀S)∨Z=1					
	BH	$addr16	PC←$addr16 if Z∨CY=0					
	BNH	$addr16	PC←$addr16 if Z∨CY=1					
	BT	saddr.bit,$addr16	PC←$addr16 if (saddr.bit)=1					
		sfr.bit,$addr16	PC←$addr16 if sfr.bit=1					
		A.bit,$addr16	PC ← $addr16 if A.bit=1					
		X.bit,$addr16	PC ← $addr16 if X.bit=1					
		PSWH.bit,$addr16	PC ← $addr16 if PSWH.bit=1					
		PSWL.bit,$addr16	PC ← $addr16 if PSWL.bit=1					
	BF	saddr.bit,$addr16	PC←$addr16 if (saddr.bit)=0					
		sfr.bit,$addr16	PC ← $addr16 if sfr.bit=0					
		A.bit,$addr16	PC ← $addr16 if A.bit=0					
		X.bit,$addr16	PC ← $addr16 if X.bit=0					
		PSWH.bit,$addr16	PC ← $addr16 if PSWH.bit=0					
		PSWL.bit,$addr16	PC ← $addr16 if PSWL.bit=0					
	BTCLR	saddr.bit,$addr16	PC←$addr16 if (saddr.bit)=1 then reset (saddr.bit)					
		sfr.bit,$addr16	PC ← $addr16 if sfr.bit=1 then reset sfr.bit					
		A.bit,$addr16	PC ← $addr16 if A.bit=1 then reset A.bit					
		X.bit,$addr16	PC ← $addr16 if X.bit=1 then reset X.bit					
		PSWH.bit,$addr16	PC ← $addr16 if PSWH.bit=1 then reset PSWH.bit					
		PSWL.bit,$addr16	PC ← $addr16 if PSWL.bit=1 then reset PSWL.bit	×	×	×	×	×

(to be continued)

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	Flag S	Z	AC	P/V	CY
Conditional branch	BFSET	saddr.bit,$addr16	PC←$addr16 if (saddr.bit)=0 then set (saddr.bit)					
		sfr.bit,$addr16	PC ← $addr16 if sfr.bit=0 then set sfr.bit					
		A.bit,$addr16	PC ← $addr16 if A.bit=0 then set A.bit					
		X.bit,$addr16	PC ← $addr16 if X.bit=0 then set X.bit					
		PSWH.bit,$addr16	PC ← $addr16 if PSW_H.bit=0 then set PSW_H.bit					
		PSWL.bit,$addr16	PC ← $addr16 if PSW_L.bit=0 then set PSW_L.bit	×	×	×	×	×
	DBNZ	r2,$addr16	r2 ← r2-1, then P0 ← $addr16 if r2≠0					
		saddr,$addr16	(saddr) ← (saddr)-1, then PC ← $addr16 if(saddr) ≠0					
Context switching	BRKCS	RBn	PC_H←→R5, PC_L←→R4, R7←PSW_H, R6←PSW_L, RBS2-0←n, RSS←0, IE←0					
	RETCS	!addr16	PC_H←R5, PC_L←R4, R5, R4←!addr16, PSW_H←R7, PSW_L←R6	R	R	R	R	R
	RETCSB	!addr16	PC_H←R5, PC_L←R4, R5, R4←!addr16, PSW_H←R7, PSW_L←R6	R	R	R	R	R

(to be continued)

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	S	Z	AC	P/V	CY
String	MOVM	[DE+],A	(DE+) ← A, C ← C-1 End if C=0					
		[DE-],A	(DE-) ← A, C ← C-1 End if C=0					
	MOVBK	[DE+],[HL+]	(DE+) ← (HL+), C ← C-1 End if C=0					
		[DE-],[HL-]	(DE-) ← (HL-), C ← C-1 End if C=0					
	XCHM	[DE+],A	(DE+)←→A, C ← C-1 End if C=0					
		[DE-],A	(DE-) ←→ A, C ← C-1 End if C=0					
	XCHBK	[DE+],[HL+]	(DE+) ←→ (HL+), C ← C-1End if C=0					
		[DE-],[HL-]	(DE-)←→(HL-), C ← C-1 End if C=0					
	CMPME	[DE+],A	(DE+)-A, C ← C-1 End if C=0 or Z=0	×	×	×	V	×
		[DE-],A	(DE-)-A, C ← C-1 End if C=0 or Z=0	×	×	×	V	×
	CMPBKE	[DE+],[HL+]	(DE+)-(HL+), C ← C-1 End if C=0 or Z=0	×	×	×	V	×
		[DE-],[HL-]	(DE-)-(HL-), C ← C-1 End if C=0 or Z=0	×	×	×	V	×
	CMPMNE	[DE+],A	(DE+)-A, C ← C-1 End if C=0 or Z=1	×	×	×	V	×
		[DE-],A	(DE-)-A, C ← C-1 End if C=0 or Z=1	×	×	×	V	×
	CMPBKNE	[DE+],[HL+]	(DE+)-(HL+), C ← C-1 End if C=0 or Z=1	×	×	×	V	×
		[DE-],[HL-]	(DE-)-(HL-), C ← C-1 End if C=0 or Z=1	×	×	×	V	×
	CMPMC	[DE+],A	(DE+)-A, C ← C-1 End if C=0 or CY=0	×	×	×	V	×
		[DE-],A	(DE-)-A, C ← C-1 End if C=0 or CY=0	×	×	×	V	×
	CMPBKC	[DE+],[HL+]	(DE+)-(HL+), C ← C-1 End if C=0 or CY=0	×	×	×	V	×
		[DE-],[HL-]	(DE-)-(HL-), C ← C-1 End if C=0 or CY=0	×	×	×	V	×

(to be continued)

(Cont'd)

Instruction type	Mnemonic	Operand	Operation	Flag				
				S	Z	AC	P/V	CY
String	**CMPMNC**	[DE+],A	(DE+)-A, C ← C-1 End if C=0 or CY=1	×	×	×	V	×
		[DE-],A	(DE-)-A, C ← C-1 End if C=0 or CY=1	×	×	×	V	×
	CMPBKNC	[DE+],[HL+]	(DE+)-(HL+), C ← C-1 End if C=0 or CY=1	×	×	×	V	×
		[DE-],[HL-]	(DE-)-(HL-), C ← C-1 End if C=0 or CY=1	×	×	×	V	×
CPU control	**MOV**	STBC,#byte	STBC ← byte(*)					
		WDM,#byte	WDM ← byte(*)					
	SWRS		RSS ← $\overline{\text{RSS}}$					
	SEL	RBn	RSS ← 0,RBS2-0 ← n					
		RBn,ALT	RSS ← 1,RBS2-0 ← n					
	NOP		No Operation					
	EI		IE ← 1(Enable Interrup)					
	DI		IE ← 0(Disable Interru)					

* An op-code trap interrupt occurs if an invalid op-code is specified in an STBC or WDM register manipulation instruction.

Trap operation:

$(SP - 1) \leftarrow PSW_H$, $(SP - 2) \leftarrow PSW_L$,
$(SP - 3) \leftarrow (PC - 4)_H$, $(SP - 4) \leftarrow (PC - 4)_L$,
$PC_L \leftarrow (003CH)$, $PC_H \leftarrow (003DH)$,
$SP \leftarrow SP - 4$, $IE \leftarrow 0$

10.2 Special Function Registers (SFR) of the Target Device

Special function registers (SFR) of the target device (uPD78330, uPD78334, or uPD78P334) are classified into two types according to the bit size of their registers.

- o 8-bit special function register
- o 16-bit special function register

Table 10-1 8-bit Special Function Registers (SFRs)

Address	SFR	Address	SFR	Address	SFR	Address	SFR
FF00	P0	FF20	PM0	FF40	PMC0	FF60	RTP
FF01	P1	FF21	PM1	FF41	PMC1	FF61	RTPR
FF02	P2	FF22		FF42		FF62	PRDC
FF03	P3	FF23	PM3	FF43	PMC3	FF63	RTPS
FF04	P4	FF24		FF44		FF64	PWMC
FF05	P5	FF25	PM5	FF45		FF65	
FF06		FF26		FF46		FF66	PWM0
FF07	P7	FF27		FF47		FF67	
FF08	P8	FF28		FF48		FF68	ADM
FF09	P9	FF29	PM9	FF49		FF69	
FF0A	TLA	FF2A		FF4A		FF6A	
FF0B		FF2B		FF4B		FF6B	
FF0C		FF2C		FF4C		FF6C	
FF0D		FF2D		FF4D		FF6D	
FF0E		FF2E		FF4E		FF6E	PWM1
FF0F		FF2F		FF4F		FF6F	
FF10		FF30		FF50		FF70	
FF11		FF31		FF51		FF71	
FF12		FF32		FF52		FF72	
FF13		FF33		FF53		FF73	
FF14		FF34		FF54		FF74	
FF15		FF35		FF55		FF75	
FF16		FF36		FF56		FF76	
FF17		FF37		FF57		FF77	
FF18		FF38		FF58		FF78	
FF19		FF39		FF59		FF79	
FF1A		FF3A		FF5A		FF7A	
FF1B		FF3B		FF5B		FF7B	
FF1C		FF3C		FF5C		FF7C	
FF1D		FF3D		FF5D		FF7D	
FF1E		FF3E		FF5E		FF7E	
FF1F		FF3F		FF5F		FF7F	

(to be continued)

Table 10-1 8-bit Special Function Registers (SFRs) (Cont'd)

Address	SFR	Address	SFR	Address	SFR	Address	SFR
FF80	CSIM	FFA0		FFC0	STBC	FFE0	IF0L
FF81		FFA1	ADCR0H	FFC1	CCW	FFE1	IF0H
FF82	SBIC	FFA2		FFC2	WDM	FFE2	IF1L
FF83		FFA3	ADCR1H	FFC3		FFE3	
FF84		FFA4		FFC4	MM	FFE4	MK0L
FF85		FFA5	ADCR2H	FFC5		FFE5	MK0H
FF86	SIO	FFA6		FFC6	PWC	FFE6	MK1L
FF87		FFA7	ADCR3H	FFC7		FFE7	
FF88	ASIM	FFA8		FFC8		FFE8	PB0L
FF89		FFA9	ADCR4H	FFC9	FCC	FFE9	PB0H
FF8A	ASIS	FFAA		FFCA		FFEA	PB1L
FF8B		FFAB	ADCR5H	FFCB		FFEB	
FF8C	RxB	FFAC		FFCC		FFEC	ISM0L
FF8D		FFAD	ADCR6H	FFCD		FFED	ISM0H
FF8E	TxS	FFAE		FFCE		FFEE	ISM1L
FF8F		FFAF	ADCR7H	FFCF		FFEF	
FF90		FFB0	TMC0	FFD0	EXTSFR0	FFF0	CSE0L
FF91		FFB1	BRGM	FFD1	EXTSFR1	FFF1	CSE0H
FF92		FFB2	TMC1	FFD2	EXTSFR2	FFF2	CSE1L
FF93		FFB3		FFD3	EXTSFR3	FFF3	
FF94		FFB4	TUM0	FFD4	EXTSFR4	FFF4	INTM0
FF95		FFB5	TUM1	FFD5	EXTSFR5	FFF5	INTM1
FF96		FFB6		FFD6	EXTSFR6	FFF6	
FF97		FFB7		FFD7	EXTSFR7	FFF7	
FF98		FFB8	TOC0	FFD8	EXTSFR8	FFF8	ISPR
FF99		FFB9	TOC1	FFD9	EXTSFR9	FFF9	PRSL
FF9A		FFBA		FFDA	EXTSFR10	FFFA	
FF9B		FFBB	PPOS	FFDB	EXTSFR11	FFFB	
FF9C		FFBC		FFDC	EXTSFR12	FFFC	
FF9D		FFBD		FFDD	EXTSFR13	FFFD	
FF9E		FFBE		FFDE	EXTSFR14	FFFE	
FF9F		FFBF		FFDF	EXTSFR15	FFFF	

Table 10-2 16-bit Special Function Registers (SFRs)

Address	SFR	Address	SFR	Address	SFR	Address	SFR
FF00		FF20		FF40		FF60	
FF01		FF21		FF41		FF61	
FF02		FF22		FF42		FF62	
FF03		FF23		FF43		FF63	
FF04		FF24		FF44		FF64	
FF05		FF25		FF45		FF65	
FF06		FF26		FF46		FF66	
FF07		FF27		FF47		FF67	
FF08		FF28		FF48		FF68	
FF09		FF29		FF49		FF69	
FF0A		FF2A	TM0	FF4A		FF6A	
FF0B		FF2B		FF4B		FF6B	
FF0C	TM2	FF2C	TM1	FF4C	BRG	FF6C	
FF0D		FF2D		FF4D		FF6D	
FF0E		FF2E	TM3	FF4E		FF6E	
FF0F		FF2F		FF4F		FF6F	
FF10	CT00	FF30		FF50		FF70	CM11
FF11		FF31		FF51		FF71	
FF12	CT01	FF32		FF52		FF72	CM12
FF13		FF33		FF53		FF73	
FF14	CT02	FF34		FF54		FF74	CM20
FF15		FF35		FF55		FF75	
FF16		FF36		FF56		FF76	CM21
FF17		FF37		FF57		FF77	
FF18		FF38		FF58		FF78	CM30
FF19		FF39		FF59		FF79	
FF1A		FF3A		FF5A		FF7A	
FF1B		FF3B		FF5B		FF7B	
FF1C	CT10	FF3C		FF5C		FF7C	
FF1D		FF3D		FF5D		FF7D	
FF1E		FF3E		FF5E		FF7E	
FF1F		FF3F		FF5F		FF7F	

(to be continued)

Table 10-2 16-bit Special Function Registers (SFRs) (Cont'd)

Address	SFR	Address	SFR	Address	SFR	Address	SFR
FF80		FFA0	ADCR0	FFC0		FFE0	IF0
FF81		FFA1		FFC1		FFE1	
FF82		FFA2	ADCR1	FFC2		FFE2	IF1
FF83		FFA3		FFC3		FFE3	
FF84		FFA4	ADCR2	FFC4		FFE4	MK0
FF85		FFA5		FFC5		FFE5	
FF86		FFA6	ADCR3	FFC6		FFE6	MK1
FF87		FFA7		FFC7		FFE7	
FF88		FFA8	ADCR4	FFC8		FFE8	PR0
FF89		FFA9		FFC9		FFE9	
FF8A		FFAA	ADCR5	FFCA		FFEA	PB1
FF8B		FFAB		FFCB		FFEB	
FF8C		FFAC	ADCR6	FFCC		FFEC	ISM0
FF8D		FFAD		FFCD		FFED	
FF8E		FFAE	ADCR7	FFCE		FFEE	ISM1
FF8F		FFAF		FFCF		FFEF	
FF90	CMX0	FFB0		FFD0		FFF0	CSE0
FF91		FFB1		FFD1		FFF1	
FF92	CM01R	FFB2		FFD2		FFF2	CSE1
FF93		FFB3		FFD3		FFF3	
FF94	CM02R	FFB4		FFD4		FFF4	
FF95		FFB5		FFD5		FFF5	
FF96	CM03R	FFB6		FFD6		FFF6	
FF97		FFB7		FFD7		FFF7	
FF98	CM04R	FFB8		FFD8		FFF8	
FF99		FFB9		FFD9		FFF9	
FF9A	CC00R	FFBA		FFDA		FFFA	
FF9B		FFBB		FFDB		FFFB	
FF9C	CC01R	FFBC		FFDC		FFFC	
FF9D		FFBD		FFDD		FFFD	
FF9E		FFBE		FFDE		FFFE	
FF9F		FFBF		FFDF		FFFF	

10.3 Online Assembler Specification

This specification applies to ASM commands.

The specification covers 14 items:

① Character set
② Symbol definition
③ Comment line
④ Numeric representation coded in the operand field
⑤ Symbolic representation coded in the operand field
⑥ Expression representation coded in the operand field
⑦ Pseudo instructions
⑧ Rule for generating instruction codes
⑨ Checking sfr manipulation instructions for errors
⑩ Checking saddr space manipulation instructions for an error
⑪ Omitting the addressing mode
⑫ Coding operands in PUSHR, POPR, PUSHU, or POPU
⑬ Error messages
⑭ List of reserved words

(1) Character set

Characters available for the assembler are listed below:

A to Z, a to z, @, ?, _, 0 to 9, +, -, *, /, $, !, [,], #, (,), ;(semicolon), .(period), ,(comma), ¥, and \

The yen sign (¥)[Note 1] and the backslash ( )[Note 2] are available only for symbolic representation.

Notes 1. To be used when the IE-78330-R is connected to the PC-9800 series.

2. To be used when the IE-78330-R is connected to the IBM PC series.

Lowercase letters are treated as uppercase letters.

(2) Symbol definition

The assembler does not allow symbols such as labels to be defined. Instead of numeric values, however, defined symbols can be used.

(3) Comment line

The assembler assumes the part from a semicolon (;) to <cr> as a comment.

(4) Numeric representation coded in the operand field

Conforms to the coding conventions for numeric values to be used for command input.

See Chapter 6 for details.

(5) Symbolic representation coded in the operand field

Conforms to the coding conventions for symbols to be used for command input.

See Chapter 6 for details.

(6) Expression representation coded in the operand field

Conforms to the coding conventions for expressions to be used for command input.

See Chapter 6 for details.

(7) Pseudo instructions

The assembler supports pseudo instructions:

(a) ORG addr16

The ORG instruction places the next instruction at addr16.

If addr16 is less than the address of the current location, "Caution" is displayed.

If addr16 is greater than 0FE7FH, "Error" is displayed.

(b) DB byte,...byte

The DB instruction places the byte data in the current location.

If multiple byte data items are separated by a comma, they are placed in sequence in the current and subsequent locations.

If data are located beyond address 0FE7FH, or if an operand field contains word data, "Error" is displayed.

If an error occurs, all the data are invalidated.

(c) DW word,...word

The DW instruction places the low-order byte of the word data in the current location and the high-order byte in the next location.

If multiple word data items are separated by a comma, they are located in sequence in the current and subsequent locations according to the above rule.

If data are located beyond address 0FE7FH, "Error" is displayed.

If an error occurs, all the data are invalidated.

(d) DS word

The DS instruction places the next instruction in the location at a displacement of the value in the word from the current location. If the value of (current location + word) exceeds 0FE7FH, "Error" is displayed. If the value of (current location + word) exceeds 0FFFFH, the digits higher than the most significant digit permitted are truncated, and "Caution" is displayed.

(e) END

The END instruction terminates ASM command execution.

(8) Rule for generating instruction codes

The uPD78330, uPD78334, and uPD78P334 can perform two types of addressing for 0FE20H to 0FFFFH memory space:

(a) Short direct addressing

The online assembler of the IE-78330-R applies the short direct addressing to the 0FE20H to 0FF1FH memory space.

(b) Special function register (SFR) addressing

The online assembler of the IE-78330-R applies the special function register (SFR) addressing to the 0FF20H to 0FFFFH memory space.

(9) Checking sfr manipulation instructions for errors

Instructions for manipulating the sfr (special function register) space are checked for errors according to the following address representation:

(a) When an address is represented by an sfr reserved word

Checking sfr or sfrp

The 16-bit manipulation instruction for sfr or the 8-bit manipulation instruction for sfrp causes "Warning" to be displayed.

Checking the attribute of read only or write only

The write instruction for sfr with R/O or the read instruction for sfr with W/O causes "Warning" to be displayed.

(b) When an address is represented by a numeric value, symbol, or expression

Checking whether sfr is found

If sfr is not found, "Warning" is displayed.

Checking sfr or sfrp

The 16-bit manipulation instruction for sfr or the 8-bit manipulation instruction for sfrp causes "Warning" to be displayed.

Checking the attribute of read only or write only

The write instruction for sfr with R/O or the read instruction for sfr with W/O causes "Warning" to be displayed.

(10) Checking saddr space manipulation instructions for an error

The assembler checks whether the address is even or odd during 16-bit manipulation by the saddr space manipulation instructions.

The 16-bit manipulation instruction for an odd address causes "Warning" to be displayed.

(11) Omitting the addressing mode

When an addressing mode (absolute or relative), conforming to branch instructions is determined clearly according to an instruction, the addressing mode can be omitted.

If no addressing mode is specified in the branch instruction (BR), "Caution" is displayed, and the shortest code is generated according to the value of addr16.

The following shows the rules for generating the shortest code:

```
                 -128 ≦ (addr16 - $ + 2) ≦ +127      Instruction generated
                                                     BM $addr16
BR addr16
                                                     BM !addr16
                 (addr16 - $ + 2) < -128
                            or
                 +127 < (addr16 - $ + 2)
```

If location counter $ is used, the addressing mode must not be omitted.

Example 1: Omit the addressing mode.

Normal commands		Abbreviations
CALL !addr16	→	CALL addr16
CALL $addr16	→	CALL addr16

Example 2: Addressing using the location counter

```
BC $$±n ← Indicates a branch to address ($ ±n)
   ||
   ||__ $ of the addressing mode
   |
   |___ $ of the location counter
```

(12) Coding operands in PUSHR, POPR, PUSHU, or POPU

The coding sequence of operands is not determined.

Coding the same operand more than once causes "Error" to be displayed.

(13) Error messages

The error messages displayed by this assembler are classified into three types:

(a) "Error"

This message is displayed if no object code can be generated or if it is clear that an error occurred.

Example: Instruction specification causing "Error" to be displayed

```
ORG OFFOOH   ← No program can be placed at
               address OFFOOH.
DB  OFFOOH   ← No word data can be specified.
MOV J,#byte  ← No symbol is registered yet.
BR  $        ← No address is specified yet.
```

(b) "Warning"

This message is displayed if normal operation cannot be attained although an object code is generated.

Example: Instruction specification causing "Warning" to appear

```
MOV  sfrp,#byte       ← 8-bit manipulation for sfrp
MOVW sfr,#word        ← 16-bit manipulation for sfr
MOVW odd saddr,#word  ← 16-bit manipulation for odd saddr
BR   !0FF00H          ← A branch to sfr space
```

(c) "Caution"

This message is displayed if an address is automatically generated.

Example: If an address is automatically generated

```
ORG $-100H
BR  100H
```

(14) List of reserved words

The reserved words of the online assembler in the IE-78330-R are listed below:

A
AC
ADCR0
ADCR0H
ADCR1
ADCR1H
ADCR2
ADCR2H
ADCR3
ADCR3H
ADCR4
ADCR4H
ADCR5
ADCR5H
ADCR6
ADCR6H
ADCR7
ADCR7H
ADD
ADDC
ADDW
ADJBA
ADJBS
ADM
ALT
AND
AND1
ASIM
ASIS
AX
B
BC
BE
BF
BFSET
BGE
BGT
BH
BL
BLE
BLT
BN
BNC
BNE
BNH
BNL
BNV
BNZ
BP
BPE
BPO
BR
BRG
BRGM
BRK
BRKCS
BT
BTCLR
BV
BZ
C
CALL
CALLF
CALLT
CC00R
CC01R
CCW
CHKL
CHKLA
CLR1
CM01R
CM02R
CM03R
CM04R
CM11
CM12
CM20
CM21
CM30
CMP
CMPBKC
CMPBKE
CMPBKNC
CMPBKNE
CMPMC
CMPME
CMPMNC
CMPMNE
CMPW
CMX0
CSE0
CSE0H
CSE0L
CSE1
CSE1L
CSIM
CT00
CT01
CT02
CT10
CVTBW
CY
D
DB
DBNZ

DE	INC	P4
DEC	INCW	P5
DECW	INTM0	P7
DI	INTM1	P8
DIVUW	ISM0	P9
DIVUX	ISM0H	PB0
DS	ISM0L	PB0H
DW	ISM1	PB0L
E	ISM1L	PB1
EI	ISPR	PB1L
END	L	PC
EXTSFR1	LT	PM0
EXTSFR2	MK0	PM1
EXTSFR3	MK0H	PM3
EXTSFR4	MK0L	PM5
EXTSFR5	MK1	PM9
EXTSFR6	MK1L	PMC0
EXTSFR7	MM	PMC1
EXTSFR8	MOV	PMC3
EXTSFR9	MOV1	POP
EXTSFR10	MOVBK	POPU
EXTSFR11	MOVM	PPOS
EXTSFR12	MOVW	PRDC
EXTSFR13	MULU	PRSL
EXTSFR14	MULUW	PSW
EXTSFR15	MULW	PSWH
FCC	NOP	PSWL
H	NOT1	PUSH
HL	OR	PUSHU
IE	OR1	PWC
IF0	ORG	PWM0
IF0H	P0	PWM1
IF0L	P1	PWMC
IF1	P2	R0
IF1L	P3	R1

R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
RB0
RB1
RB2
RB3
RB4
RB5
RB6
RB7
RBS0
RBS1
RBS2
RET
RETB
RETCS
RETCSB
RETI
ROL
ROL4
ROLC
ROR
ROR4

RORC
RP0
RP1
RP2
RP3
RP4
RP5
RP6
RP7
RSS
RTP
RTPR
RTPS
RXB
S
SBIC
SEL
SET1
SHL
SHLW
SHR
SHRW
SIO
SP
STBC
SUB
SUBC
SUBW
SWRS
TLA
TM0
TM1
TM2
TM3
TMC0

TMC1
TOC0
TOC1
TUM0
TUM1
TXS
UF
UP
VP
WDM
X
XCH
XCHBK
XCHM
XCHW
XOR
XOR1
Z

10.4 Disassembler Specification

This specification applies to the disassemble display commands such as the DAS and TRD I commands.

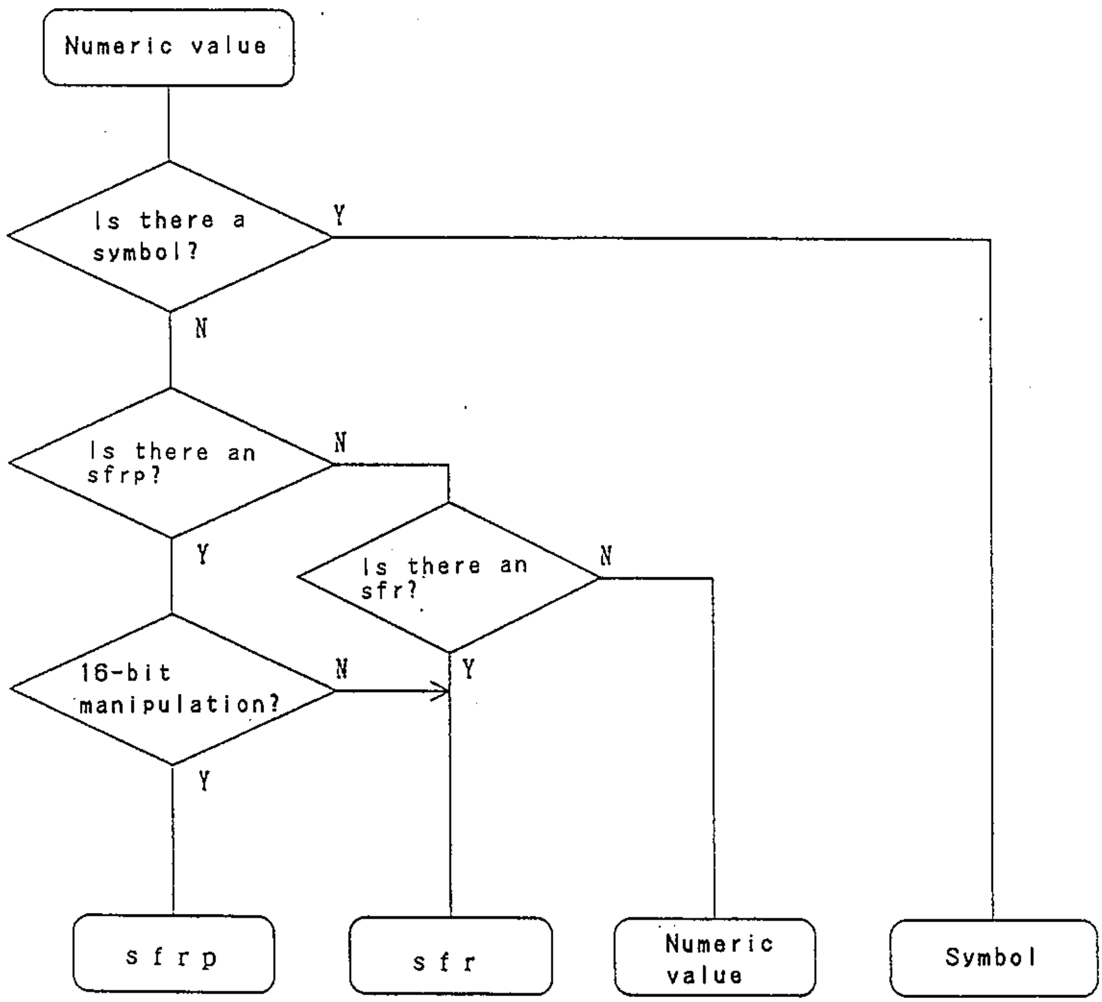
The specification covers seven items:

① Numeric display of operands
② Symbolic display of operands
③ Label line display
④ Branch instruction display
⑤ Checking sfr and saddr manipulation instructions
⑥ Checking MOV STBC,#byte or MOV WDM,#byte instruction
⑦ Error messages

(1) Numeric display of operands

If a registered symbol, sfr, and sfrp correspond to numeric values, operands are displayed in symbol name.

The following flowchart shows conversion from numeric values to symbols.



If there are no registered symbol, sfr, or sfrp that correspond to numeric values, operands are displayed in hexadecimal with radix H.

The numeric value always begins with a number from 0 to 9.

(2) Symbolic display of operands

Symbols are displayed without module names.

(3) Label line display

When there is a symbol which corresponds to the current location, the symbol is displayed as a label line.

Example: Display a label line.

```
        (Note)
module¥local symbol:     ← A local symbol always contains one
                           additional colon (:).
PUBLIC¥public symbol::   ← A public symbol always contains two
                           additional colons (::).
```

Note: A backslash (\) is used instead of a yen sign (¥) when the IBM PC series is used as the host machine.

(4) Branch instruction display

Two mnemonics are assigned to the same object code for some branch instructions.

In this case, the mnemonic used more frequently is displayed as follows.

BNZ	BNZ	BZ	BZ
BNE		BE	
BNC	BNC	BC	BC
BNL		BL	
BNV	BNV	BV	BV
BPO		BPE	

(5) Checking sfr and saddr manipulation instructions

The disassembler checks manipulation instructions for sfr and saddr spaces. When the instruction performs an invalid operation, the assembler displays an error message.

One of the following operations causes an error message to appear.

(a) sfr

- o Access to non-existing sfr or sfrp
- o Writing to sfr having R/O
- o Reading from sfr having W/O
- o 16-bit manipulation for the space other than sfrp

(b) saddr

- o 16-bit manipulation for an odd address

(6) Checking the MOV STBC,#byte or MOV WDM,#byte instruction

The disassembler displays an error message when the complement of the byte in the operand of the MOV STBC,#byte or MOV WDM,#byte instruction does not match the byte.

In this case, the byte is displayed in the operand of the mnemonic.

The execution of the MOV sfr,#byte instruction for the STBC or WDM register causes an error message to be displayed.

The STBC or WDM register can be accessed only with a target device control instruction.

(7) Error message

Error messages are classified into two types:

(a) ???

If the disassembler cannot disassemble an object code, ??? is displayed in the mnemonic field.

While an instruction code of two or more bytes is being disassembled, if decoding the second and following bytes shows an error, only the first byte is displayed as erroneous.

The disassembler then continues disassembling, regarding the second byte as the first byte of the next instruction code.

Example: The disassembler cannot disassemble an object code.

```
ADDR    OBJECT              MNEMONIC
                             (Note)
                        PUBLIC¥START::
0100    0B FC 00 FE     MOVW   SP,#STACK
0104    01              ???                  <- Not disassembled
0105    64 00 10        MOVW   RP2,#WORK1
0108    25              ???                  <- Not disassembled
0109    59              MOV    [HL+],A
```

Note: A backslash (\) is used instead of a yen sign (¥) when the IBM PC series is used as the host machine.

(b) ?

If the disassembler cannot disassemble an object code completely, ? is displayed just before the disassembled mnemonic.

Example: The disassembler cannot disassemble an object code completely.

```
ADDR    OBJECT              MNEMONIC

0105    0C 21 34 12     ? MOVW  0FE21H,#1234H <- 16-bit manipula-
                                                 tion at odd saddr
0113    3A 02 20        ? MOV   P2,#20H       <- Writing to sfr
                                                 with R/O
0120    10 43           ? MOV   A,PMC3        <- Reading from sfr
                                                 with W/O
0127    2C 00 FF        ? BR    !0FF00H       <- A branch to sfr
                                                 space
```

CHAPTER 11 OPERATING THE PROM PROGRAMMER (PG-1500 OR PG-2000)

This chapter explains how to control of the PROM programmer (PG-1500 or PG-2000) remotely when it is connected to the IE-78330-R.

For connecting the PROM programmer to the IE-78330-R, refer to Chapter 5 in the IE-78330-R In-circuit Emulator: Hardware.

Refer to the user's manual for the details of operating the PG-1500 or PG-2000.

Conventions

o ______: Indicates that the underlined item needs to be entered from the keyboard.

o <cr>: Indicates that the return key (CR (0DH)) needs to be pressed.

o <ESC>: Indicates that the escape key needs to be pressed.

o ^: Indicates that the character on the right side of the caret (^) needs to be pressed while the control key is held down.

o The screen display and input examples in this manual apply when a PC-9800 series personal computer is used as the host machine.

11.1 Starting and Terminating Remote Control of the PROM Programmer

(1) Starting remote control of the PROM programmer

Entering the PROM programmer control command (PGM command) makes the prompt of the PROM programmer appear.

If no prompt appears, check the connection between the PROM programmer and the IE-78330-R.

Example 1: Start remote control of the PG-1500.

```
brk:0>PGM <cr>
 Beginning of PGM mode ← Remote control start message of PROM
                         programmer
PG>                    ← Prompt output by PG-1500
```

Example 2: Start remote control of the PG-2000.

```
brk:0>PGM <cr>
 Beginning of PGM mode ← Remote control start message of PROM
                         programmer
*                      ← Prompt output by PG-2000
```

(2) Ending remote control of the PROM programmer

Entering ^Z from the keyboard terminates remote control of the PROM programmer.

Example 1: Terminate remote control of the PG-1500.

```
PG>^Z                          ← Enter ^Z.
 Exit PGM mode (Y/N) Y <cr>  ← Remote control termination inquiry
                                 message of PROM programmer
 Termination of PGM mode      ← Remote control termination message
                                 of PROM programmer
brk:0>
```

Example 2: Terminate remote control of the PG-2000.

```
*^Z                            ← Enter ^Z.
 Exit PGM mode (Y/N) Y <cr>  ← Remote control termination inquiry
                                 message of PROM programmer
 Termination of PGM mode      ← Remote control termination message
                                 of PROM programmer
brk:0>
```

11.2 PROM Programmer Commands

This section explains the commands of the PROM programmer (PG-1500 or PG-2000), and uploading and downloading an object code.

(1) PROM programmer's commands

The following tables list commands used in the PROM programmers (PG-1500 and PG-2000).

(a) PG-1500 commands

Table 11-1 PG-1500 Commands

Command	Format	Function
RR	PG>RR P_S_ADR,R_E_ADR,PG_S_ADR,CONV <cr>	Reads the contents of PROM.
RS	PG>RS sub <cr> sub=C/R/A	Selects a device.
RV	PG>RV P_S_ADR,R_E_ADR,PG_S_ADR,CONV <cr>	Compares the PROM contents with the contents of memory in PG.
RW	PG>RW PG_S_ADR,PG_E_ADR,R_S_ADR,CONV <cr>	Writes on PROM.
RZ	PG>RZ <cr>	Checks whether the contents of PROM are erased.
MC	PG>MC PG_S_ADR <cr>	Changes the contents of memory in PG.
MD	PG>MD PG_S_ADR,PG_E_ADR <cr>	Displays the contents of memory in PG.
MF	PG>MF PG_S_ADR,PG_E_ADR,INT_DATA <cr>	Initializes memory in PG.
LI	PG>LI <cr>	Transfers data from IE-78230-R to PG.
SI	PG>SI PG_S_ADR,PG_E_ADR <cr>	Transfers data from PG to IE-78230-R.
??	PG>?? <cr>	Command help

Refer to the PG-1500 User's Manual.

Remark:
- . PG_S_ADR: PG start address
- . PG_E_ADR: PG end address
- . R_S_ADR: PROM start address
- . R_E_ADR: PROM end address
- . CONV: Address conversion
- . INT_DATA: Data to be initialized

(b) PG-2000 commands

Table 11-2 PG-2000 Commands

Command	Format	Function
A	*As,e,r <cr>	Sets parameters.
E	*Er <cr>	Changes data of the buffer in PG as follows: *Er <cr> r XX- YY- XX- YY- XX- <cr> Input data (YY) Currently set data (XX) Data input format . Enter data. (When non-hexadecimal data are entered, ' ?' is displayed.) . Enter spaces. (No change in data)
F	*Fr,re,d <cr>	Initializes the buffer in PG with d.
I	*I <cr>	Makes transition to the intelligent mode (mode for echo-back to the display).
O	*Or,re <cr>	Displays the contents of the buffer in PG. Display format → r, 00 00 00 00 00 00 00
R	*Rr,re <cr>	Transfers the contents of PROM to the buffer in PG.
S	*S <cr>	Selects PROM.
T	*T <cr>	Makes transition to the transient mode (mode for no echo-back to the display)

(to be continued)

Table 11-2 PG-2000 Commands (Cont'd)

Com- mand	Format	Function
V	*Vs,e,r <cr>	Compares the contents of PROM with the contents of the buffer in PG. → When they do not match, ' ?' is displayed.
W	*Ws,e,r <cr>	Writes the contents of the buffer in PG onto PROM. → Occurrence of a write error causes ' ?' to be displayed.
Y	*Y <cr>	Displays currently set parameters.
Z	*Z <cr>	Checks whether data are already written on PROM. → Only if data are already written on PROM, ' ?' appears; otherwise, nothing is displayed.
L	*Lbias <cr> partition= i,ie <cr>	Transfers data at addresses i to ie in mapped IE-78330-R memory to addresses (i + bias) to (ie + bias) in the buffer of PG.
P	*Pr,re <cr> Bias =i <cr>	Transfers data at addresses r to re in the buffer of PG to addresses (r + i) to (re + 1) in mapped IE-78330-R memory.

Refer to the PG-2000 User's Manual.

Remark:
- s: PROM start address
- e: PROM end address
- r: PG buffer start address
- re: PG buffer end address
- i: IE-78330-R start address
- ie: IE-78330-R end address
- d: data

Caution: Error conditions

s > e

r > re

i > ie

Non-hexadecimal input

(2) Uploading and downloading an object code

(a) PG-1500 and uploading the object code (executing an LI command)

The LI command is used to transfer the contents of mapped IE-78330-R memory to the buffer in the PG-1500.

If no transfer range is specified, the command transfers the contents of the whole mapped IE-78330 memory. To stop data transfer, press the ESC key.

Example: Upload the object code.

```
PG>LI <cr>
                    (Note 1) (Note 2)
  Partition = YYYY,ZZZZ <cr>
PG>
```

Notes 1. Enter the transfer start address of IE-78330-R memory in YYYY.

2. Enter the transfer end address of IE-78330-R memory in ZZZZ.

(b) PG-1500 and downloading the object code (executing an SI command)

The SI command is used to transfer the contents of the buffer in the PG-1500 to mapped IE-78330-R memory.

The load bias of the IE-78330-R cannot be omitted.

To stop data transfer, press the ESC key.

Example: Download the object code.

```
     (Note 1)(Note 2)
PG>SI XXXX,YYYY <cr>
                (Note 3)
     Bias = ZZZZ <cr>
     complete
PG>
```

Notes 1. Enter the transfer start address of PG buffer in XXXX.

2. Enter the transfer end address of PG buffer in YYYY.

3. Enter the load bias of the IE-78330-R in ZZZZ.

(c) PG-2000 and uploading the object code (executing an L command)

The L command is used to transfer the contents of mapped IE-78330-R memory to the buffer in the PG-2000.

If no transfer range is specified, the command transfers the contents of the whole mapped IE-78330 memory. To stop data transfer, press the ESC key.

Example: Upload the object code.

```
   (Note 1)
*LXXX <cr>
               (Note 2)(Note 3)
 Partition = YYYY,ZZZZ <cr>
*
```

Notes 1. Enter the load bias of the PG in XXXX.

2. Enter the transfer start address of IE-78330-R memory in YYYY.

3. Enter the transfer end address of IE-78330-R memory in ZZZZ.

(d) PG-2000 and downloading the object code (executing a P command)

The P command is used to transfer the contents of the buffer in the PG-2000 to mapped IE-78330-R memory.

The load bias of the IE-78330-R cannot be omitted.

To stop data transfer, press the ESC key.

Example: Download the object code.

```
(Note 1)(Note 2)
*PXXXX,YYYY <cr>
          (Note 3)
 Bias = ZZZZ <cr>
 complete
*
```

Notes 1. Enter the transfer start address of PG buffer in XXXX.

2. Enter the transfer end address of PG buffer in YYYY.

3. Enter the load bias of the IE-78330-R in ZZZZ.

CHAPTER 12 CAUTIONS

This chapter describes the cautions when using the IE-78330-R or the emulated devices (uPD78330, uPD78334, and uPD78P334).

Read this chapter before developing application products.

12.1 Cautions Related to the IE-78330-R

This section collects the cautions on the use of the IE-78330-R described in this manual.

(1) Cautions in Chapter 1

1. 640K bytes or more are required for internal memory. (p.1-8)

2. Distribution of a control program on an 8-inch floppy disk (2D) has been discontinued. Understand that the control program is distributed on a 5-inch floppy disk to users using an 8-inch floppy disk when the control program is upgraded. (p.1-9)

(2) Cautions in Chapter 2

1. MS-DOS or PC DOS must be started again by resetting the host machine after the device drivers are installed. (p.2-2)

2. If N <cr> is entered in response to the above message, the system displays the same message, and the user cannot proceed to the next step. (p.2-6)

(3) Cautions in Chapter 3

1. If the internal RAM (0FE00H to 0FEFFH) is read with addressing other than short direct addressing, the trace data value is undefined. However, instructions (target program) are executed normally. (p.3-42)

2. Cautions related to the trigger signal external output function (p.3-47)

① The IE-78330-R has eight external sense clips 1 to 8 to trace input data and detect events. Normally, the eight external sense clips are set as input lines. External sense clip 1 can also be set as a trigger signal output with the OUT ON command.

② When external sense clip 1 is set as a trigger signal output with the OUT ON command, NEVER connect the sense clip to the signal output line of the target system. Otherwise, the target system and IE-78330-R may be damaged.

3. To terminate the IE-78330-R, be sure to use this function. (p.3-55)

4. After memory or register manipulation, the emulation CPU restarts operation. However, the CPU performs the operation after executing the above command stops the program temporarily (that is, does not enter the real-time execution mode). (p.3-56)

5. When a HALT or STOP instruction is executed in nonreal-time execution (RUN T command), the emulation CPU does not enter the standby mode. (p.3-57)

(4) Caution in Chapter 6

Decimal X representation is not allowed. (p.6-4)

(5) Caution in Chapter 7

The command list is provided based on the single-line command input format. (p.7-10)

(6) Cautions in Chapter 8

1. The ASM command is available only when prompt brk:0 appears. (p.8-1)

2. When a command to change memory contents is entered during trace (trc:0> mode) or emulation (emu:0> mode), execution of the emulation CPU is temporarily suspended. (p.8-48)

3. Cautions on the use of the PGM command (p.8-65)

① When an NEC PROM programmer other than PG-1500 and PG-2000 is connected to channel 2 and the IE-78330-R is used as a terminal, the following control characters cannot be used.

^A ^B ^C ^D ^E ^F ^H ^I ^J ^K ^L
^M ^N ^Q ^S ^W ^Z

When one or more of the above control characters are to be used, enter PGM C <cr> to release the restrictions on the use of the control characters or to change some of the control characters.

② Control characters are changed interactively.

The following 16 control characters can be used.

^A ^B ^E ^F ^G ^N ^O ^P ^R ^T ^U
^V ^W ^X ^Y ^Z

The same control character cannot be specified more than once.

When the DEL key is pressed or ^H is entered, the initial values of the control characters are displayed.

When the escape key is pressed during change of control characters, the changed control characters are invalidated.

③ When an attempt is made to change the restrictions on the use of the control characters, the system automatically enters the terminal mode.

4. Cautions on the use of the PSD command (p.8-70)

① After the sample address is set by the PSA command, sample data remain undefined until a write operation is performed.

② Sample data are cleared when real-time execution is performed again.

③ The internal RAM data sampler is stopped during single-step execution or procedure execution.

5. When data are read from or written into a register during trace (trc:0> mode) or emulation (emu:0> mode), execution of the emulation CPU is temporarily suspended. (p.8-75)

6. When a command to manipulate contents of a register, memory, or SFR is entered during trace (trc:0> mode) or emulation (emu:0> mode), execution of the emulation CPU is temporarily suspended. (p.8-82)

7. All interrupts during the step execution are held. (p.8-89)

8. All interrupts during the procedure execution are held. (p.8-93)

9. Cautions on the use of the SFR command (p.8-99)

   ① SFRs in parentheses are used for reading only and SFRs in angle brackets are used for writing only.

   ② When SFRs are to be manipulated in the emu:0> prompt mode, reading or writing of SFRs is stopped temporarily.

10. SFRs in parentheses are used for reading only and SFRs in angle brackets are used for writing only. (p.8-101)

11. SFRs in parentheses are used for reading only and SFRs in angle brackets are used for writing only. (p.8-103)

12. If the internal RAM (0FE00H to 0FEFFH) is addressed in a mode other than the short direct addressing mode, trace data become unpredictable. In this case, instructions of the target program are executed normally, however. (p.8-125)

13. Cautions on the use of the TRD F command (p.8-126)

    ① xxx is entered for the time tag of the first frame because the value is undefined.

    ② When a time tag value is 255 system clocks or greater, xxx is entered.

14. If the internal RAM (0FE00H to 0FEFFH) is addressed in a mode other than the short direct addressing mode, trace data become unpredictable. In this case, instructions of the target program are executed normally, however. (p.8-132)

15. Cautions on the use of the TRD I command (p.8-133)

    ① xxx is entered for the time tag of the first frame, because the value is undefined.

    ② When a time tag value is 255 system clocks or greater, xxx is entered for the time tag.

    ③ Since read/write accesses to instructions are displayed, the order of displayed trace data may differ from that of the actual operation of the emulation CPU.

④ Data traced in the qualified trace mode (TRM TRX) are displayed in units of frames, even if instruction trace display is specified by entering the TRD I command.

⑤ The first and last frames in data that are traced in the section trace mode (TRM SEC) may become undefined when instruction trace display is specified by entering the TRD I command.

(7) Caution in Chapter 11

Error conditions (p.11-6)

s > e
r > re
i > ie
Non-hexadecimal input

12.2 Cautions Related to the Development Target Products (uPD78330, uPD78334, and uPD78P334)

12.2.1 Cautions related to the internal and external control functions

When the STOP mode is released by NMI

1. When the stop mode is released by entering NMI, the instruction following a STOP mode set instruction (MOV STBC,#byte) is executed and the program branches to the NMI interrupt service routine. To prevent this, enter the NOP instruction after the STOP mode set instruction.

2. If an interrupt occurs during execution of a STOP mode set instruction, the interrupt is handled after the STOP mode is released. Then the program branches to the NMI interrupt service routine. Mask all the maskable interrupts before the STOP mode set instruction is executed.

   The operation after the STOP mode is released depends on the processing mode in which the interrupt occurred.

   1 If a maskable interrupt request for which the vector interrupt service is to be processed occurs, the program branches to the maskable interrupt service routine. Then one instruction is executed and the program branches to the NMI interrupt service routine.

② If a maskable interrupt request for which the macro service is to be processed occurs, the macro service is executed and the program branches to the NMI interrupt service routine. NMI processing is held until the macro service terminates.

3. If a nonmaskable watchdog timer interrupt request occurs during execution of a STOP mode set instructions, the program branches to the watchdog timer interrupt service routine even if the interrupt priority of the watchdog timer interrupt request is lower than that of the NMI interrupt request. Then, one instruction is executed and the program branches to the NMI interrupt service routine. To prevent this, enter NOP at the beginning of the watchdog timer interrupt service routine.

   When setting the stack area, consider that interrupts are nested at two levels.

12.2.2 Cautions related to the instruction set

(1) Cautions when using the ADDC/SUBC sfr,#byte instruction

If one of the following special function registers is specified in the first operand, the operation result becomes incorrect. Do not specify them in the first operand.

o P4, P5, PM5, MM, external SFR

(2) Cautions when using the ADDC/SUBC saddr,#byte instruction

If one of the following special function registers is specified as short direct memory in the first operand, the operation result becomes incorrect. Do not specify them in the first operand.

o P4, P5

(3) Cautions when using the ADDC/SUBC saddr,saddr instruction

If one of the following special function registers is specified as short direct memory in the first operand, the operation result becomes incorrect. Do not specify them in the first operand.

o P4, P5

NEC