User's Manual

SH7612 E10A Emulator

User's Manual

Renesas Electronics
www.renesas-electoronics.com

Rev.2.0 1999.11

Preface

Thank you for purchasing the E10A emulator.

CAUTION

READ section 2, Preparation before Use, of this User's Manual before using the emulator product. Incorrect operation will damage the user system and the emulator product.

This emulator is an efficient development tool for software and hardware of user systems based on Hitachi's original microprocessor. The emulator operates using the Hitachi debugging interface (hereafter referred to as the HDI), which is the interface program that runs on Microsoft® Windows® 95, Microsoft® Windows® 98, or Microsoft® Windows NT® operating system.

This manual describes the functions and operating procedures of the E10A emulator. Sections 1 to 5 describe common features of all types of E10A emulators. Section 6 describes supplements to the E10A emulator. Read section 1.1, Warnings, carefully before using the emulator.

This manual consists of six sections. The information contained in each section is summarized below:

- Section 1, Overview, gives the emulator overview.
- Section 2, Preparation before Use, gives instructions for first-time users, such as preparation before use and system connection.
- Section 3, Tutorial, describes HDI operating examples.
- Section 4, Descriptions of Windows, describes HDI windows for operating the emulator.
- Section 5, Command-line Functions describes how to input HDI commands and command types.
- Section 6, SHxxxx E10A Emulator Specifications describes the features of the E10A emulator for each MCU. Read this section before using the E10A emulator.

The emulator contains three 3.5-inch HDI installation disks in IBM PC 1.44-Mbyte format or the emulator setup program provided with the CD-R.

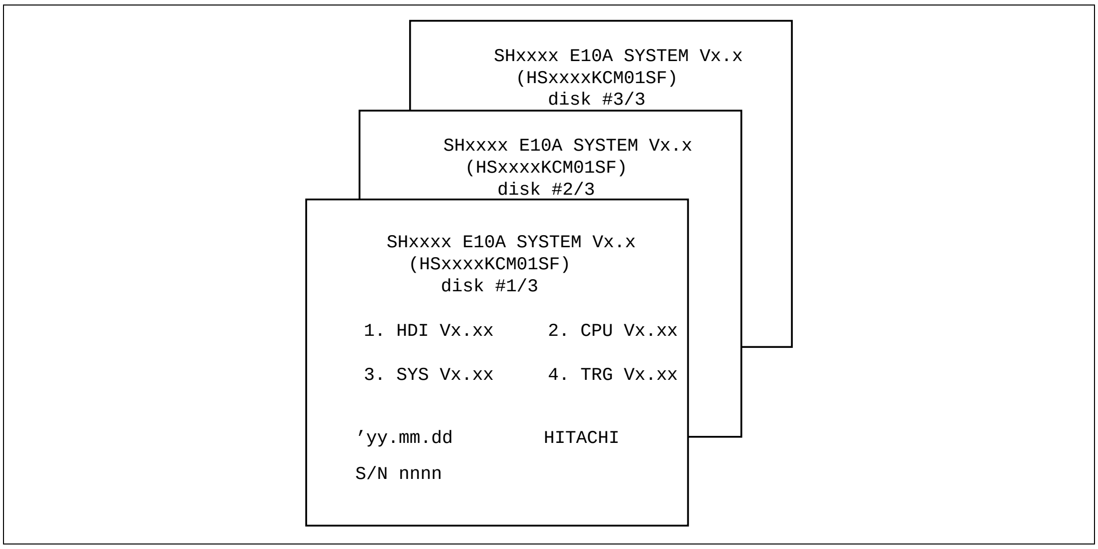


Figure 1 HDI Installation Disks for the E10A Emulator

Before using the HDI installation disks, copy them to other floppy disks according to the descriptions in the manuals of the host computer or operating system. For details, refer to section 2.2, HDI Installation, in this manual.

Related Manuals:

- SH Series Cross Assembler User's Manual
- H Series Linkage Editor User's Manual
- H Series Librarian User's Manual
- SuperH RISC Engine C/C++ Compiler User's Manual
- Hitachi Debugging Interface User's Manual
- Hardware Manual for each MCU
- Programming Manual for each MCU

Notes: 1. IBM PC is a registered trademark of International Business Machines Corporation.

2. Microsoft®, Windows®, and Windows NT® are registered trademarks of Microsoft Corporation in the United States and/or other countries.
Microsoft® Windows® 95 operating system is referred to as Windows® 95 in this user's manual.
Microsoft® Windows® 98 operating system is referred to as Windows® 98 in this user's manual.

Contents

Figures

Tables

IMPORTANT INFORMATION

READ FIRST

• READ this user's manual before using this emulator product.

• <u>KEEP the user's manual handy for future reference.</u>

Do not attempt to use the emulator product until you fully understand its mechanism.

Emulator Product:

Throughout this document, the term "emulator product" shall be defined as the following products produced only by Hitachi, Ltd. excluding all subsidiary products.

- Emulator
- User system interface cable

The user system or a host computer is not included in this definition.

Purpose of the Emulator Product:

This emulator product is a software and hardware development tool for systems employing the Hitachi microcomputer SH7612. This emulator product must only be used for the above purpose.

Limited Applications:

This emulator product is not authorized for use in MEDICAL, atomic energy, aeronautical or space technology applications without consent of the appropriate officer of a Hitachi sales company. Such use includes, but is not limited to, use in life support systems. Buyers of this emulator product must notify the relevant Hitachi sales offices before planning to use the product in such applications.

Improvement Policy:

Hitachi, Ltd. (including its subsidiaries, hereafter collectively referred to as Hitachi) pursues a policy of continuing improvement in design, performance, and safety of the emulator product. Hitachi reserves the right to change, wholly or partially, the specifications, design, user's manual, and other documentation at any time without notice.

Target User of the Emulator Product:

This emulator product should only be used by those who have carefully read and thoroughly understood the information and restrictions contained in the user's manual. Do not attempt to use the emulator product until you fully understand its mechanism.

It is highly recommended that first-time users be instructed by users that are well versed in the operation of the emulator product.

State Law:

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may have other rights which may vary from state to state.

The Warranty is Void in the Following Cases:

Hitachi shall have no liability or legal responsibility for any problems caused by misuse, abuse, misapplication, neglect, improper handling, installation, repair or modifications of the emulator product without Hitachi's prior written consent or any problems caused by the user system.

All Rights Reserved:

This user's manual and emulator product are copyrighted and all rights are reserved by Hitachi. No part of this user's manual, all or part, may be reproduced or duplicated in any form, in hard-copy or machine-readable form, by any means available without Hitachi's prior written consent.

Other Important Things to Keep in Mind:

1. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Hitachi's semiconductor products. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.

2. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi.

Figures:

Some figures in this user's manual may show items different from your actual system.

MCU names:

This user’s manual uses SHxxxx as an example of the MCU names.

Limited Anticipation of Danger:

Hitachi cannot anticipate every possible circumstance that might involve a potential hazard. The warnings in this user's manual and on the emulator product are therefore not all inclusive. Therefore, you must use the emulator product safely at your own risk.

SAFETY PAGE

READ FIRST

• READ this user's manual before using this emulator product.

• KEEP the user's manual handy for future reference.

Do not attempt to use the emulator product until you fully understand its mechanism.

DEFINITION OF SIGNAL WORDS

⚠ **This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.**

⚠ DANGER	**DANGER** indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
⚠ WARNING	**WARNING** indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
⚠ CAUTION	**CAUTION** indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.
CAUTION	**CAUTION** used without the safety alert symbol indicates a potentially hazardous situation which, if not avoided, may result in property damage.

NOTE emphasizes essential information.

⚠ WARNING

Observe the precautions listed below. Failure to do so will result in a FIRE HAZARD and will damage the user system and the emulator product or will result in PERSONAL INJURY. The USER PROGRAM will be LOST.

1. **Do not repair or remodel the emulator product by yourself for electric shock prevention and quality assurance.**
2. **Always switch OFF the host computer and user system before connecting or disconnecting any CABLES or PARTS.**
3. **Connect the connectors in the user system and in the user interface cable by confirming the correct direction.**
4. **If the PCI interface board for the E6000 or E8000 emulator (HS6000EIC01H) and the E10A emulator PCI card are mounted on the same host computer, the connectors may be illegally connected.**

Warnings on Emulator Usage

Be sure to read and understand the warnings below before using this emulator. Note that these are the main warnings, not the complete list.

⚠ WARNING

Always switch OFF the host computer and user system before connecting or disconnecting any CABLES or PARTS.

Failure to do so will result in a FIRE HAZARD and will damage the user system and the emulator product or will result in PERSONAL INJURY. The USER PROGRAM will be LOST.

CAUTION

Place the host computer and user system so that no cable is bent or twisted. A bent or twisted cable will impose stress on the user interface leading to connection or contact failure.

Make sure that the host computer and the user system are placed in a secure position so that they do not move during use nor impose stress on the user interface.

Section 1 Overview

The E10A emulator (hereafter referred to as the emulator) is a software and hardware development support tool for application systems using the microprocessor developed by Hitachi, Ltd.

The PCMCIA card emulator or PCI card emulator (hereafter referred to as the card emulator), which is the main unit of the emulator, is connected, through the Hitachi-UDI (user debug interface) port*, to the user system. The user system can be debugged under the conditions similar to the actual application conditions. The emulator enables debugging anywhere indoors or out. The host computer for controlling the emulator must be an IBM PC compatible machine with a PCMCIA type II or PCI slot.

Figures 1.1 and 1.2 show the system configuration using the emulator.

Note: The Hitachi-UDI is an interface compatible with the Joint Test Action Group (JTAG) specifications.

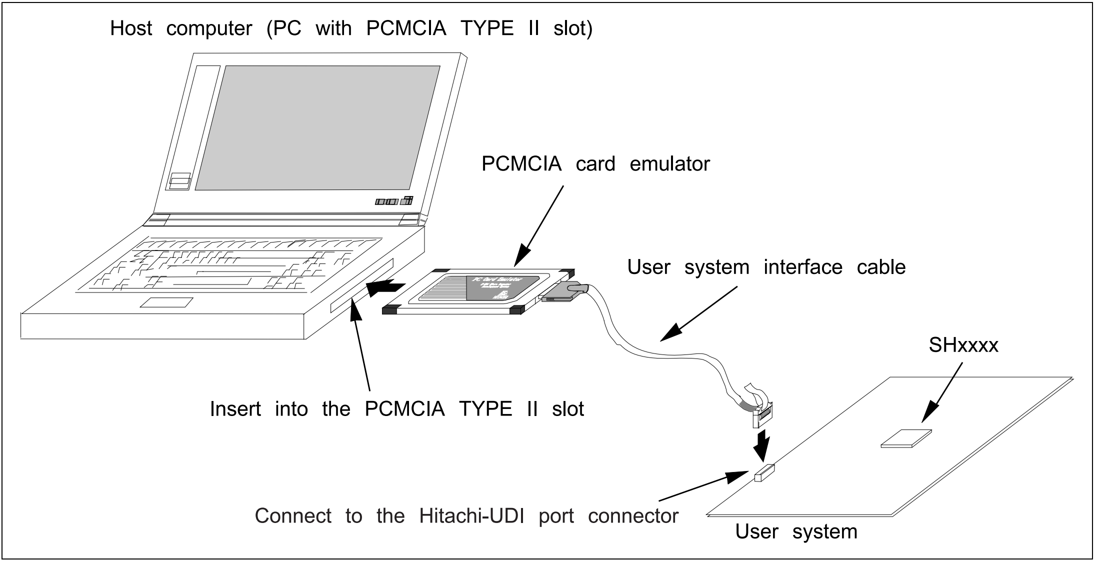

Host computer (PC with PCMCIA TYPE II slot)
PCMCIA card emulator
User system interface cable
SHxxxx
Insert into the PCMCIA TYPE II slot
Connect to the Hitachi-UDI port connector
User system

Figure 1.1 System Configuration with the Emulator (PCMCIA Card Emulator Used)

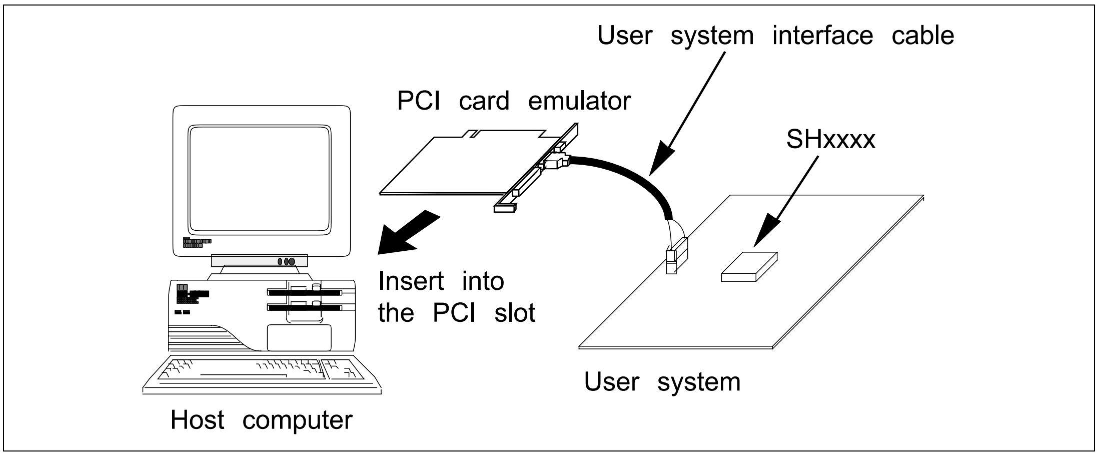


Figure 1.2 System Configuration with the Emulator (PCI Card Emulator Used)

The emulator provides the following features:

- Excellent cost-performance card emulator

  Compactness and low price are implemented using the PCMCIA interface or the PCI interface.
- Realtime emulation

  Realtime emulation of the user system is enabled at the maximum operating frequency of the CPU.
- Excellent operability

  Using the Hitachi Debugging Interface (HDI) on the Microsoft® Windows® 95, Microsoft® Windows® 98, and Microsoft® Windows NT® operating systems enable user program debugging using a pointing device such as a mouse. The HDI enables high-speed down-loading of load module files.
- Various debugging functions

  Various break and trace functions enable efficient debugging. Breakpoints and break conditions can be set by the specific window, trace information can be displayed on a window, and command-line functions can be used.
- Memory access during emulation

  During emulation, the memory contents can be read and modified.
- Debugging of the user system in the final development stage

  The user system can be debugged under conditions similar to the actual application conditions.
- Compact debugging environment

  When the card emulator specific to the PCMCIA interface is used, a laptop computer can be used as a host computer, creating a debugging environment in any place.
- AUD trace function

  The AUD trace function enables realtime trace.

Note: The AUD is an abbreviation of the advanced user debugger.

1.1 Warnings

CAUTION

READ the following warnings before using the emulator product. Incorrect operation will damage the user system and the emulator product. The USER PROGRAM will be LOST.

1. Check all components against the component list after unpacking the emulator.
2. Never place heavy objects on the casing.
3. Protect the emulator from excessive impacts and stresses. For details, refer to section 1.2, Environmental Conditions.
4. Do not insert the emulator into any slot (PCMCIA TYPE II slot or PCI slot) other than the specified one.
5. When moving the host computer or user system, take care not to vibrate or damage it.
6. After connecting the cable, check that it is connected correctly. For details, refer to section 2, Preparation before Use.
7. Supply power to the connected equipment after connecting all cables. Cables must not be connected or removed while the power is on.

1.2 Environmental Conditions

CAUTION

Observe the conditions listed in tables 1.1 and 1.2 when using the emulator. Failure to do so will damage the user system and the emulator product. The USER PROGRAM will be LOST.

Table 1.1 Environmental Conditions

Item	Specifications
Temperature	Operating: +10°C to +35°C Storage: –10°C to +50°C
Humidity	Operating: 35% RH to 80% RH, no condensation Storage: 35% RH to 80% RH, no condensation
Vibration	Operating: 2.45 m/s^2 max. Storage: 4.9 m/s^2 max. Transportation: 14.7 m/s^2 max.
Ambient gases	There must be no corrosive gases present

Table 1.2 lists the acceptable operating environments.

Table 1.2 Operating Environments

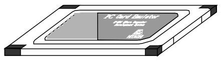
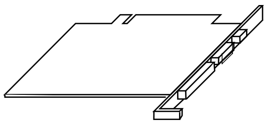
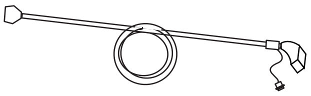
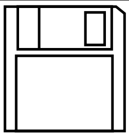
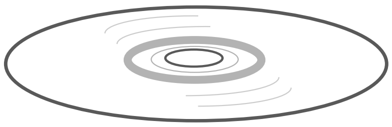
Item	Description
Host computer	Built-in Pentium or higher-performance CPU (166 MHz or higher recommended); IBM PC or compatible machine with the PCMCIA TYPE II slot or the PCI slot.
OS	Windows® 95, Windows® 98, or Windows NT®
Minimum memory capacity	32 Mbytes or more (double of the load module size recommended)
Hard-disk capacity	Installation disk capacity: 5 Mbytes or more. (Prepare an area at least double the memory capacity (four-times or more recommended) as the swap area.)
Floppy-disk drive*	Required to install the HDI.
Pointing device such as mouse	Connectable to the host computer; compatible with Windows® 95, Windows® 98, and Windows NT®.
Power voltage	5.0 ± 0.25 V
Current consumption	HSxxxxKCM01H: 110 mA (max) HSxxxxKCM02H: 230 mA (max) HSxxxxKCI01H: 340 mA (max) HSxxxxKCI02H: 480 mA (max)
CD-ROM drive	Required to refer to the emulator user's manual provided by the CD-R.

Note: In some types of emulators, the installation program is provided with the CD-R. In this case, prepare the CD-ROM drive.

1.3 Components

Check all the components listed below after unpacking. If the components are not complete, contact a Hitachi sales agency. For details, refer to section 6.1, Components of the Emulator.

Table 1.3 Components of the Emulator

Classification	Component	Appearance
Hardware	Card emulator	 or 
	User system interface cable	
Software	Emulator setup program (3.5-inch floppy disks*)	
	SHxxxx E10A Emulator User's Manual and Hitachi Debugging Interface User's Manual (CD-R)	

Note: Since both the emulator software and manuals are included on the CD-R, the floppy disk may not be included with some types of emulators.

Section 2 Preparation before Use

2.1 Emulator Preparation

WARNING

READ the reference sections shaded in figure 2.1 before using the emulator product. Incorrect operation will damage the user system and the emulator product. The USER PROGRAM will be LOST.

Unpack the emulator and prepare it for use as follows:

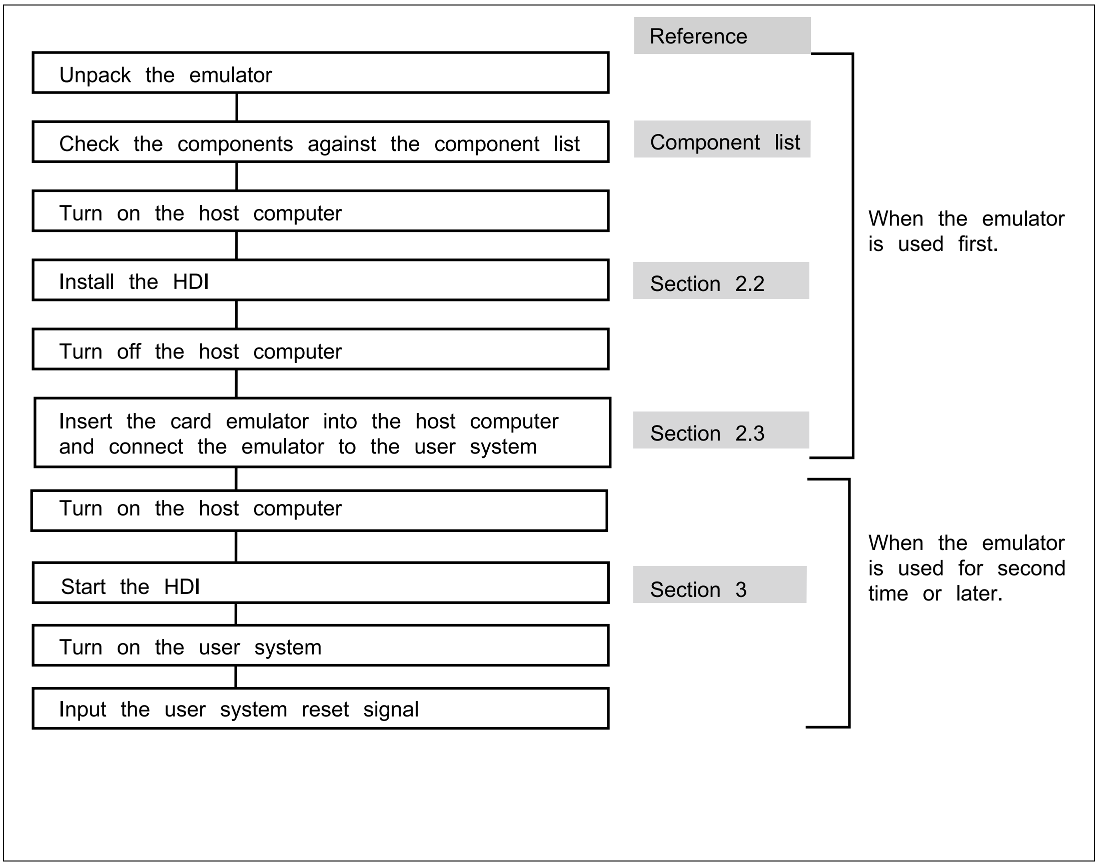

Reference
Unpack the emulator
Check the components against the component list
Component list
Turn on the host computer
When the emulator is used first.
Install the HDI
Section 2.2
Turn off the host computer
Insert the card emulator into the host computer and connect the emulator to the user system
Section 2.3
Turn on the host computer
When the emulator is used for second time or later.
Start the HDI
Section 3
Turn on the user system
Input the user system reset signal

Figure 2.1 Emulator Preparation Flow Chart

2.2 HDI Installation

This section describes how to install the HDI stored in the HDI installation disks into the host computer.

2.2.1 HDI Installation Disk

Three HDI installation disks, which are formatted as IBM PC 1.44-Mbyte disks, are provided with the emulator (figure 2.2).

Note: The floppy disks may not be included with some types of emulators since the emulator setup program is provided with the CD-R.

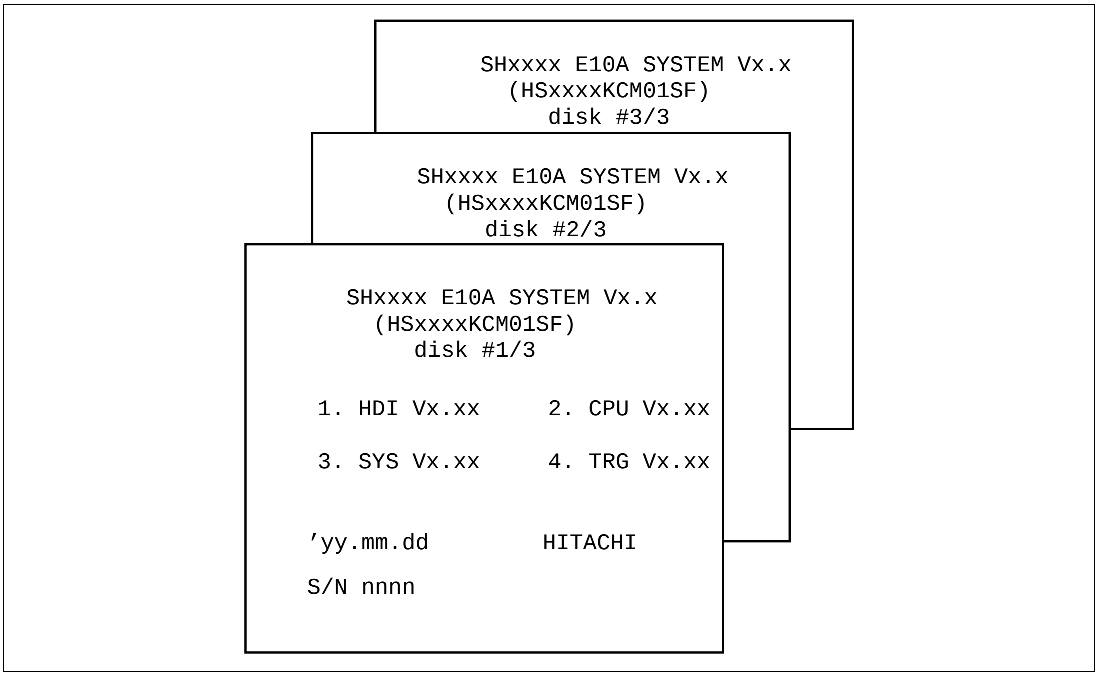


Figure 2.2 HDI Installation Disks

2.2.2 Installing the HDI

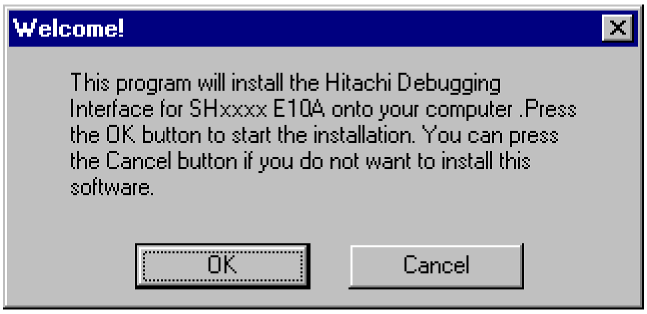
This section describes an example of installing the HDI on the IBM PC. For an emulator with floppy disks, use backup installation floppy disks for installing.

The following describes the installation by the floppy disks. When the emulator setup program is provided with the CD-R, start [SETUP.EXE] in the \SETUP directory.

- Insert the HDI installation disk #1 into the floppy disk drive (assumed to be drive A) of the host computer.
- Start [setup.exe] of the floppy disk.

Figure 2.3 [setup.exe] Icon

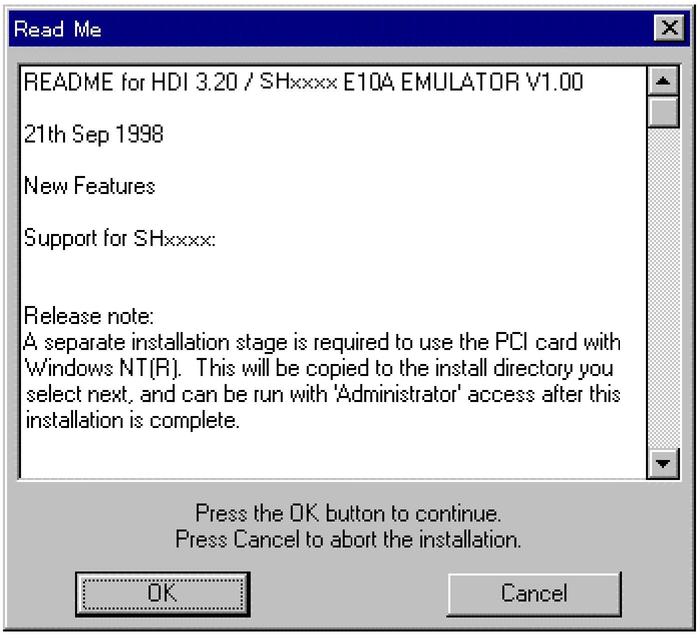
- This runs the HDI installer, and the following [Welcome!] dialog box will be displayed.



This program will install the Hitachi Debugging Interface for SHxxxx E10A onto your computer .Press the OK button to start the installation. You can press the Cancel button if you do not want to install this software.

Figure 2.4 [Welcome!] Dialog Box

- Click the [OK] button to proceed with the installation.
- The [Read Me] dialog box is then displayed. Click the [OK] button to proceed.



README for HDI 3.20 / SHxxxx E10A EMULATOR V1.00

21th Sep 1998

New Features

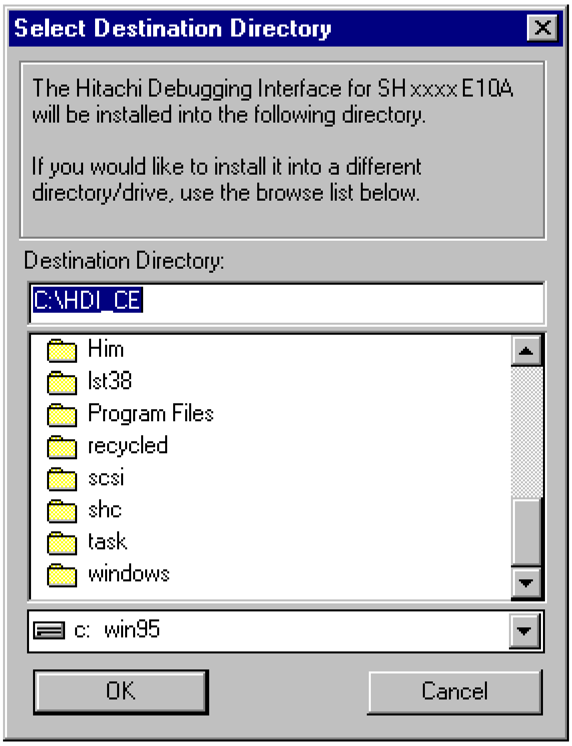
Support for SHxxxx:

Release note:
A separate installation stage is required to use the PCI card with Windows NT(R). This will be copied to the install directory you select next, and can be run with 'Administrator' access after this installation is complete.

Press the OK button to continue.
Press Cancel to abort the installation.

Figure 2.5 [Read Me] Dialog Box (Example)

- After the [Select Destination Directory] dialog box then allows the user to select a directory for installing the HDI, click the [OK] button. When installing the HDI into the default directory (`C:\HDI_CE`), just click the [OK] button.



Select Destination Directory

The Hitachi Debugging Interface for SH xxxx E10A will be installed into the following directory.

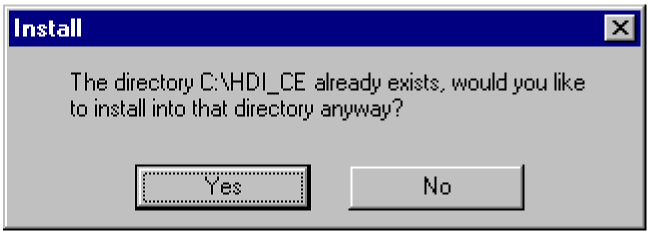
If you would like to install it into a different directory/drive, use the browse list below.

Destination Directory:

C:\HDI_CE

Figure 2.6 [Select Destination Directory] Dialog Box

- When the specified directory name already exists, the [Install] dialog box is displayed. When installing the HDI into the existing directory, click the [Yes] button. If the user wants to change the directory, click the [No] button. The [Select Destination Directory] dialog box then allows the user to select another directory.

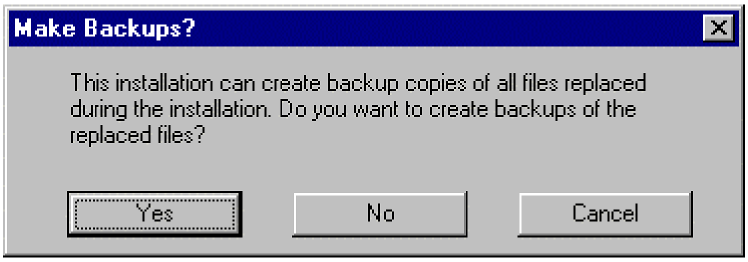


Install

The directory C:\HDI_CE already exists, would you like to install into that directory anyway?

Figure 2.7 [Install] Dialog Box

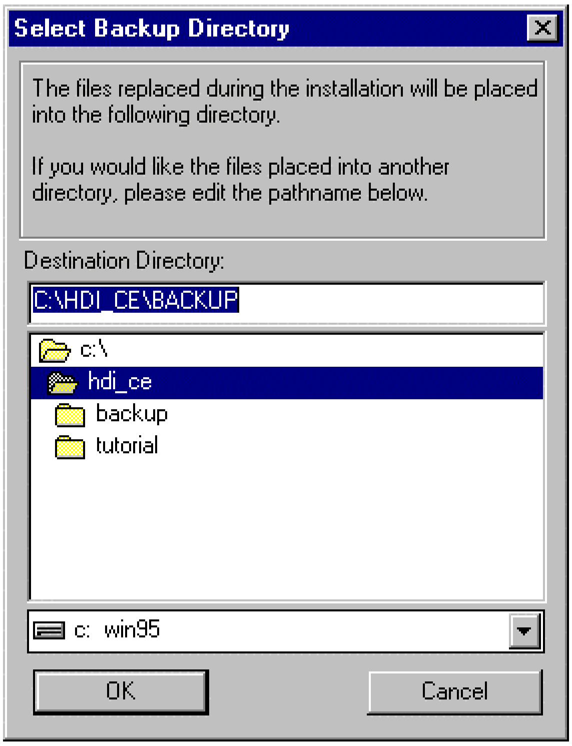
- Clicking the [Yes] button in the [Install] dialog box displays the [Make Backups?] dialog box to ask the user whether backups should be made of the files replaced by the installation. Click the [Yes] button to save any files or the [No] button if the user does not want to make a backup.



This installation can create backup copies of all files replaced during the installation. Do you want to create backups of the replaced files?

Figure 2.8 [Make Backups?] Dialog Box

- When the user selects the [Yes] button in the [Make Backups?] dialog box, the [Select Backup Directory] dialog box is displayed. Specify the backup file name then click the [OK] button to proceed. To save into the default directory (`C:\HDI_CE\BACKUP`), just click the [OK] button.


Select Backup Directory

The files replaced during the installation will be placed into the following directory.

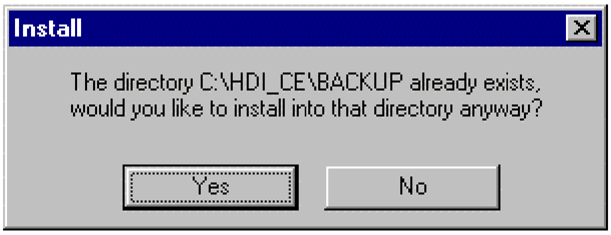
If you would like the files placed into another directory, please edit the pathname below.

Destination Directory:

C:\HDI_CE\BACKUP

Figure 2.9 [Select Backup Directory] Dialog Box

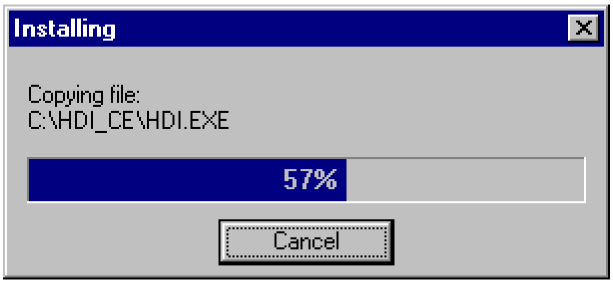
- When the specified directory name already exists, the [Install] dialog box is displayed. When installing the HDI into the existing directory, click the [Yes] button. If the user wants to change the directory, click the [No] button. The [Select Backup Directory] dialog box then allows the user to select another directory.


Install

The directory C:\HDI_CE\BACKUP already exists, would you like to install into that directory anyway?

Figure 2.10 [Install] Dialog Box

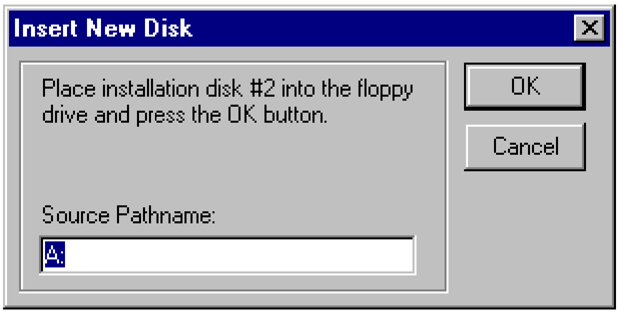
- The installer then installs the HDI files into the specified directory.


Installing

Copying file:
C:\HDI_CE\HDI.EXE

Figure 2.11 [Installing] Dialog Box

- During the installation, the following message is displayed to prompt to insert HDI installation disk #2. Take out HDI installation disk #1, insert HDI installation disk #2, and click the [OK] button.

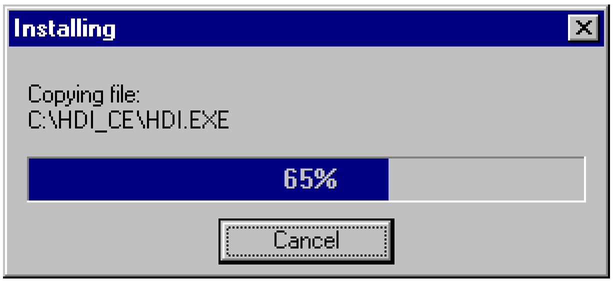

Insert New Disk

Place installation disk #2 into the floppy drive and press the OK button.

Source Pathname:
A:

Figure 2.12 [Insert New Disk] Dialog Box

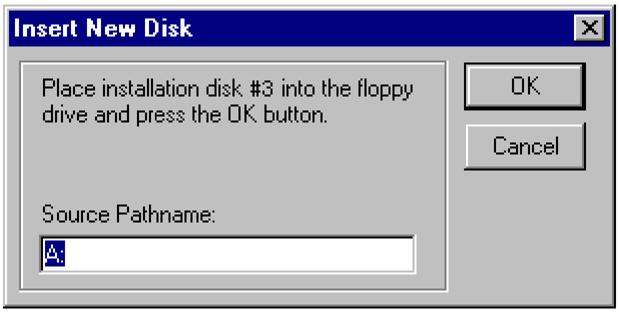
- The installer then installs the HDI files into the specified directory.


Installing

Copying file:
C:\HDI_CE\HDI.EXE

Figure 2.13 [Installing] Dialog Box

- During the installation, the following message is displayed to prompt to insert HDI installation disk #3. Take out HDI installation disk #2, insert HDI installation disk #3, and click the [OK] button.


Insert New Disk

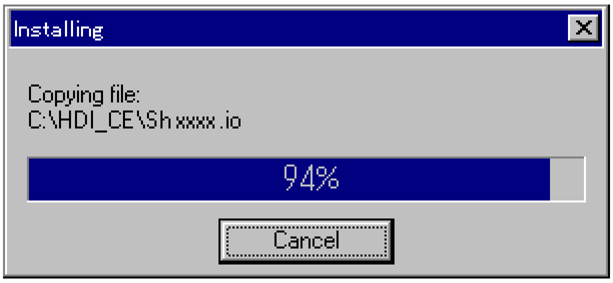
Place installation disk #3 into the floppy drive and press the OK button.

Source Pathname:

A:

Figure 2.14 [Insert New Disk] Dialog Box

- The installer then installs the HDI files into the specified directory.

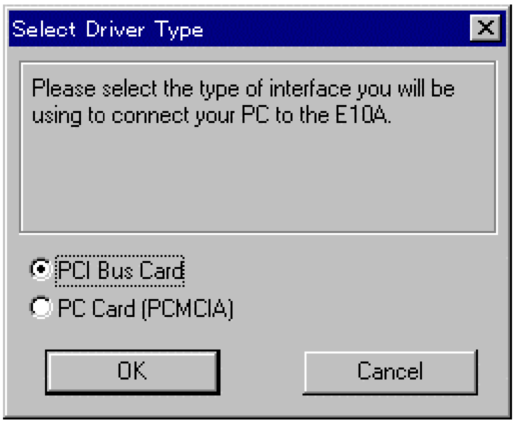

Installing

Copying file:
C:\HDI_CE\Sh xxxx .io

94%

Figure 2.15 [Installing] Dialog Box

- During the installation, to select whether the PCI card emulator or the PCMCIA card emulator is used, the following message is displayed. Then click the [OK] button. (This message is not displayed when Windows NT® is used.)

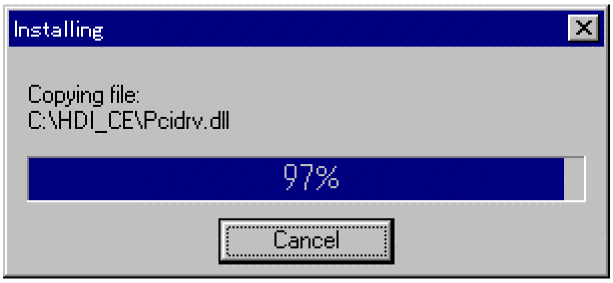

Select Driver Type

Please select the type of interface you will be using to connect your PC to the E10A.

[✓ PCI Bus Card]
PC Card (PCMCIA)

Figure 2.16 [Select Driver Type] Dialog Box

- The installer then installs the HDI files into the specified directory.

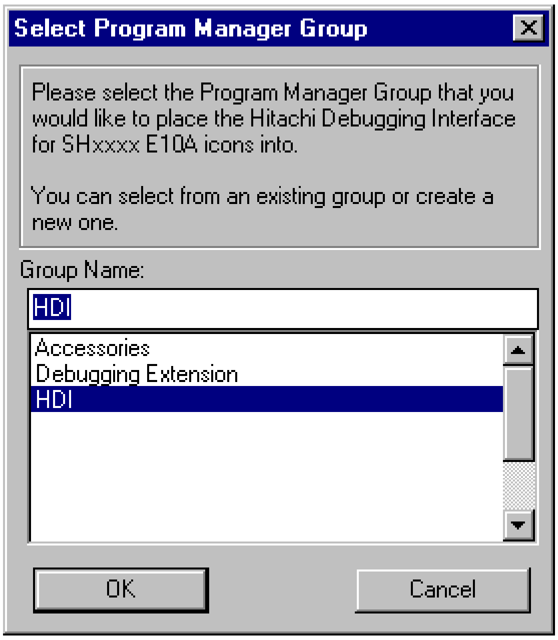

Installing

Copying file:
C:\HDI_CE\Pcidrv.dll

97%

Figure 2.17 [Installing] Dialog Box

- The [Select Program Manager Group] dialog box allows the user to specify the program group name for the HDI icons. To specify `HDI` (the default group name) for a program group name, just click the [OK] button.



Select Program Manager Group

Please select the Program Manager Group that you would like to place the Hitachi Debugging Interface for SHxxxx E10A icons into.

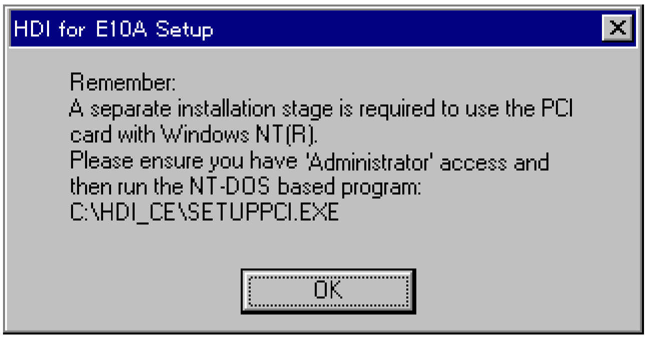
You can select from an existing group or create a new one.

Group Name: HDI

Accessories
Debugging Extension
HDI

Figure 2.18 [Select Program Manager Group] Dialog Box

- When Windows NT® is used for the operating system, the [HDI for E10A Setup] dialog box is displayed. Click the [OK] button.



HDI for E10A Setup

Remember:
A separate installation stage is required to use the PCI card with Windows NT(R).
Please ensure you have 'Administrator' access and then run the NT-DOS based program:
C:\HDI_CE\SETUPPCI.EXE

Figure 2.19 [HDI for E10A Setup] Dialog Box

- Specifying the program group name enables the installer to create the following icons in the program group the user specified.

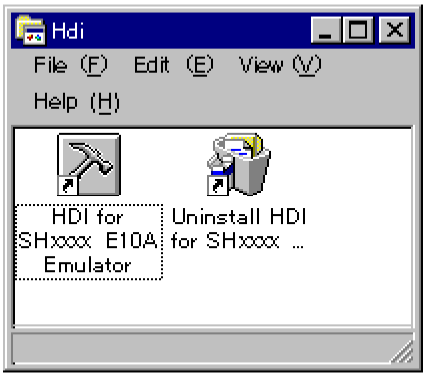

Hdi
File (F) Edit (E) View (V)
Help (H)
HDI for SHxxxx E10A Emulator
Uninstall HDI for SHxxxx ...

Figure 2.20 [HDI] Program Group (When Windows® 95 is Used)

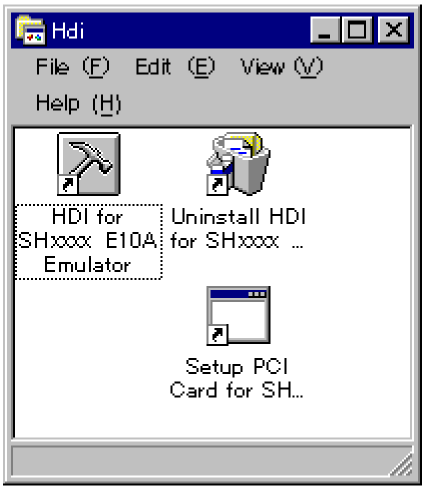

Hdi
File (F) Edit (E) View (V)
Help (H)
HDI for SHxxxx E10A Emulator
Uninstall HDI for SHxxxx ...
Setup PCI Card for SH...

Figure 2.21 [HDI] Program Group (When Windows NT® is Used)

- These icons represent the following functions:
  [HDI for SHxxxx E10A Emulator] executes the HDI program.
  [Uninstall HDI for SHxxxx E10A Emulator] deletes the HDI software and the associated files when the HDI is uninstalled.
  [Setup PCI Card for SHxxxx E10A Emulator] sets up the PCI driver when Windows NT® is used.

Notes: 1. When Windows NT® is used, be sure to set up the PCI driver by double clicking the [Setup PCI Card for SHxxxx E10A Emulator] icon. If the icon is not double-clicked, the PCI driver is incorrectly set.

2. When the PCMCIA card is used, a message for installing the driver is displayed if the card is being inserted for the first time. The driver is provided on installation disk #3 (the \SETUP directory when the emulator setup program is provided with the CD-R). Install it according to the screen instructions.

2.3 Connecting the Host Computer with the Card Emulator

Insert the card emulator into the PCMCIA TYPE II slot or the PCI slot of the host computer (figures 2.22 and 2.23).

Note: Be sure to install the HDI before the card emulator is inserted.

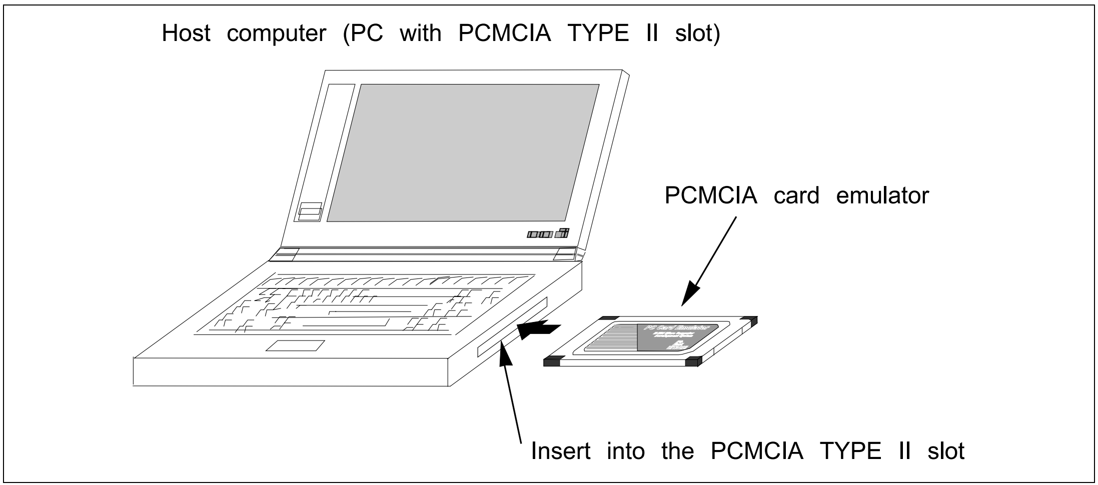

Host computer (PC with PCMCIA TYPE II slot)
PCMCIA card emulator
Insert into the PCMCIA TYPE II slot

Figure 2.22 Inserting the PCMCIA Card Emulator into the Host Computer

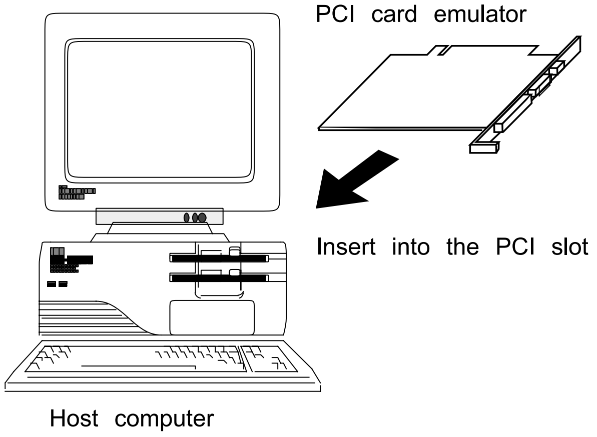


Figure 2.23 Inserting the PCI Card Emulator into the Host Computer

Use the procedure, described in section 2.4, to connect the emulator to the user system with the user system interface cable, or to disconnect them when moving the emulator or the user system.

Note: When installing the PCI card emulator, note the following:

1. Turn off the host computer.
2. Insert the emulator into the PCI slot in parallel.
3. Screw in the emulator after confirming the connector and cable positions.

2.4 Connecting the Card Emulator with the User System

(1) The Hitachi-UDI port connector must be installed to the user system. Table 2.1 shows the recommended Hitachi-UDI port connector for the emulator.

Table 2.1 Recommended Hitachi-UDI Port Connector

Type Number	Manufacturer	Specifications
7614-6002, 2514-6002	Sumitomo 3M Limited	14-pin straight type
DX10M-36S	Hirose Electric Co., Ltd.	Screw type
DX10M-36SE, DX10GM-36SE		Lock-pin type

Note: When the 14-pin connector is used, do not install any components within 3 mm of the Hitachi-UDI port connector. When the 36-pin connector is used, do not connect any components under the Hitachi-UDI connector.

(2) Connect the user system interface cable connector to the Hitachi-UDI port connector installed on the user system. Section 6.2 shows the pin arrangement of the user system interface cable connector.

Note that the TDO signal of the user system interface cable connector must be connected to the TDI pin of the Hitachi-UDI port connector and the TDI signal of the user system interface cable connector must be connected to the TDO pin of the Hitachi-UDI port connector. Section 6.2 shows the pin arrangement of the Hitachi-UDI port connector.

(3) Figure 2.24 shows how to connect the user system interface cable to the user system when the 14-pin straight type connector is used. Connect the ground line of the cable to the user system ground. The end of the ground line has a hole having a diameter of 3 mm, and therefore, when the ground line is screwed to the user system, the screw diameter must be 3 mm.

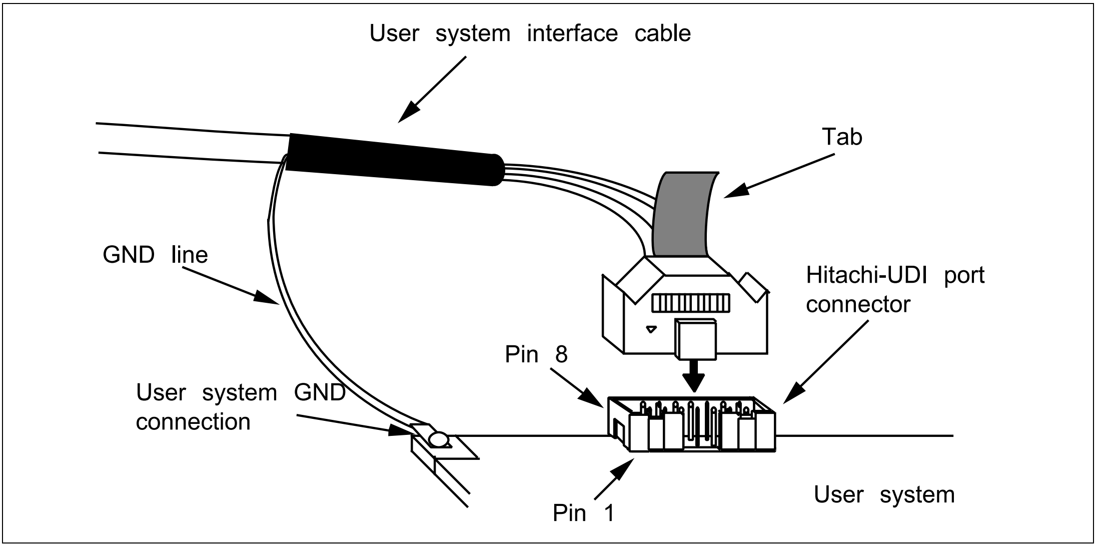

User system interface cable
Tab
GND line
Hitachi-UDI port connector
Pin 8
User system GND connection
Pin 1
User system

Figure 2.24 Connecting the User System Interface Cable to the User System when the 14-pin Straight Type Connector is Used

Notes: 1. To connect the signals output from the Hitachi-UDI port connector, refer to the MCU pin alignment.

2. To remove the user system interface cable from the user system, pull the tab on the connector upward.

3. The range of frequencies that the Hitachi-UDI operates at is different according to the MCUs used. For details, refer to section 6.4.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK).

4. Connect the Hitachi-UDI signals from the Hitachi-UDI port connector directly to the MCU.

5. When developing user systems, do not connect the TDI and TDO signals of the MCU to the boundary scan loop, or separate them by using a switch.

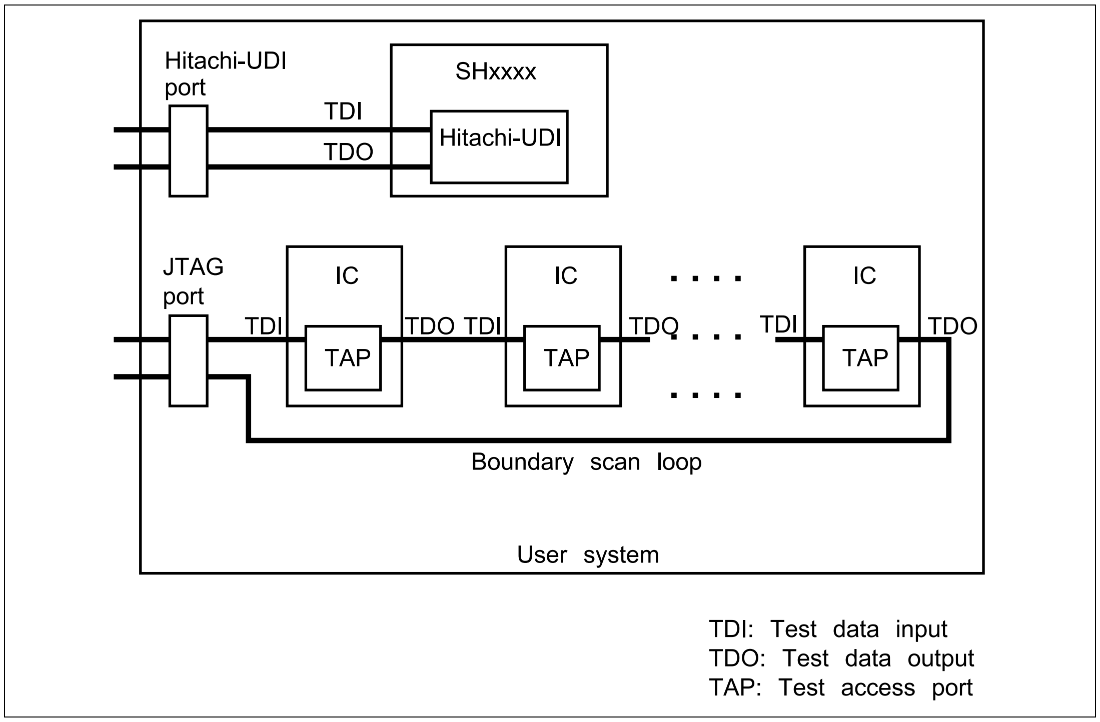


Figure 2.25 User System Example

2.5 System Check

When the HDI program is executed, check that the emulator operates correctly according to the following procedure:

1. Insert the card emulator into the host computer.
2. Connect the user system interface cable to the connector of the card emulator.
3. Connect the user system interface cable to the Hitachi-UDI port connector.
4. Power on the host computer and select [HDI for SHxxxx E10A Emulator] from the [Start] menu.

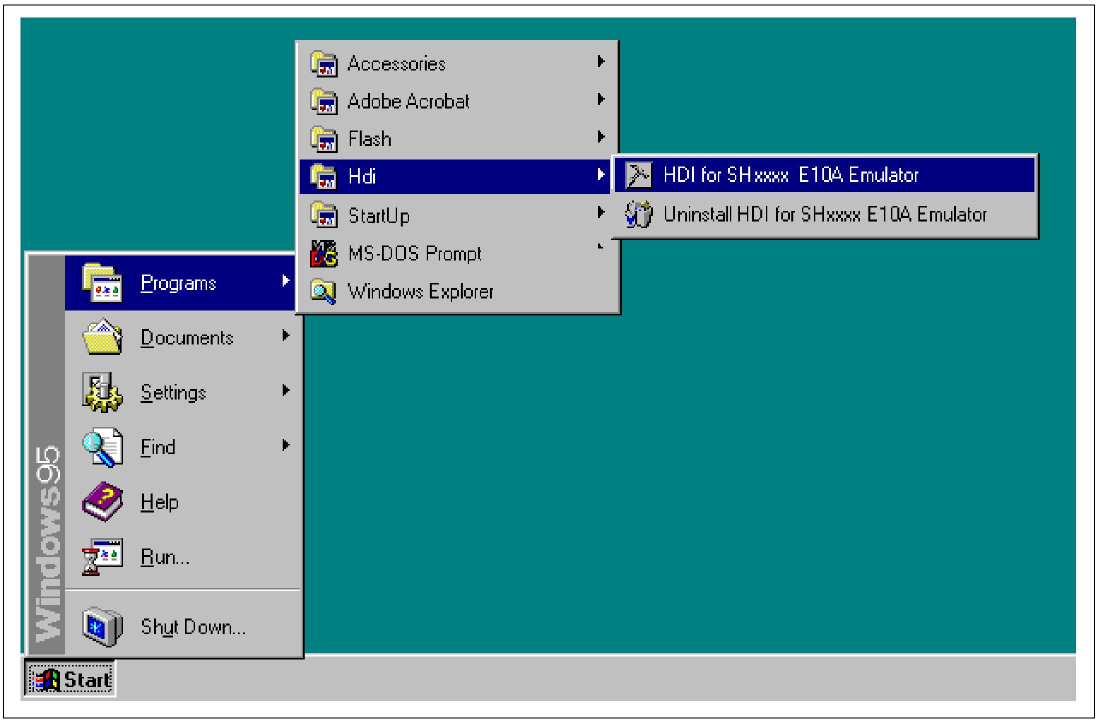


Figure 2.26 [Start] Menu

5. Select setting to be used. (This procedure will not be executed by target MCUs.)

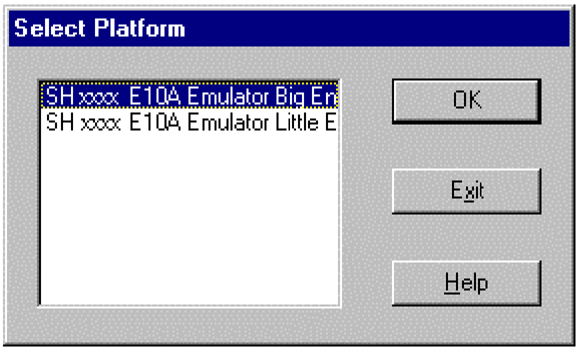
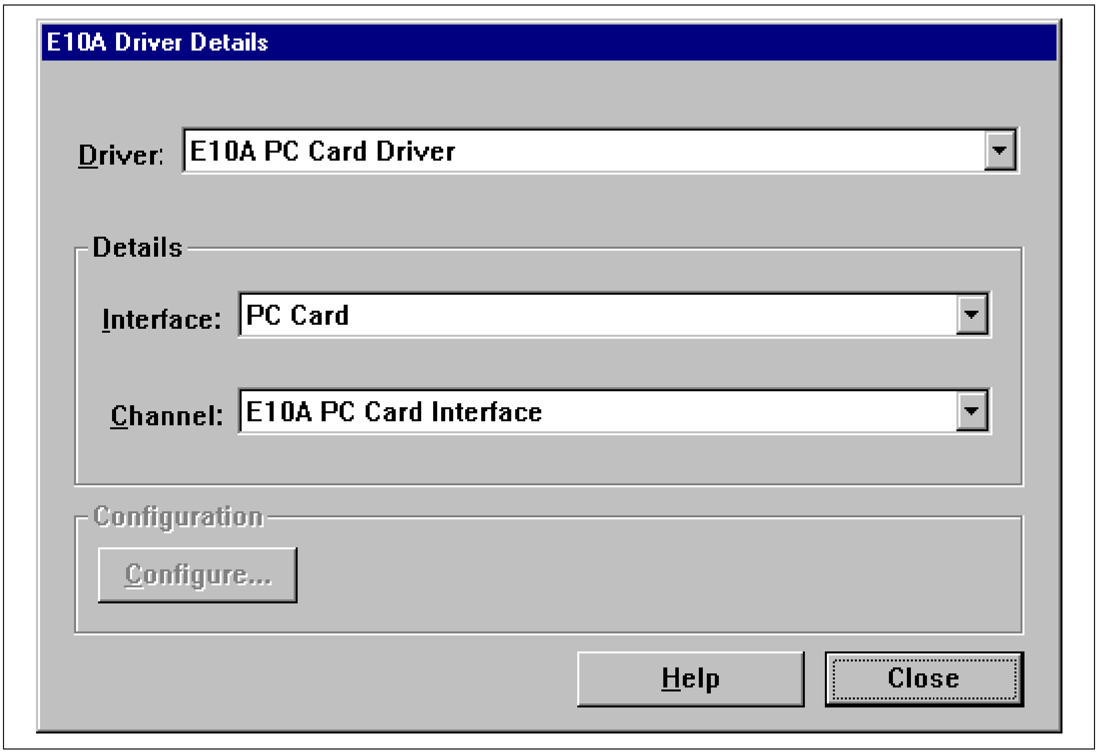


Figure 2.27 [Select Platform] Dialog Box

6. The [E10A Driver Details] dialog box is displayed. With the [Driver] combo box, select the driver to connect the HDI with the emulator. [Interface] displays the interface name of the PC interface board to be connected, and [Channel] displays the interface to which the board is connected. Once the driver is selected in the [E10A Driver Details] dialog box, this dialog box is not displayed when the HDI is run next time. (This procedure will not be executed by target MCUs.)


E10A Driver Details

Driver: E10A PC Card Driver

Details

Interface: PC Card

Channel: E10A PC Card Interface

Figure 2.28 [E10A Driver Details] Dialog Box

- With the [Driver] combo box, select the driver to connect the HDI with the emulator.

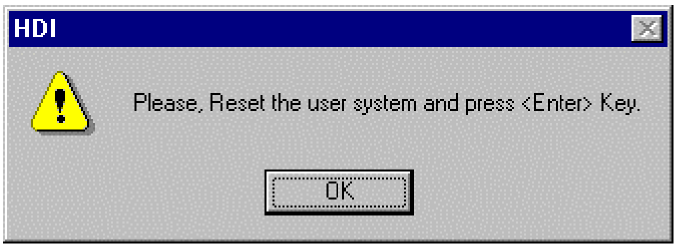
- [Interface] displays the interface name of the card to be connected, and [Channel] displays the interface to which the board is connected.

  [Driver] combo box: Select [E10A PC Card Driver] to use the PCMCIA card.
  Select [E10A PCI Card Driver] to use the PCI card. For details, refer to table 6.3.

  [Interface] combo box: Select [PC Card] to use the PCMCIA card.
  Select [PCI] to use the PCI card. (If the driver is not installed, the [PC Card] or [PCI] is not displayed.)

- Click the [Close] button.

7. The HDI window is displayed, and the dialog box is displayed as shown in figure 2.29.


HDI

Please, Reset the user system and press <Enter> Key.

Figure 2.29 Dialog Box of the RESET Signal Input Request Message

8. Power on the user system.
9. Input the reset signal from the user system, and click the [OK] button.
10. When "Link Up" is displayed on the status bar, the HDI initiation is completed.

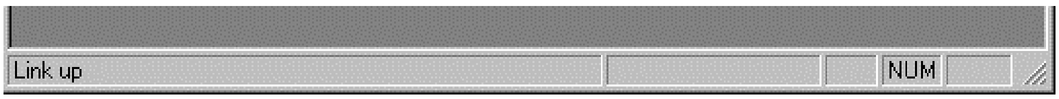
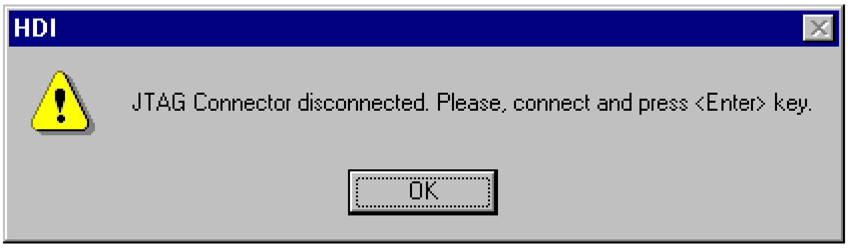

Link up

Figure 2.30 [HDI] Status Bar

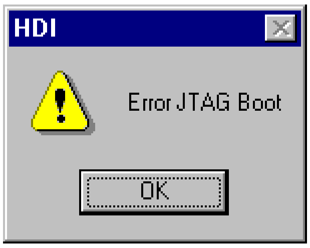
Notes: 1. If the user system interface cable is disconnected from the user system during user program execution, the following dialog box will be displayed.


HDI

JTAG Connector disconnected. Please, connect and press <Enter> key.

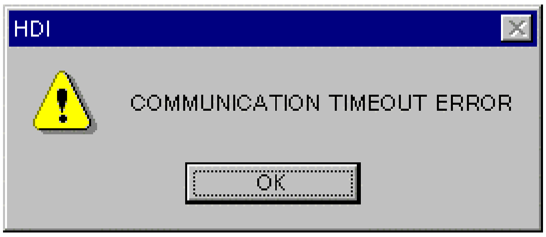
Figure 2.31 [JTAG Connector Disconnected] Dialog Box

2. If the emulator is not initiated, the following dialog boxes shown in figures 2.32 and 2.33 will be displayed. Check that the Hitachi-UDI port connector on the user system is correctly connected.


HDI

Error JTAG Boot

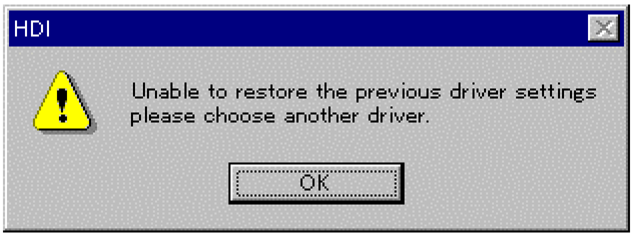
Figure 2.32 [Error JTAG Boot] Dialog Box


HDI

COMMUNICATION TIMEOUT ERROR

Figure 2.33 [COMMUNICATION TIMEOUT ERROR] Dialog Box

3. **If the driver is not correctly connected, the following dialog box will be displayed.**


HDI

Unable to restore the previous driver settings
please choose another driver.

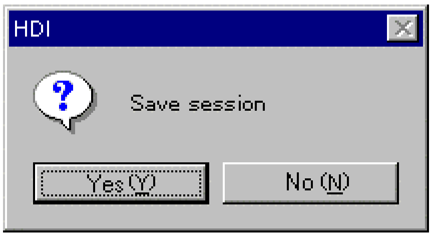
Figure 2.34 [Unable to restore the previous driver settings] Dialog Box

The [E10A Driver Details] dialog box is displayed when the [OK] button is clicked. Select the correct driver.

2.6 Ending the HDI

Power off the emulator by using the following procedure:

1. Power off the user system.
2. Select [Exit] from the [File] menu to end the HDI. The [Save session] dialog box is displayed.


HDI

Save session

Figure 2.35 [Save session] Dialog Box

If necessary, click the [Yes] button to save session. After saving session, the HDI ends. If not necessary, click the [No] button to end the HDI.

2.7 Uninstalling the HDI

Uninstallation of the HDI is described below.

1. First, select [Uninstall HDI for SHxxxx E10A Emulator] from the [Start] menu.

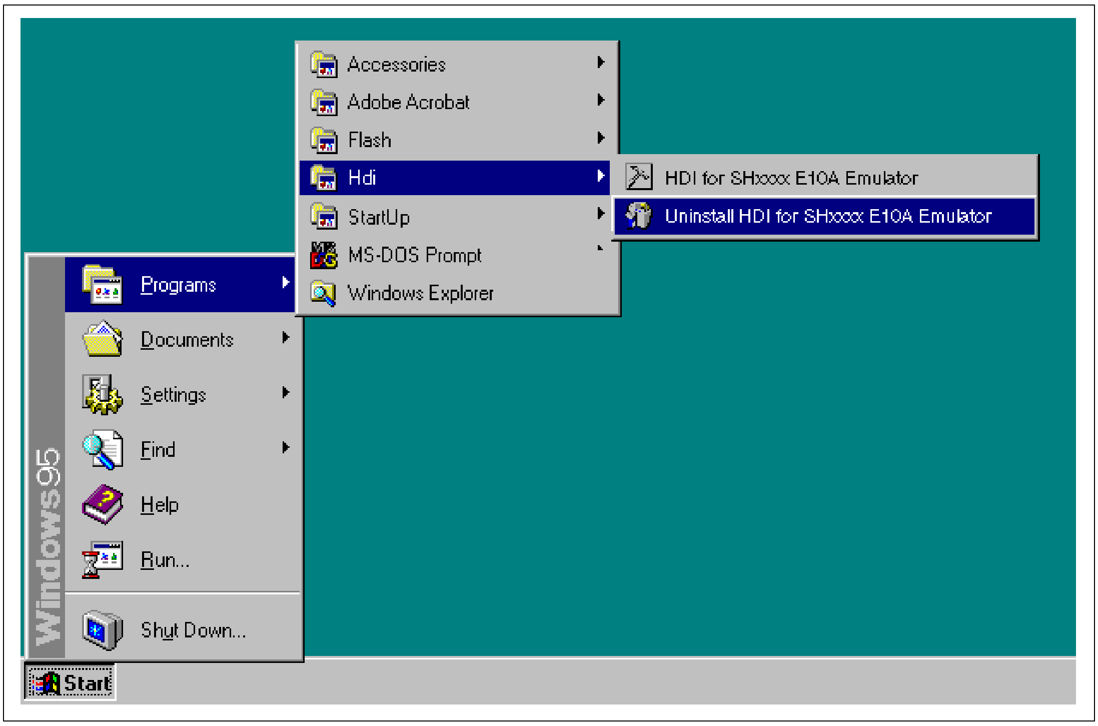
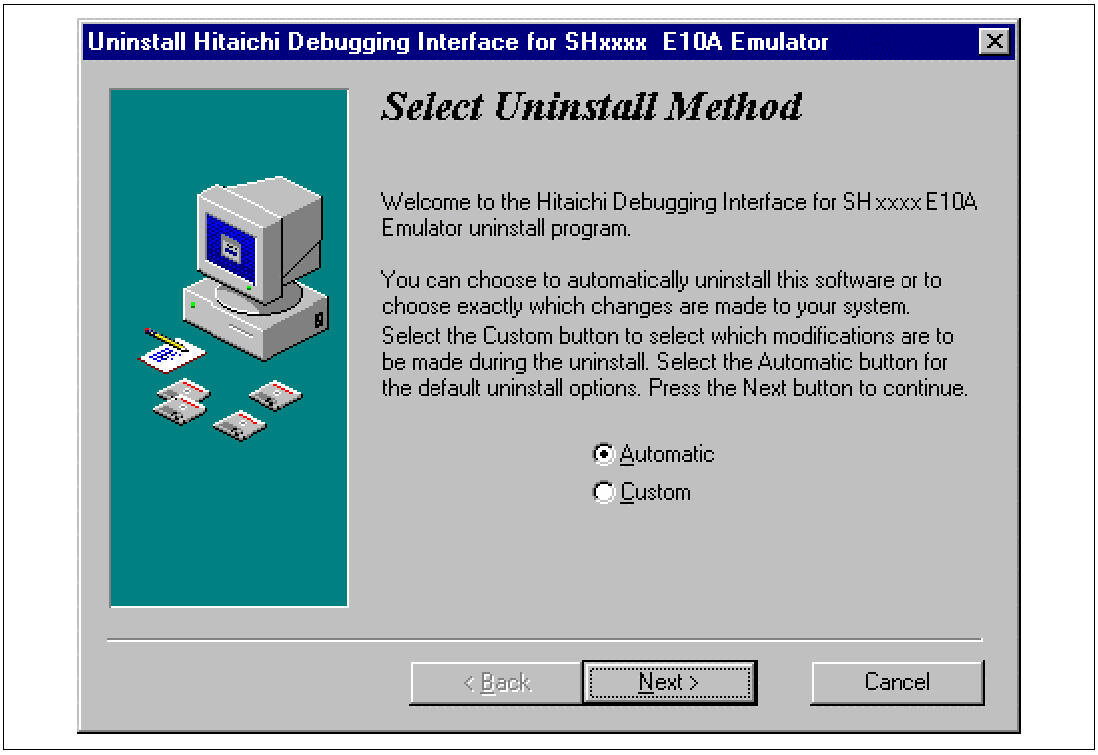


Figure 2.36 [Start] Menu

The [Uninstall Hitachi Debugging Interface for SHxxxx E10A Emulator] dialog box is displayed.


Uninstall Hitaichi Debugging Interface for SHxxxx E10A Emulator
Select Uninstall Method

Welcome to the Hitaichi Debugging Interface for SH xxxx E10A Emulator uninstall program.

You can choose to automatically uninstall this software or to choose exactly which changes are made to your system. Select the Custom button to select which modifications are to be made during the uninstall. Select the Automatic button for the default uninstall options. Press the Next button to continue.

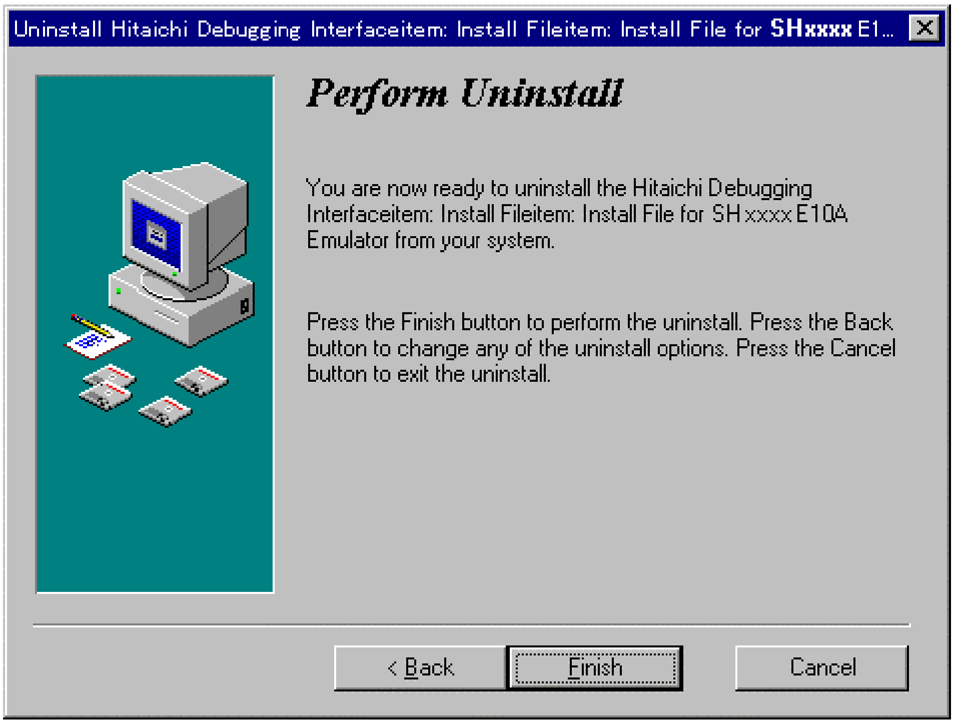
[✓ Automatic] Custom

Figure 2.37 [Uninstall Hitachi Debugging Interface for SHxxxx E10A Emulator] Dialog Box

The function of each button is as follows:

Automatic: The HDI is automatically removed.
Custom: The files to be removed can be selected.
Cancel: Uninstallation is cancelled.

2. Always select the [Automatic] button and click [Next] to display the [Uninstall Hitachi Debugging Interface item: Install File item: Install File for SHxxxx E10A Emulator] dialog box.



Uninstall Hitaichi Debugging Interfaceitem: Install Fileitem: Install File for SHxxxx E1...

Perform Uninstall

You are now ready to uninstall the Hitaichi Debugging Interfaceitem: Install Fileitem: Install File for SHxxxx E10A Emulator from your system.

Press the Finish button to perform the uninstall. Press the Back button to change any of the uninstall options. Press the Cancel button to exit the uninstall.

Figure 2.38 [Uninstall Hitachi Debugging Interface item: Install File item: Install File for SHxxxx E10A Emulator] Dialog Box

3. Click the [Finish] button to uninstall the HDI.

Section 3 Tutorial

3.1 Introduction

The following describes the main functions of the HDI by using a tutorial program. This section uses the SH7729 E10A emulator using the PCMCIA card as an example.

The tutorial program is based on the C program that sorts ten random data items in ascending or descending order. The tutorial program performs the following actions:

- The `main` function generates random data to be sorted.
- The `sort` function sorts the generated random data in ascending order.
- The `change` function then sorts the data in descending order.

The tutorial program is included in the `\tutorial\sort.c` file. The compiled load module is provided in the SYSROF format and is included in the `sort.abs` file.

Table 3.1 lists the tutorial program configuration.

Table 3.1 Tutorial Program Configuration

Item	Contents
Tutorial file (load module)	c:\hdi_ce\tutorial\sort.abs
Tutorial file (source file)	c:\hdi_ce\tutorial\sort.c
Make file (DOS batch file)	c:\hdi_ce\tutorial\tutorial.bat
Subcommand file for linkage editor	c:\hdi_ce\tutorial\tutorial.sub

For the operating environment, use area 0 (CS0 space) and a memory bus width of 32 bits. The MMU function is not used.

Note: The tutorial program is compiled with `C:\HDI_CE\TUTORIAL`. When installing a tutorial file into a directory other than `C:\HDI_CE\TUTORIAL`, recompile the tutorial program. `sort.abs` operates in big endian. `sort.abs` must be recompiled to operate in little endian.

3.2 Running the HDI

To run the HDI, select the [HDI for SHxxxx E10A Emulator] from the [Start] menu.

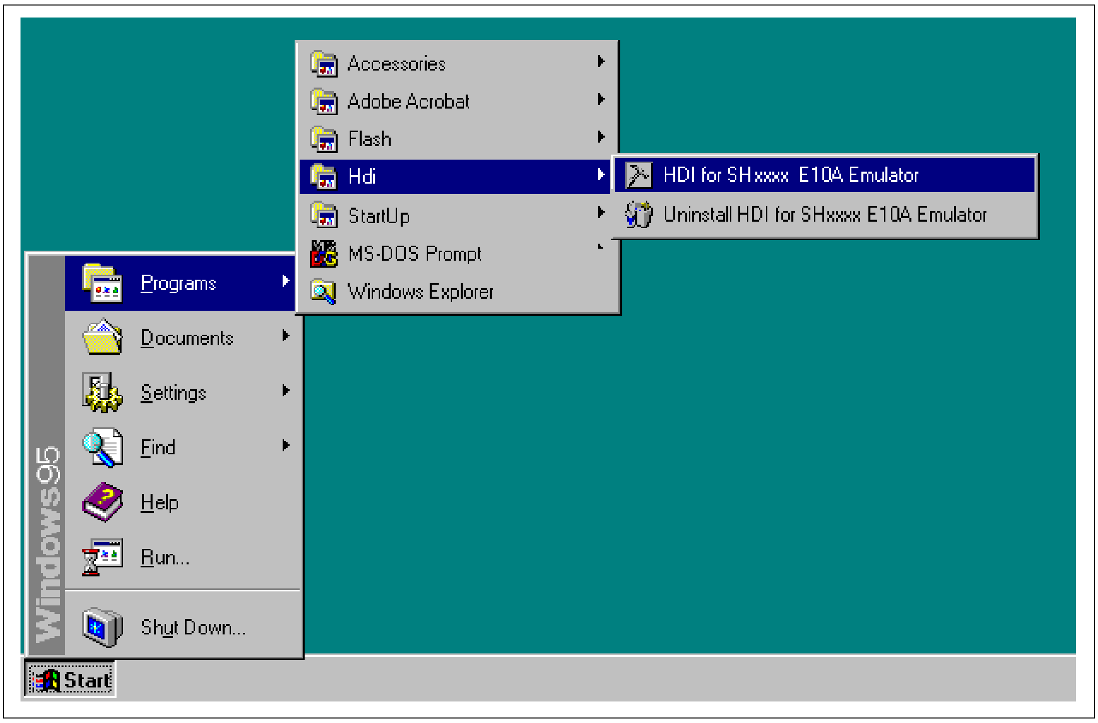
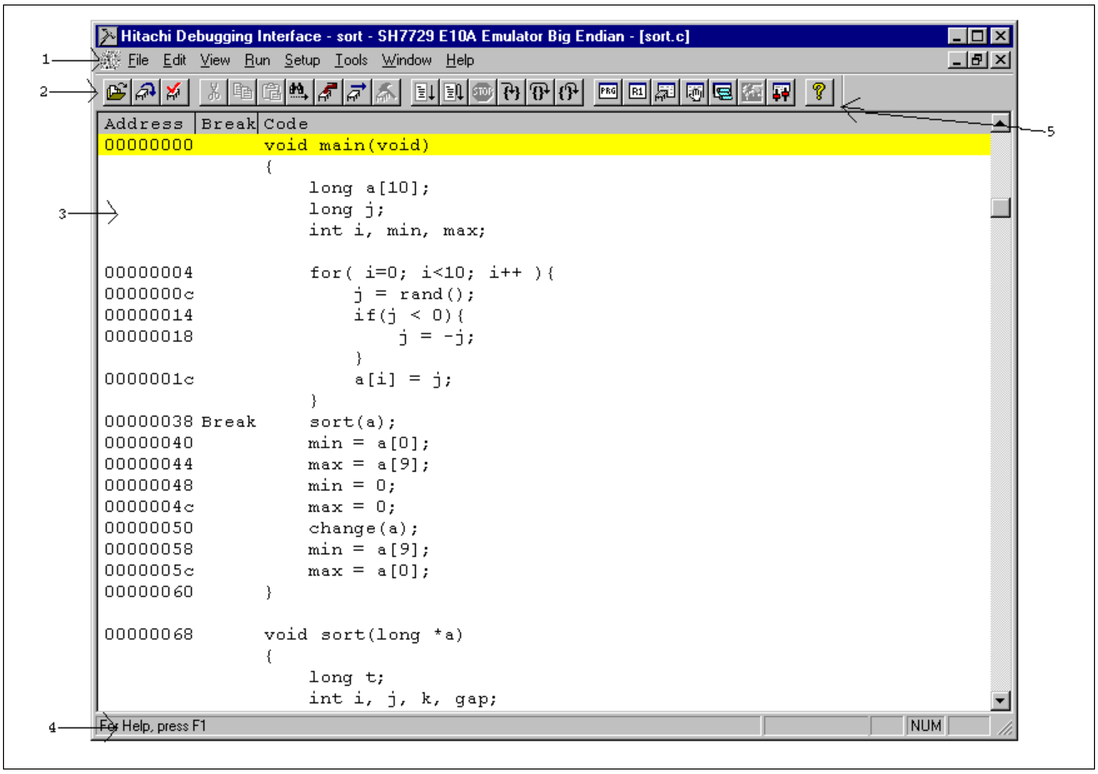


Figure 3.1 [Start] Menu

For the procedure of running the HDI, refer to section 2.5, System Check.

3.3 [HDI] Window



Hitachi Debugging Interface - sort - SH7729 E10A Emulator Big Endian - [sort.c]
File Edit View Run Setup Tools Window Help
Address Break Code
For Help, press F1
NUM
1
2
3
4
5

```
00000000        void main(void)
                {
                    long a[10];
                    long j;
                    int i, min, max;

00000004            for( i=0; i<10; i++ ){
0000000c                j = rand();
00000014                if(j < 0){
00000018                    j = -j;
                        }
0000001c                a[i] = j;
                    }
00000038 Break      sort(a);
00000040            min = a[0];
00000044            max = a[9];
00000048            min = 0;
0000004c            max = 0;
00000050            change(a);
00000058            min = a[9];
0000005c            max = a[0];
00000060        }

00000068        void sort(long *a)
                {
                    long t;
                    int i, j, k, gap;
```

Figure 3.2 [HDI] Window

The key functions of the HDI are described in section 4, Descriptions of Windows. Numbers in figure 3.2 indicate the following:

1. Menu bar: Gives the user access to the HDI commands for using the HDI debugger.

2. Toolbar: Provides convenient buttons as shortcuts for the most frequently used menu commands.

3. Program window: Displays the source program being debugged.

4. Status bar: Displays the status of the emulator, and progress information about downloading.

5. [Help] button: Activates context sensitive help about any features of the HDI user interface.

3.4 Setting up the Emulator

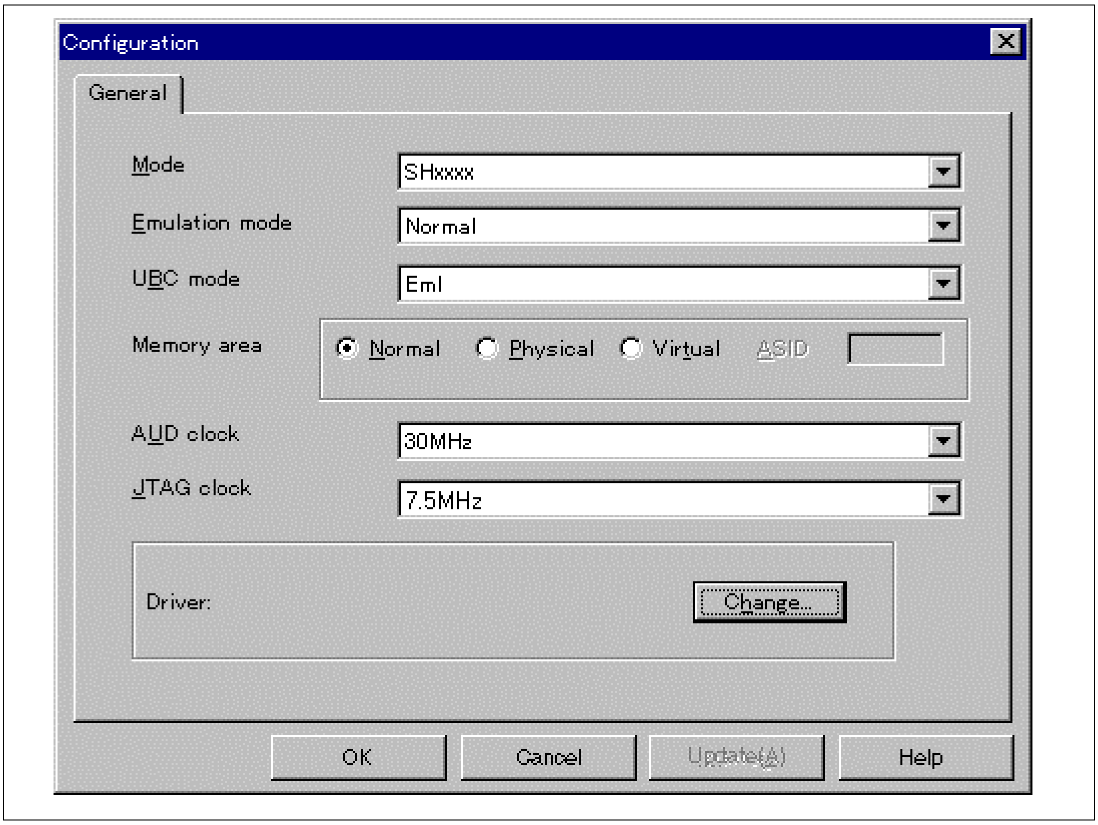
The following MCU conditions must be set up on the emulator before downloading the program:

- Device type
- Execution mode

The following describes how to set up the emulator for the tutorial programs.

3.5 Setting the [Configuration] Dialog Box

- Select [Configure Platform...] from the [Setup] menu to set configuration. The [Configuration] dialog box is displayed.



Configuration

General

Mode: SHxxxx

Emulation mode: Normal

UBC mode: Eml

Memory area: [✓ Normal] Physical Virtual ASID

AUD clock: 30MHz

JTAG clock: 7.5MHz

Driver:

Figure 3.3 [Configuration] Dialog Box

Set options as follows:

Table 3.2 Setting the [Configuration] Dialog Box

Option	Value
Mode	SHxxxx (default)
Emulation mode	Normal (normal execution, default)
UBC mode	Eml (default)
Memory area (address setting of memory space)	Normal (default)
AUD clock	30 MHz (default)
JTAG clock	7.5 MHz (default)

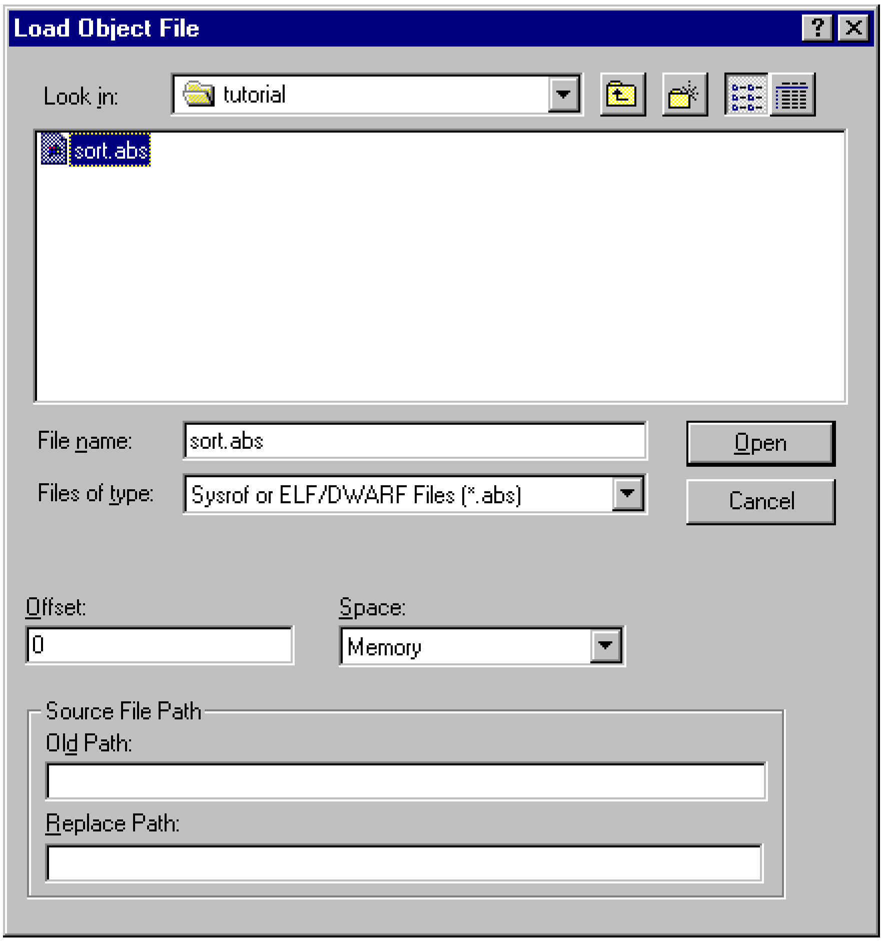
- Click the [OK] button to set any changes in the configuration.

3.6 Downloading the Tutorial Program

3.6.1 Downloading the Tutorial Program

Download the object program to be debugged.

- Select [Load Program...] from the [File] menu. The [Load Object File] dialog box is displayed.
- Select the file `sort.abs` in the directory, and click the [Open] button.



File name: sort.abs

Files of type: Sysrof or ELF/DWARF Files (*.abs)

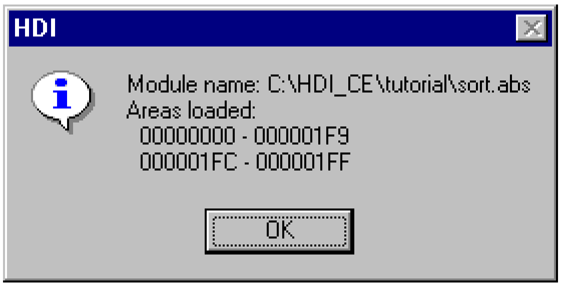
Offset: 0

Space: Memory

Figure 3.4 [Load Object File] Dialog Box

HITACHI

When the file has been loaded, the following dialog box displays information about the memory areas that have been filled with the program code.



Module name: C:\HDI_CE\tutorial\sort.abs
Areas loaded:
  00000000 - 000001F9
  000001FC - 000001FF

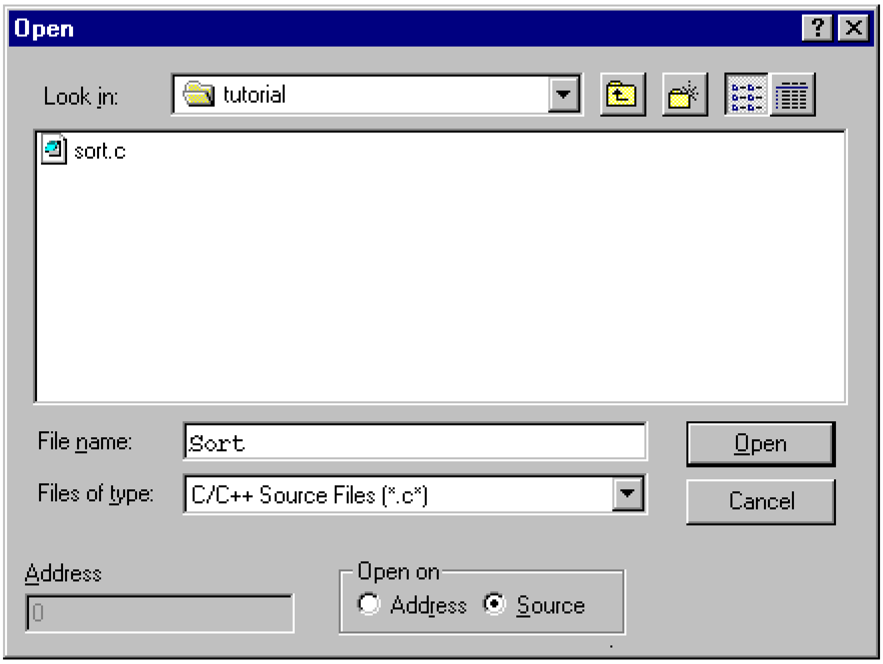
Figure 3.5 [HDI] Dialog Box

- Click the [OK] button to continue.

3.6.2 Displaying the Source Program

The HDI allows the user to debug a program at the source level, so that the user can see a list of the C program alongside the machine code at debugging. To do this, the C source file that corresponds to the object file needs to be read.

- Select [Program Window...] from the [View] menu. The [Open] dialog box is displayed.
- Select the C source file that corresponds to the object file the user has loaded.



File name: Sort

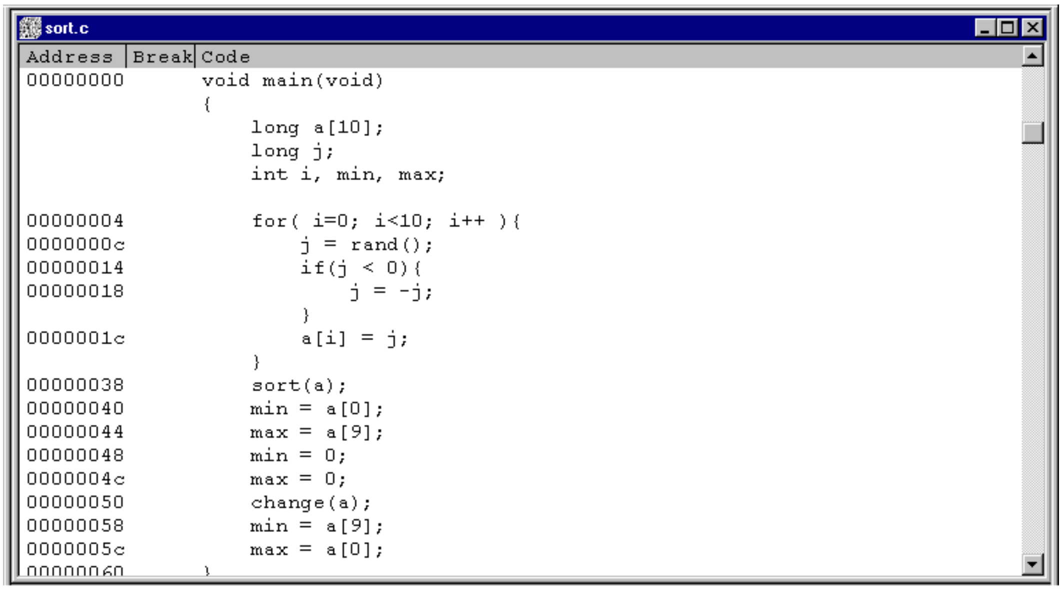
Files of type: C/C++ Source Files (*.c*)

Address: 0

Open on: ( ) Address [✓ Source]

Figure 3.6 [Open Program Window] Dialog Box

- Select [sort.c] and click the [Open] button. The [Program] window is displayed.



```
Address  |Break|Code
00000000       void main(void)
               {
                   long a[10];
                   long j;
                   int i, min, max;

00000004           for( i=0; i<10; i++ ){
0000000c               j = rand();
00000014               if(j < 0){
00000018                   j = -j;
                       }
0000001c               a[i] = j;
                   }
00000038           sort(a);
00000040           min = a[0];
00000044           max = a[9];
00000048           min = 0;
0000004c           max = 0;
00000050           change(a);
00000058           min = a[9];
0000005c           max = a[0];
00000060       }
```

Figure 3.7   [Program] Window (Displaying the Source Program)

- If necessary, select the [Font] option from the [Customise] submenu on the [Setup] menu to select a clear font and size.

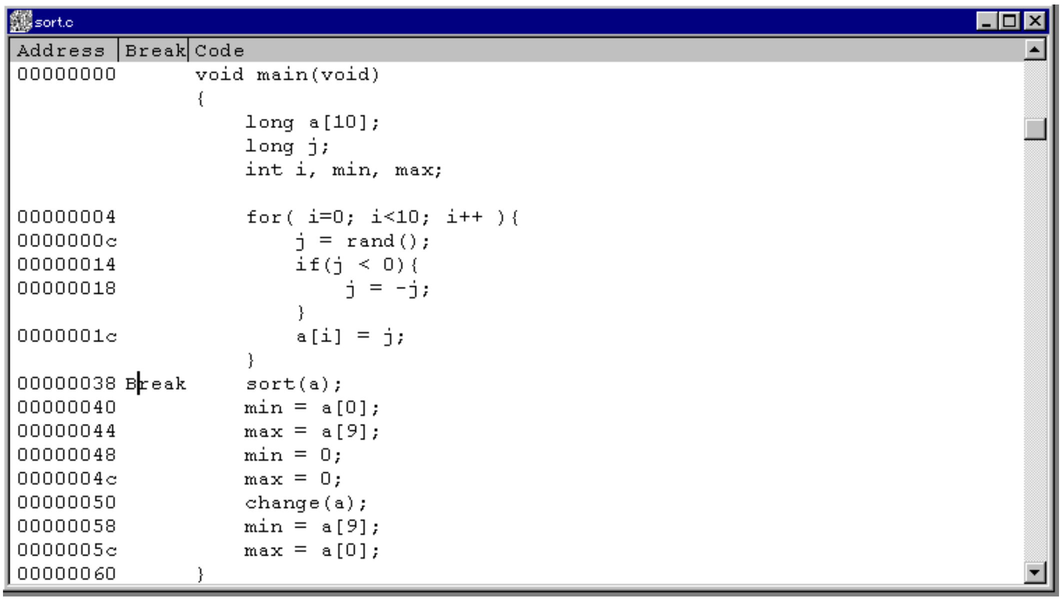
Initially the [Program] window shows the start of the main program, but the user can use the scroll bar to scroll through the program to see the other statements.

3.7 Setting the Software Breakpoint

A breakpoint is one of the easy debugging functions.

The [Program] window provides a very simple way of setting a software breakpoint at any point in a program. For example, to set a breakpoint at the `sort` function call:

- Select by double-clicking the [Break] column on the line containing the `sort` function call.



```
Address  Break Code
00000000       void main(void)
               {
                   long a[10];
                   long j;
                   int i, min, max;

00000004           for( i=0; i<10; i++ ){
0000000c               j = rand();
00000014               if(j < 0){
00000018                   j = -j;
                       }
0000001c               a[i] = j;
                   }
00000038 Break     sort(a);
00000040           min = a[0];
00000044           max = a[9];
00000048           min = 0;
0000004c           max = 0;
00000050           change(a);
00000058           min = a[9];
0000005c           max = a[0];
00000060       }
```

Figure 3.8 [Program] Window (Setting a Software Breakpoint)

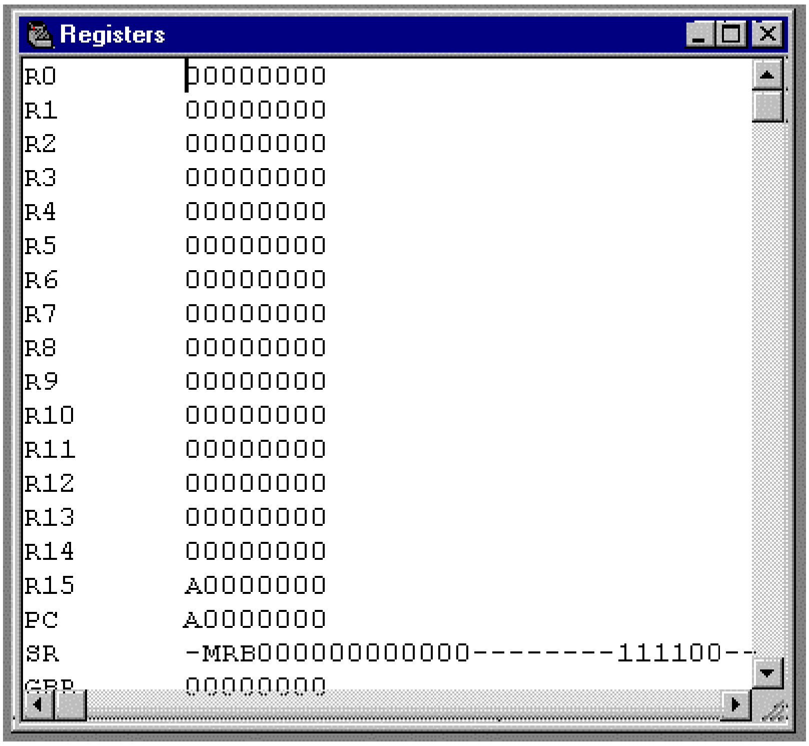
The word `Break` will be displayed on the line containing the `sort` function to show that a software breakpoint is set at that address.

Note: The software breakpoint cannot be set in the ROM area.

3.8 Setting Registers

Set values of the program counter and the stack pointer before executing the program.

- Select [Register Window] from the [View] menu. The [Registers] window is displayed.



```
R0          00000000
R1          00000000
R2          00000000
R3          00000000
R4          00000000
R5          00000000
R6          00000000
R7          00000000
R8          00000000
R9          00000000
R10         00000000
R11         00000000
R12         00000000
R13         00000000
R14         00000000
R15         A0000000
PC          A0000000
SR          -MRB000000000000--------111100--
GBR         00000000
```

Figure 3.9   [Registers] Window

- To change the value of the program counter (PC), move the mouse pointer to the value to be changed in the [PC] value area in the [Registers] window and enter the new value by the keyboard, or double-click the value area with the mouse. The following dialog box is then displayed.

HITACHI

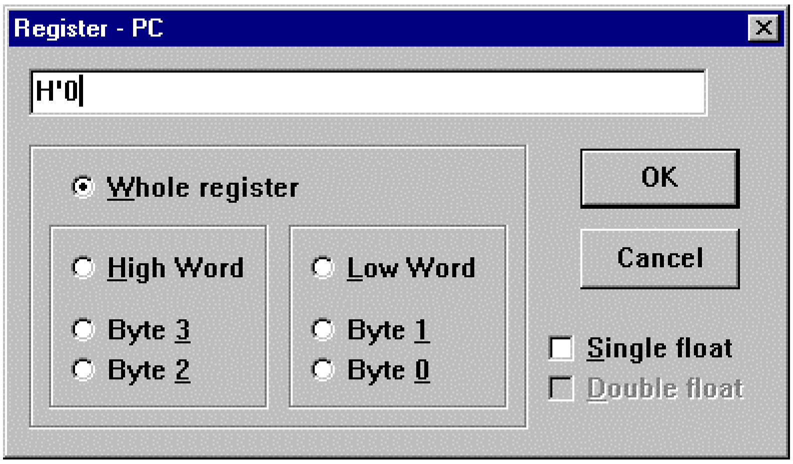


Figure 3.10 [Register] Dialog Box (PC)

- Set the program counter to H'0 in this tutorial program, and click the [OK] button.
- To change the value of the stack pointer (SP), double-click the mouse pointer on the value to be changed in the [R15] value area in the [Registers] window and enter the new value by the keyboard, or select the value area with the mouse. The following dialog box is then displayed.

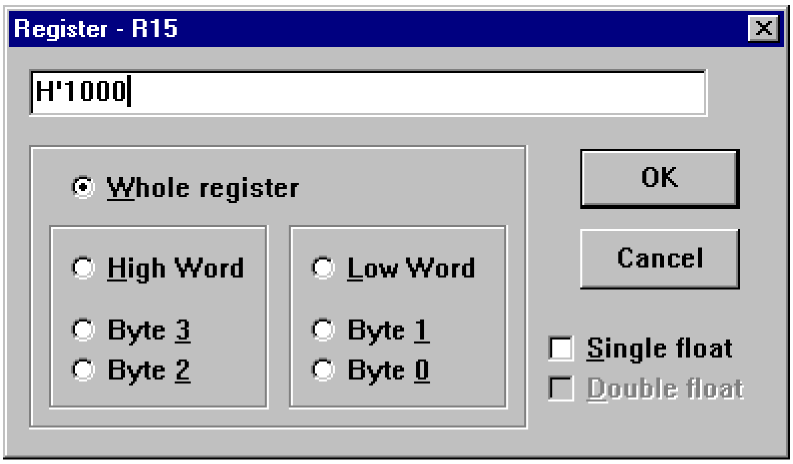


Figure 3.11 [Register] Dialog Box (SP)

- Set H'1000 for the value of the stack pointer in this tutorial program, and click the [OK] button.

3.9 Executing the Program

Execute the program as described in the following:

- To execute the program, select [Go] from the [Run] menu, or click the [Go] button on the toolbar.

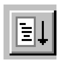
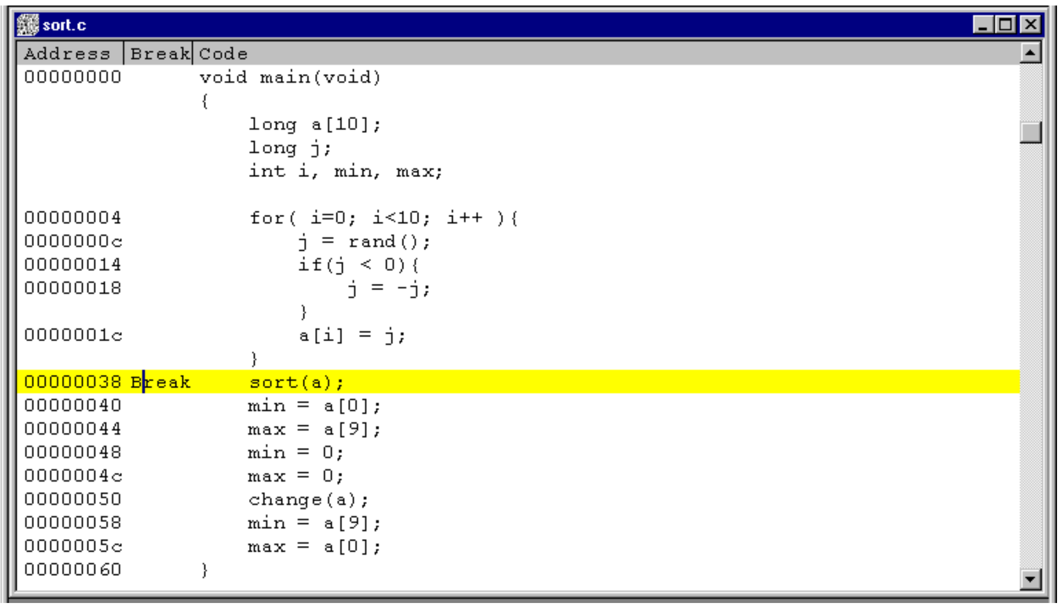


Figure 3.12 [Go] Button

The program will be executed up to the breakpoint that has been inserted, and a statement will be highlighted in the [Program] window to show the position that the program has halted, with the message [Break=BREAKPOINT] in the status bar.



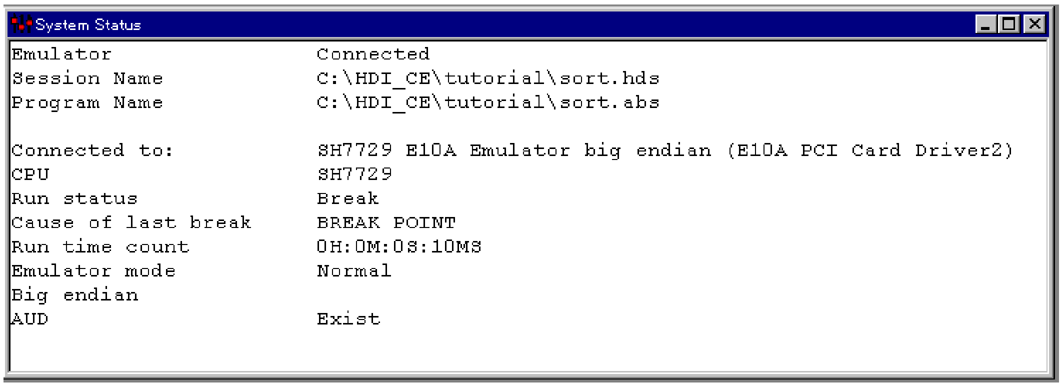
```
Address   Break Code
00000000        void main(void)
                {
                    long a[10];
                    long j;
                    int i, min, max;

00000004            for( i=0; i<10; i++ ){
0000000c                j = rand();
00000014                if(j < 0){
00000018                    j = -j;
                        }
0000001c                a[i] = j;
                    }
00000038  Break     sort(a);
00000040            min = a[0];
00000044            max = a[9];
00000048            min = 0;
0000004c            max = 0;
00000050            change(a);
00000058            min = a[9];
0000005c            max = a[0];
00000060        }
```

Figure 3.13 [Program] Window (Break Status)

HITACHI

The user can see the cause of the break that occurred last time in the [System Status] window.

- Select [Status Window] from the [View] menu.



```
Emulator                 Connected
Session Name             C:\HDI_CE\tutorial\sort.hds
Program Name             C:\HDI_CE\tutorial\sort.abs

Connected to:            SH7729 E10A Emulator big endian (E10A PCI Card Driver2)
CPU                      SH7729
Run status               Break
Cause of last break      BREAK POINT
Run time count           0H:0M:0S:10MS
Emulator mode            Normal
Big endian
AUD                      Exist
```

Figure 3.14 [System Status] Window

The [System Status] window displays the following items.

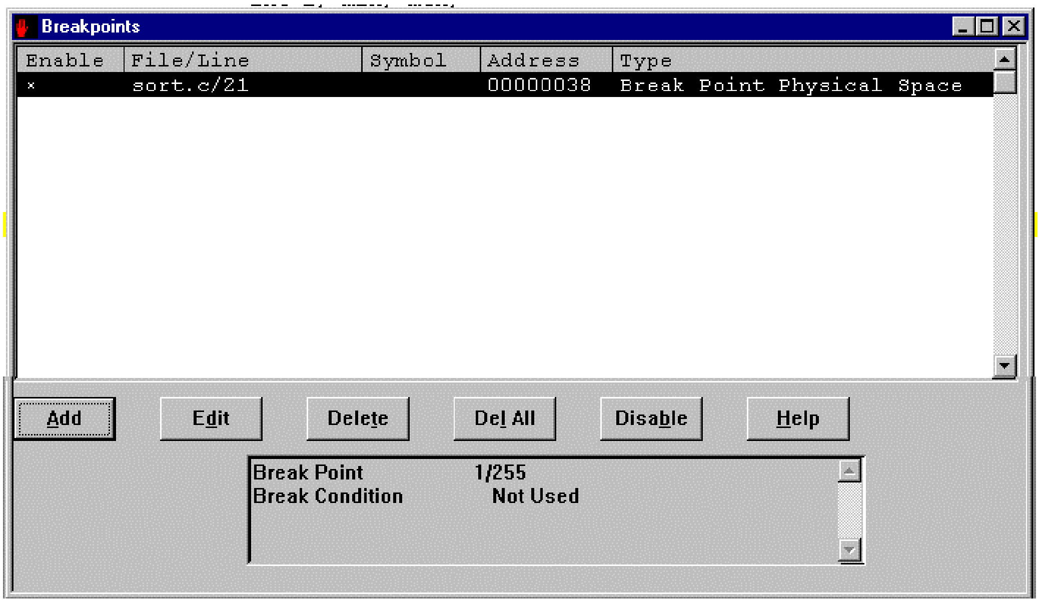
Table 3.3 Contents of the [System Status] Window

Item	Description
Emulator	Displays whether the emulator is connected.
Session Name	Displays the session file name.
Program Name	Displays the load module file name.
Connected To	Displays the name of the connected emulator (SH7729 E10A Emulator big endian (E10A PCI Card Driver2) for this example).
CPU	Displays the target CPU name (SH7729 for this example).
Run status	Displays the execution status: RUNNING: Being executed Break: Stopped
Cause of last break	Displays the cause of the emulator stopping at break. In this example, the cause of the stop is BREAK POINT.
Run time count	Displays the program execution time. The display format is H: hours, M: minutes, S: seconds, and MS: milliseconds. In this example, 0H:0M:0S:10MS is displayed.
Emulator mode	Displays the emulator operating mode (setting information for [Emulation Mode] of the [Configuration] dialog box).
Big Endian/Little Endian	Displays the endian state (Big Endian or Little Endian). In this example, Big Endian is displayed.
AUD	Displays whether the AUD function can be used. In this example, the AUD function can be used.

3.10 Reviewing Breakpoints

The user can see all the breakpoints set in the program in the [Breakpoints] window.

- Select [Breakpoint Window] from the [View] menu.



```
Enable  File/Line          Symbol    Address     Type
×       sort.c/21                    00000038    Break Point Physical Space
```

```
Break Point              1/255
Break Condition            Not Used
```

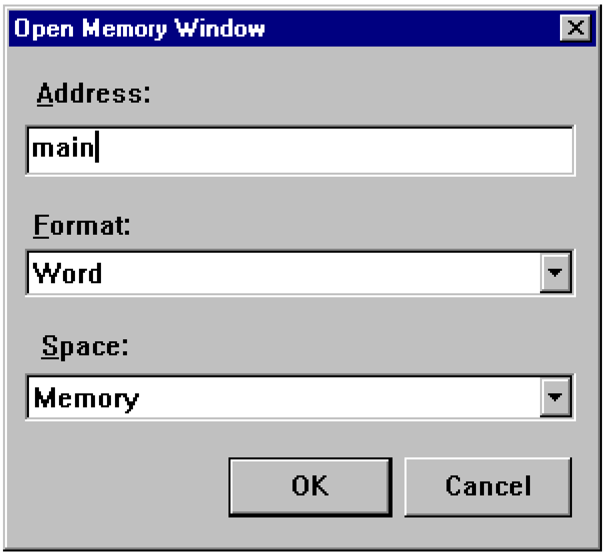
Figure 3.15 [Breakpoints] Window

The [Breakpoints] window also allows the user to set breakpoints, define new breakpoints, and delete breakpoints.

3.11 Viewing Memory

The user can view the contents of a memory block in the [Memory] window. For example, to view the memory contents corresponding to the `main` in word size:

- Select [Memory Window…] from the [View] menu, enter ***`main`*** in the [Address] edit box, and set `Word` in the [Format] combo box.



Open Memory Window

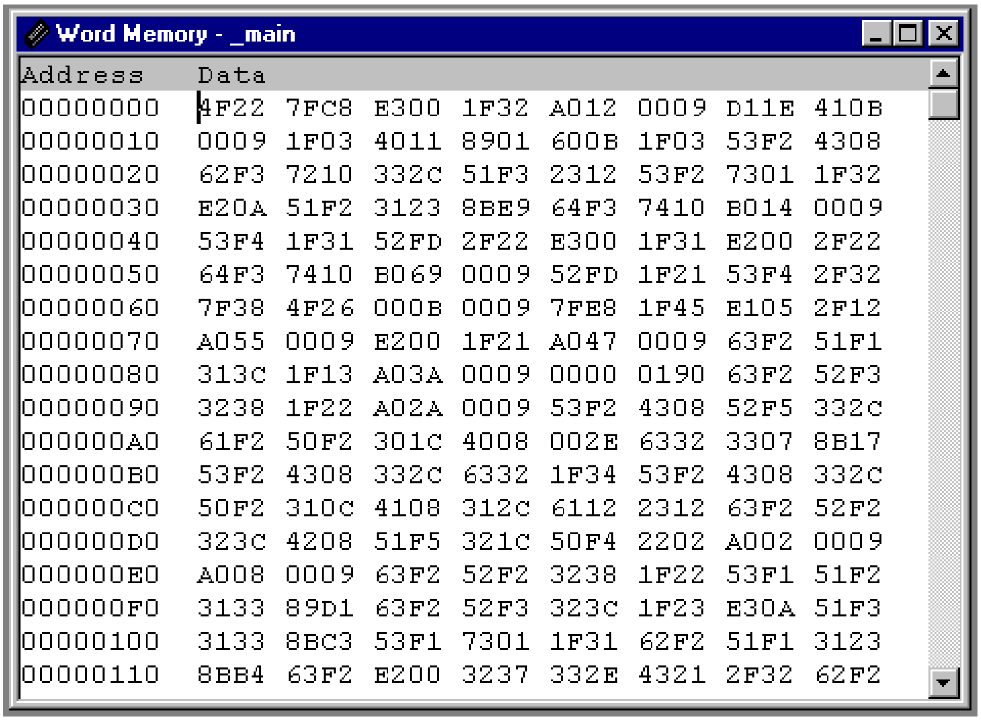
Address: main

Format: Word

Space: Memory

Figure 3.16 [Open Memory Window] Dialog Box

- Click the [OK] button. The [Word Memory] window showing the specified area of memory is displayed.



```
Address    Data
00000000   4F22 7FC8 E300 1F32 A012 0009 D11E 410B
00000010   0009 1F03 4011 8901 600B 1F03 53F2 4308
00000020   62F3 7210 332C 51F3 2312 53F2 7301 1F32
00000030   E20A 51F2 3123 8BE9 64F3 7410 B014 0009
00000040   53F4 1F31 52FD 2F22 E300 1F31 E200 2F22
00000050   64F3 7410 B069 0009 52FD 1F21 53F4 2F32
00000060   7F38 4F26 000B 0009 7FE8 1F45 E105 2F12
00000070   A055 0009 E200 1F21 A047 0009 63F2 51F1
00000080   313C 1F13 A03A 0009 0000 0190 63F2 52F3
00000090   3238 1F22 A02A 0009 53F2 4308 52F5 332C
000000A0   61F2 50F2 301C 4008 002E 6332 3307 8B17
000000B0   53F2 4308 332C 6332 1F34 53F2 4308 332C
000000C0   50F2 310C 4108 312C 6112 2312 63F2 52F2
000000D0   323C 4208 51F5 321C 50F4 2202 A002 0009
000000E0   A008 0009 63F2 52F2 3238 1F22 53F1 51F2
000000F0   3133 89D1 63F2 52F3 323C 1F23 E30A 51F3
00000100   3133 8BC3 53F1 7301 1F31 62F2 51F1 3123
00000110   8BB4 63F2 E200 3237 332E 4321 2F32 62F2
```

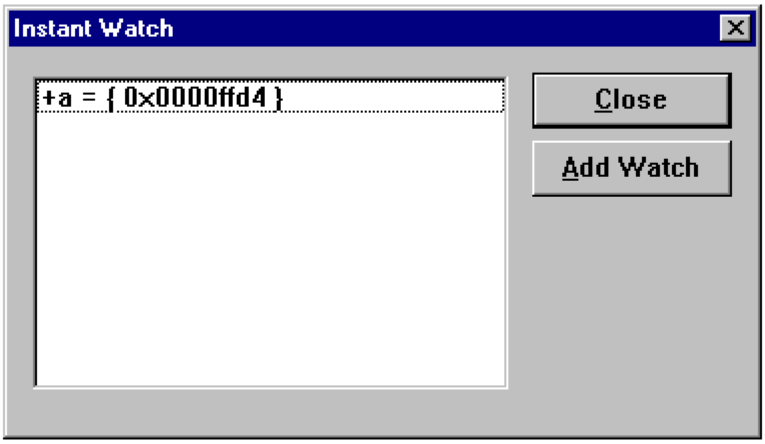
Figure 3.17 [Word Memory] Window

3.12 Watching Variables

As the user steps through a program, it is possible to watch that the values of variables used in the user program are changed. For example, set a watch on the long-type array a declared at the beginning of the program, by using the following procedure:

- Click the left of displayed array a in the [Program] window to position the cursor.
- Click the [Program] window with the right mouse button and select [Instant Watch...] from a pop-up menu.

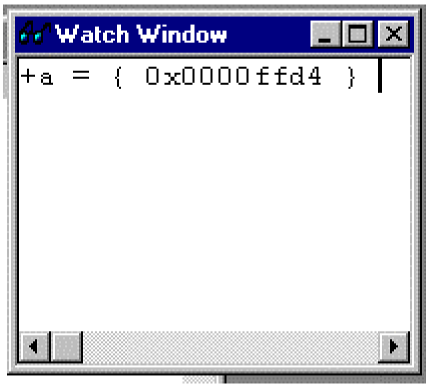
  The following dialog box will be displayed.



```
+a = { 0x0000ffd4 }
```

Figure 3.18   [Instant Watch] Dialog Box

- Click [Add Watch] button to add a variable to the [Watch] window.



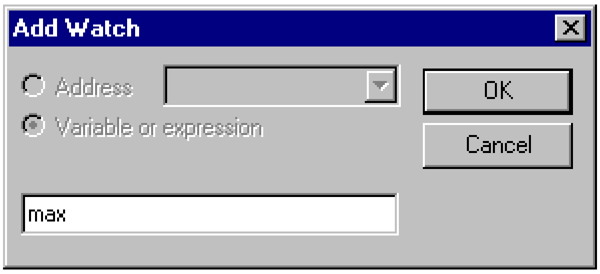
```
+a = { 0x0000ffd4 }
```

Figure 3.19 [Watch] Window (Displaying the Array)

The user can also add a variable to the [Watch] window by specifying its name.

- Click the [Watch] window with the right mouse button and select [Add Watch] from the pop-up menu.

  The following dialog box will be displayed.



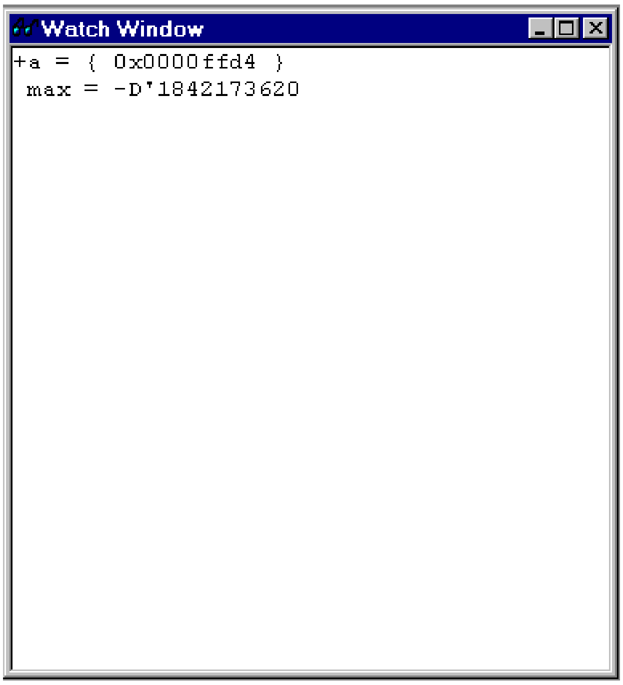
( ) Address
(●) Variable or expression

max

Figure 3.20 [Add Watch] Dialog Box

- Input variable ***max*** and click the [OK] button.

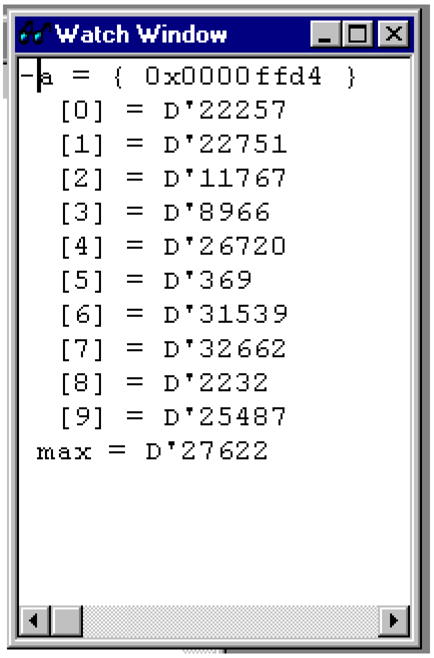
The [Watch] window will now also show the long-type variable `max`.



```
+a = { 0x0000ffd4 }
 max = -D'1842173620
```

Figure 3.21 [Watch] Window (Displaying the Variable)

The user can double-click the + symbol to the left of any variable in the [Watch] window to watch the all elements in array a.



```
-a = { 0x0000ffd4 }
   [0] = D'22257
   [1] = D'22751
   [2] = D'11767
   [3] = D'8966
   [4] = D'26720
   [5] = D'369
   [6] = D'31539
   [7] = D'32662
   [8] = D'2232
   [9] = D'25487
 max = D'27622
```

Figure 3.22 [Watch] Window (Displaying Array Elements)

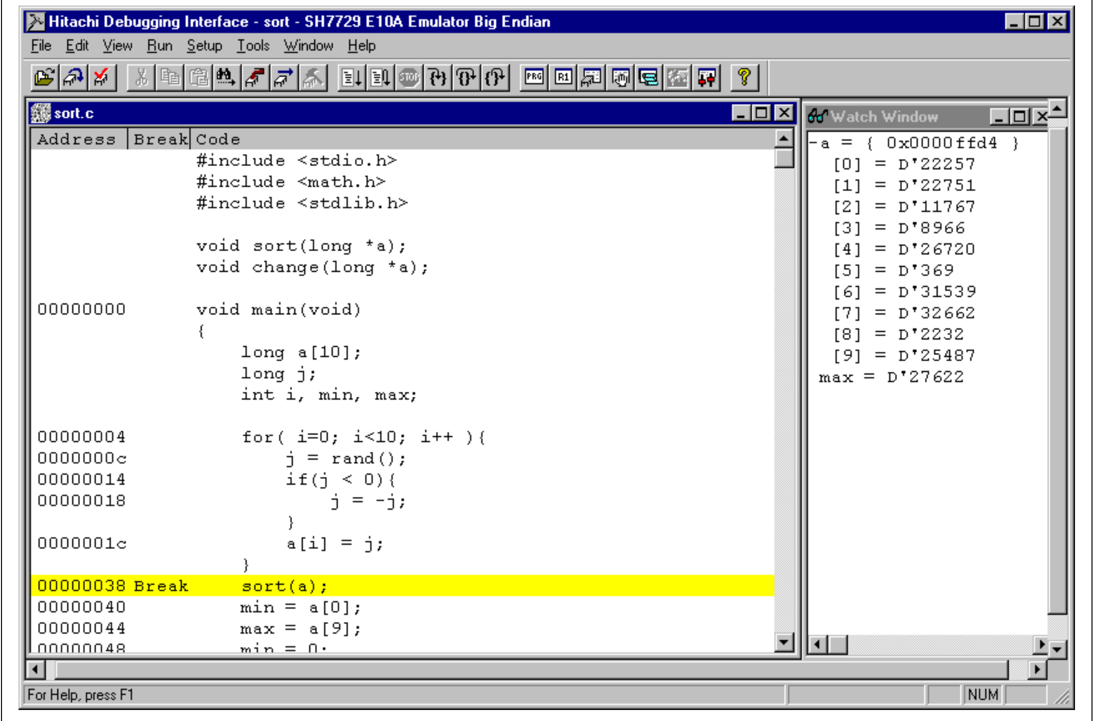
3.13 Stepping Through a Program

The HDI provides a range of step menu commands that allow efficient program debugging.

Table 3.4 Step Option

Menu Command	Description
Step In	Executes each statement, including statements within functions.
Step Over	Executes a function call in a single step.
Step Out	Steps out of a function, and stops at the statement following the statement that called the function.
Step…	Steps the specified times repeatedly at a specified rate.

Execute the step commands to confirm that the `sort` function statement at address H'38 has been executed.


Hitachi Debugging Interface - sort - SH7729 E10A Emulator Big Endian
File Edit View Run Setup Tools Window Help
sort.c
Watch Window
For Help, press F1
NUM

```
Address   Break Code
                #include <stdio.h>
                #include <math.h>
                #include <stdlib.h>

                void sort(long *a);
                void change(long *a);

00000000        void main(void)
                {
                    long a[10];
                    long j;
                    int i, min, max;

00000004            for( i=0; i<10; i++ ){
0000000c                j = rand();
00000014                if(j < 0){
00000018                    j = -j;
                        }
0000001c                a[i] = j;
                    }
00000038 Break      sort(a);
00000040            min = a[0];
00000044            max = a[9];
00000048            min = 0;
```

```
-a = { 0x0000ffd4 }
  [0] = D'22257
  [1] = D'22751
  [2] = D'11767
  [3] = D'8966
  [4] = D'26720
  [5] = D'369
  [6] = D'31539
  [7] = D'32662
  [8] = D'2232
  [9] = D'25487
 max = D'27622
```

Figure 3.23 [Program] Window (Step Execution)

3.13.1 Executing [Step In] Command

The [Step In] steps into the called function and stops at the first statement of the called function.

- To step through the `sort` function, select [Step In] from the [Run] menu, or click the [Step In] button in the toolbar.

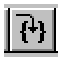
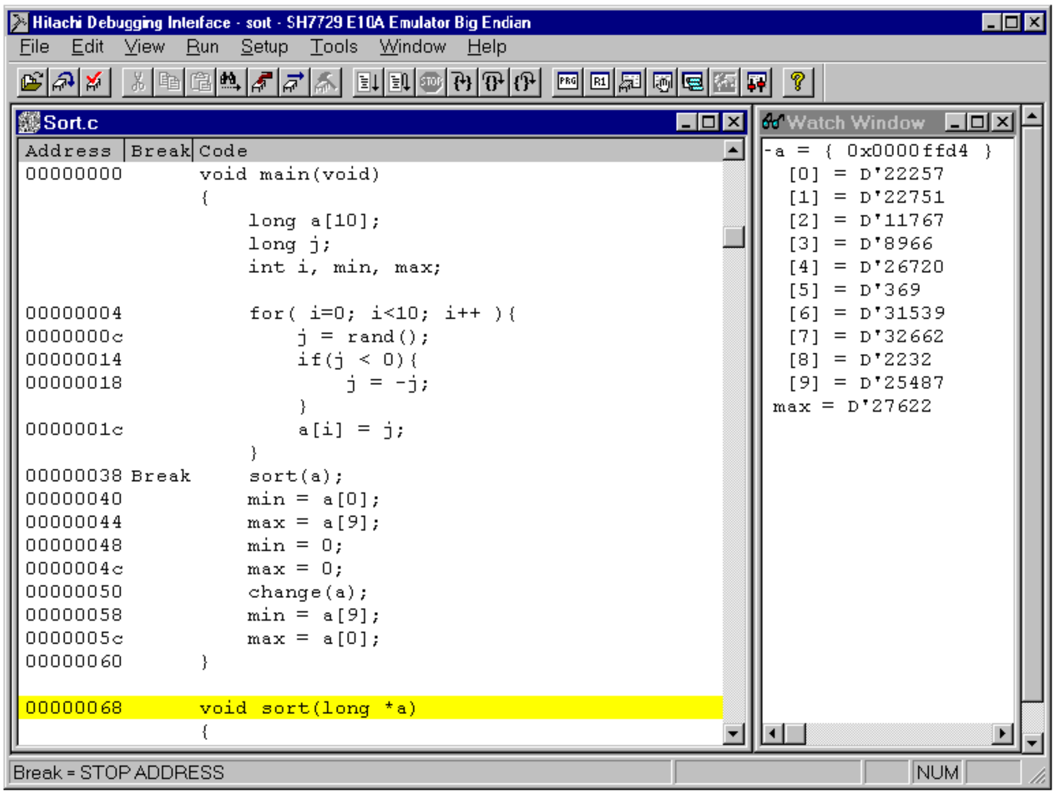


Figure 3.24 [Step In] Button


Hitachi Debugging Interface - sort - SH7729 E10A Emulator Big Endian
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window

```
Address  Break Code
00000000       void main(void)
               {
                   long a[10];
                   long j;
                   int i, min, max;

00000004           for( i=0; i<10; i++ ){
0000000c               j = rand();
00000014               if(j < 0){
00000018                   j = -j;
                       }
0000001c               a[i] = j;
                   }
00000038 Break     sort(a);
00000040           min = a[0];
00000044           max = a[9];
00000048           min = 0;
0000004c           max = 0;
00000050           change(a);
00000058           min = a[9];
0000005c           max = a[0];
00000060       }

00000068       void sort(long *a)
               {
```

```
-a = { 0x0000ffd4 }
   [0] = D'22257
   [1] = D'22751
   [2] = D'11767
   [3] = D'8966
   [4] = D'26720
   [5] = D'369
   [6] = D'31539
   [7] = D'32662
   [8] = D'2232
   [9] = D'25487
 max = D'27622
```

```
Break = STOP ADDRESS                    NUM
```

Figure 3.25 [Program] Window (Step In)

- The highlighted line moves to the first statement of the `sort` function in the [Program] window.

HITACHI

3.13.2 Executing [Step Out] Command

The [Step Out] steps out of the called function and stops at the next statement of the calling statement in the main function.

- To step out of the `sort` function, select [Step Out] from the [Run] menu, or click the [Step Out] button in the toolbar.

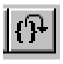
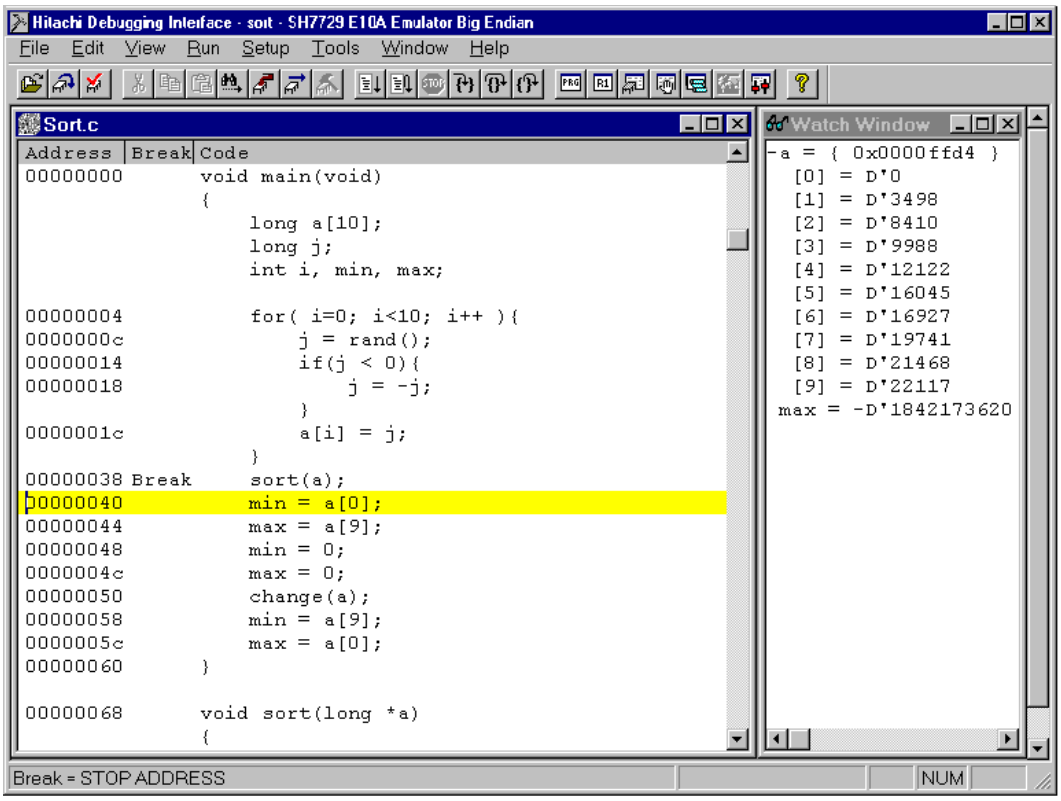


Figure 3.26 [Step Out] Button


Hitachi Debugging Interface - sort - SH7729 E10A Emulator Big Endian
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window
NUM

```
Address  Break Code
00000000        void main(void)
                {
                    long a[10];
                    long j;
                    int i, min, max;

00000004            for( i=0; i<10; i++ ){
0000000c                j = rand();
00000014                if(j < 0){
00000018                    j = -j;
                        }
0000001c                a[i] = j;
                    }
00000038 Break      sort(a);
00000040            min = a[0];
00000044            max = a[9];
00000048            min = 0;
0000004c            max = 0;
00000050            change(a);
00000058            min = a[9];
0000005c            max = a[0];
00000060        }

00000068        void sort(long *a)
                {
```

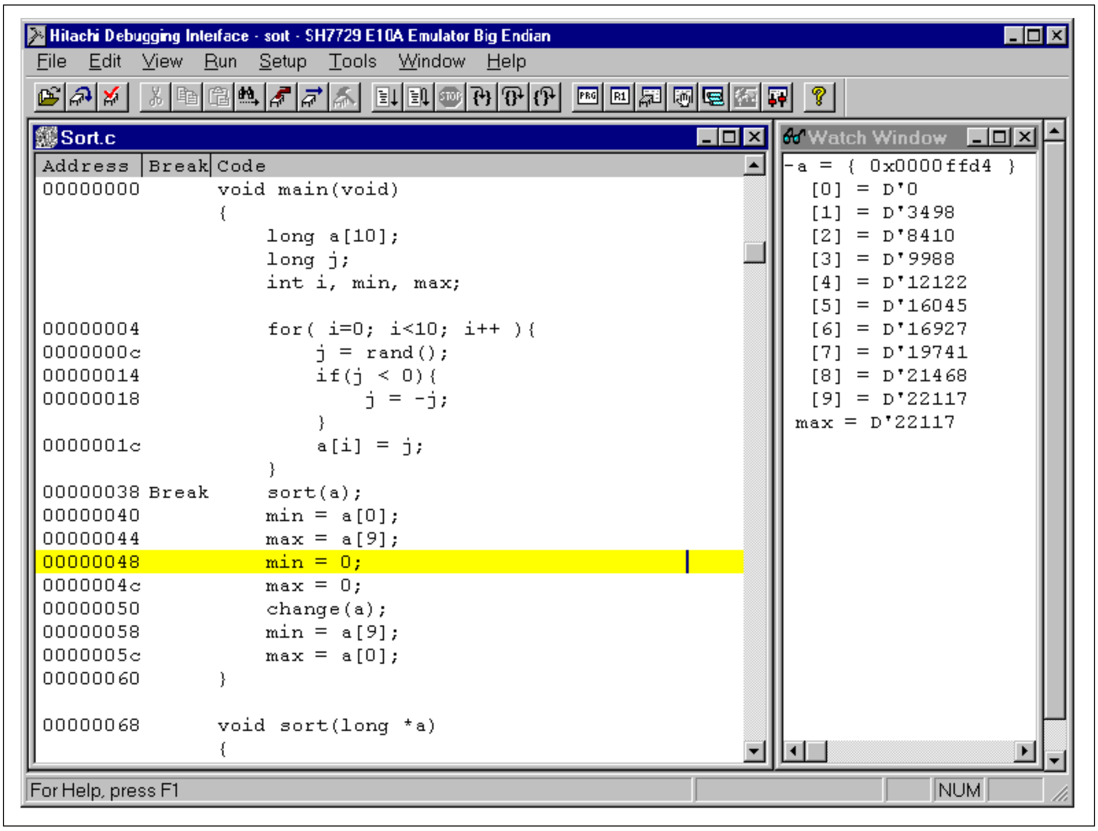
```
-a = { 0x0000ffd4 }
   [0] = D'0
   [1] = D'3498
   [2] = D'8410
   [3] = D'9988
   [4] = D'12122
   [5] = D'16045
   [6] = D'16927
   [7] = D'19741
   [8] = D'21468
   [9] = D'22117
 max = -D'1842173620
```

Break = STOP ADDRESS

Figure 3.27 [Program] Window (Step Out)

- The data of variable `a` displayed in the [Watch] window is sorted in ascending order.

- To execute two steps, use [Step In] twice.



Hitachi Debugging Interface - sort - SH7729 E10A Emulator Big Endian
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window
For Help, press F1
NUM

```
Address  Break Code
00000000       void main(void)
               {
                   long a[10];
                   long j;
                   int i, min, max;

00000004           for( i=0; i<10; i++ ){
0000000c               j = rand();
00000014               if(j < 0){
00000018                   j = -j;
                       }
0000001c               a[i] = j;
                   }
00000038 Break     sort(a);
00000040           min = a[0];
00000044           max = a[9];
00000048           min = 0;
0000004c           max = 0;
00000050           change(a);
00000058           min = a[9];
0000005c           max = a[0];
00000060       }

00000068       void sort(long *a)
               {
```

```
-a = { 0x0000ffd4 }
   [0] = D'0
   [1] = D'3498
   [2] = D'8410
   [3] = D'9988
   [4] = D'12122
   [5] = D'16045
   [6] = D'16927
   [7] = D'19741
   [8] = D'21468
   [9] = D'22117
 max = D'22117
```

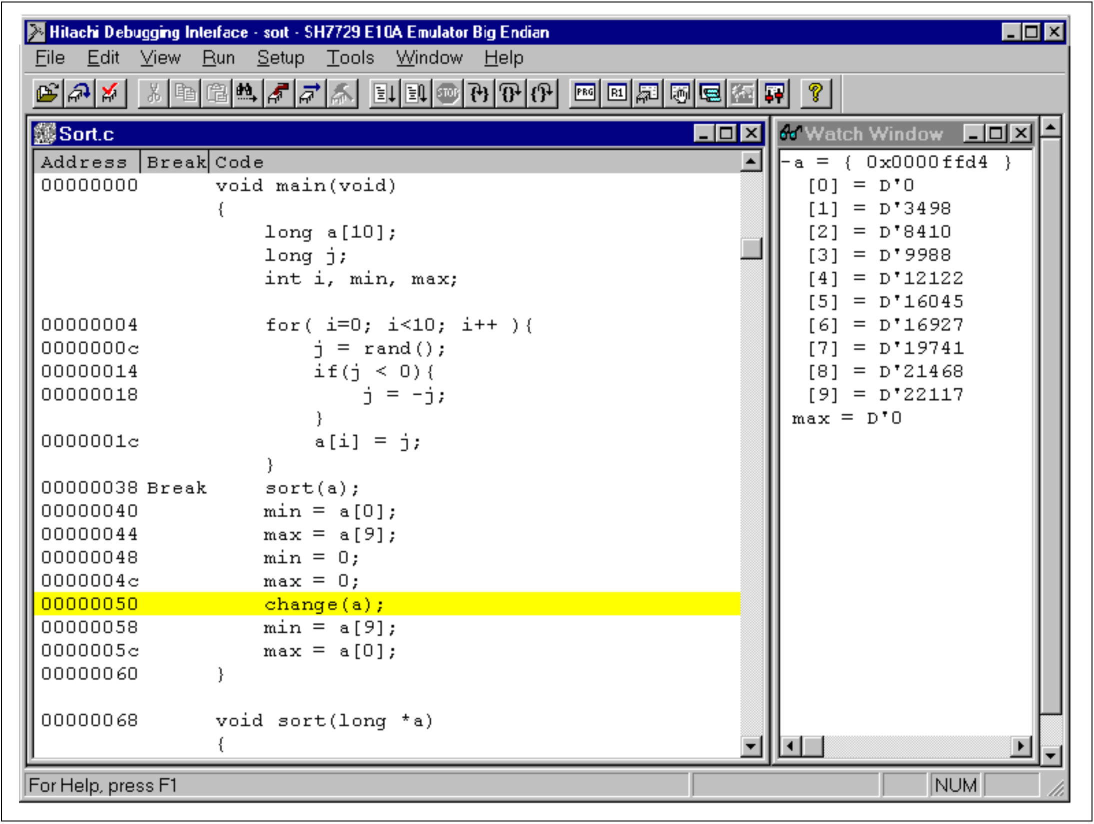
Figure 3.28 [Program] Window (Step In –> Step In)

- The value of `max` displayed in the [Watch] window is changed to the maximum data value.

3.13.3 Executing [Step Over] Command

The [Step Over] executes a function call as a single step and stops at the next statement of the main program.

- Using [Step Over], execute two steps to reach the change function statement.



Hitachi Debugging Interface - sort - SH7729 E10A Emulator Big Endian
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window
For Help, press F1
NUM

```
Address  Break Code
00000000       void main(void)
               {
                   long a[10];
                   long j;
                   int i, min, max;

00000004           for( i=0; i<10; i++ ){
0000000c               j = rand();
00000014               if(j < 0){
00000018                   j = -j;
                       }
0000001c               a[i] = j;
                   }
00000038 Break     sort(a);
00000040           min = a[0];
00000044           max = a[9];
00000048           min = 0;
0000004c           max = 0;
00000050           change(a);
00000058           min = a[9];
0000005c           max = a[0];
00000060       }

00000068       void sort(long *a)
               {
```

```
-a = { 0x0000ffd4 }
  [0] = D'0
  [1] = D'3498
  [2] = D'8410
  [3] = D'9988
  [4] = D'12122
  [5] = D'16045
  [6] = D'16927
  [7] = D'19741
  [8] = D'21468
  [9] = D'22117
 max = D'0
```

Figure 3.29 [Program] Window (Before Step Over Execution)

- To step through all statements in the change function at a single step, select [Step Over] from the [Run] menu, or click the [Step Over] button in the toolbar.

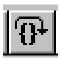
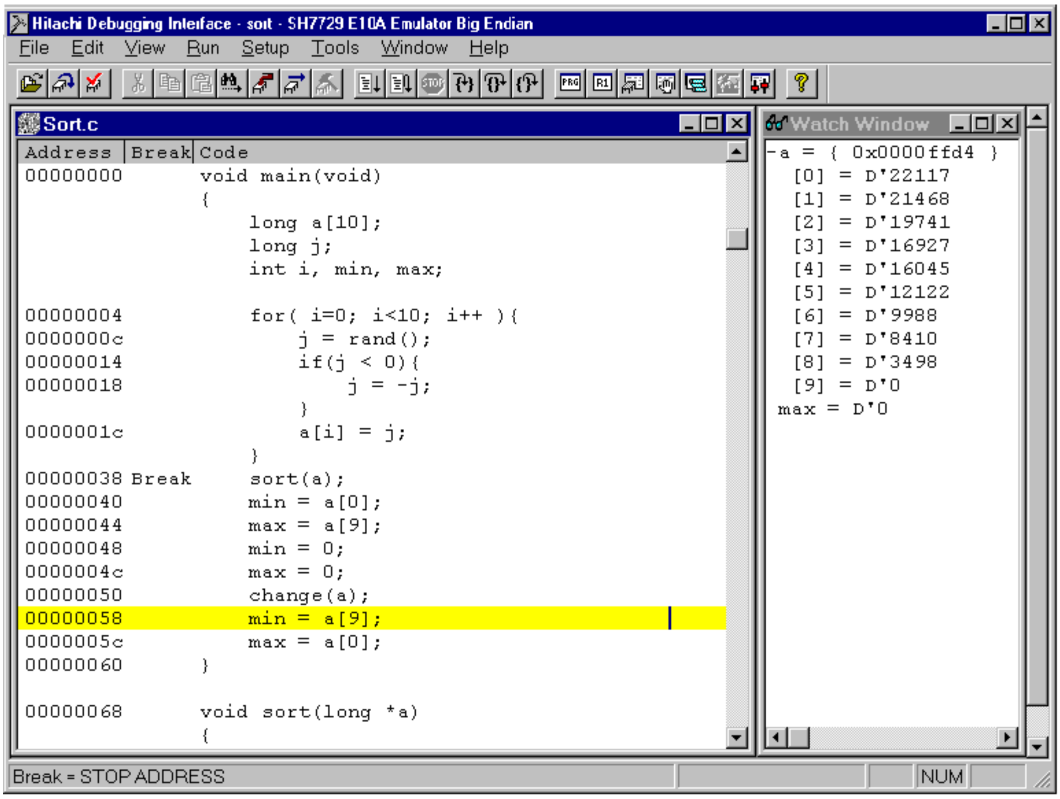


Figure 3.30 [Step Over] Button


Hitachi Debugging Interface - sort - SH7729 E10A Emulator Big Endian
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window

```
Address   Break Code
00000000        void main(void)
                {
                    long a[10];
                    long j;
                    int i, min, max;

00000004            for( i=0; i<10; i++ ){
0000000c                j = rand();
00000014                if(j < 0){
00000018                    j = -j;
                        }
0000001c                a[i] = j;
                    }
00000038 Break      sort(a);
00000040            min = a[0];
00000044            max = a[9];
00000048            min = 0;
0000004c            max = 0;
00000050            change(a);
00000058            min = a[9];
0000005c            max = a[0];
00000060        }

00000068        void sort(long *a)
                {
```

```
-a = { 0x0000ffd4 }
   [0] = D'22117
   [1] = D'21468
   [2] = D'19741
   [3] = D'16927
   [4] = D'16045
   [5] = D'12122
   [6] = D'9988
   [7] = D'8410
   [8] = D'3498
   [9] = D'0
 max = D'0
```

Break = STOP ADDRESS   NUM

Figure 3.31 [Program] Window (Step Over)

When the last statement of the change function is executed, the data of variable a, which is displayed in the [Watch] window, is sorted in descending order.

3.14 Displaying Local Variables

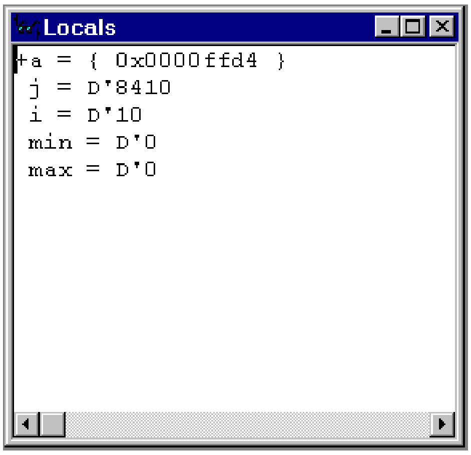
The user can display local variables in a function using the [Locals] window. For example, we will examine the local variables in the `main` function, which declares five local variables: `a`, `j`, `i`, `min`, and `max`.

- Select [Local Variable Window] from the [View] menu. The [Locals] window is displayed.

  Initially, the [Locals] window is empty because local variables have not yet been declared.

- Select [Step In] from the [Run] menu to execute a single step.

  The [Locals] window will now show the local variables and their values.



```
+a = { 0x0000ffd4 }
 j = D'8410
 i = D'10
 min = D'0
 max = D'0
```

Figure 3.32 [Locals] Window

- Double-click the + symbol in front of array `a` in the [Locals] window to display the elements of array `a`.
- Refer to the elements of array `a` before and after the execution of the `sort` function, and confirm that random data is sorted in descending order.

3.15 Break Function

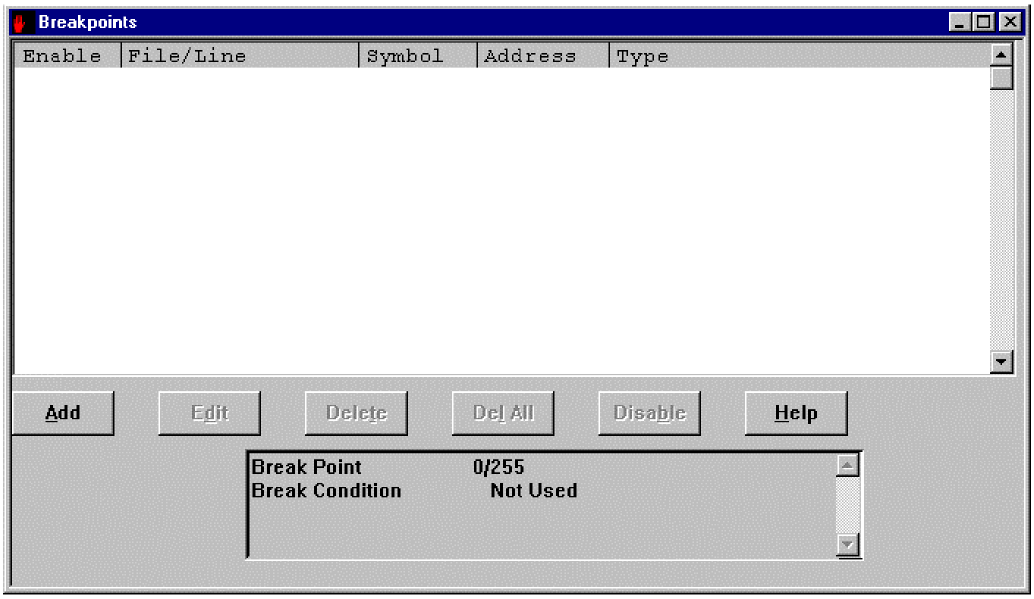
The emulator has software and hardware break functions. With the HDI, a software breakpoint can be set using the [Breakpoints] window, and a hardware break condition can be set using the [Break Condition 1,2,3] dialog box.

An overview and setting of the break function are described below.

3.15.1 Software Break Function

The emulator can set up to 255 software breakpoints. Setting a software breakpoint is described below.

- Select [Breakpoint Window] from the [View] menu. The [Breakpoints] window is displayed.
- Click the [Del All] button to cancel all the breakpoints that have been set.
- Click the [Add] button.



```
Break Point        0/255
Break Condition      Not Used
```

Figure 3.33 [Breakpoints] Window (Before Software Breakpoint Setting)

The [Break] dialog box is displayed. The [Point] page is displayed as a default.

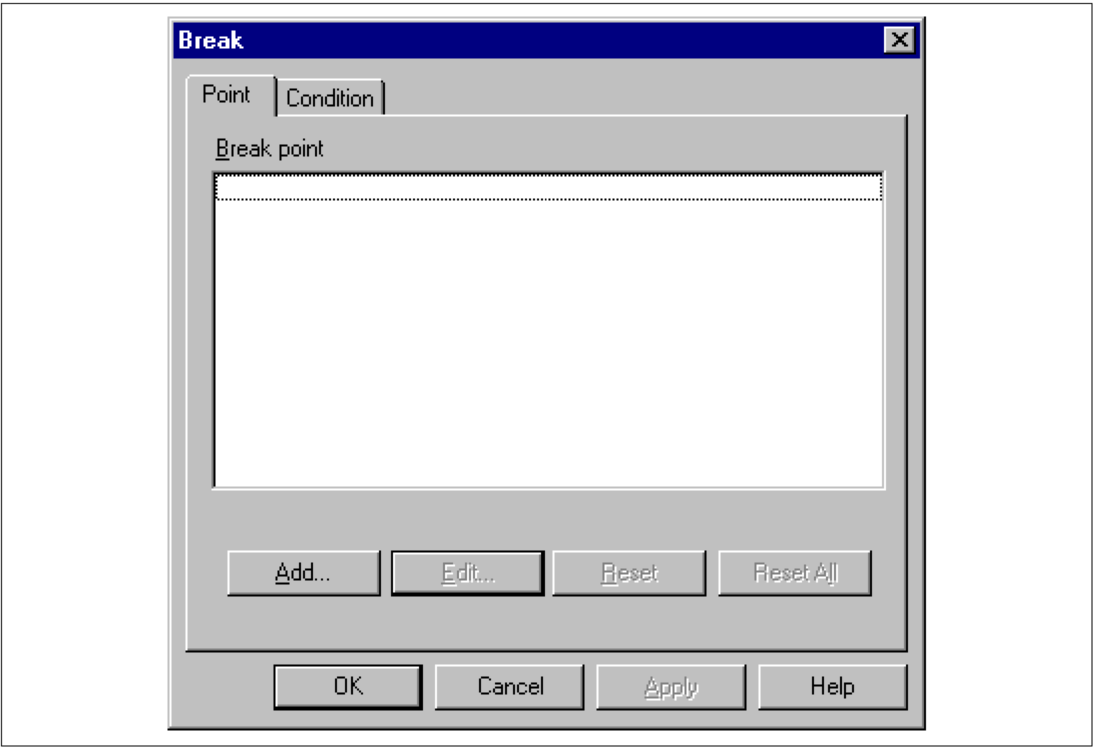
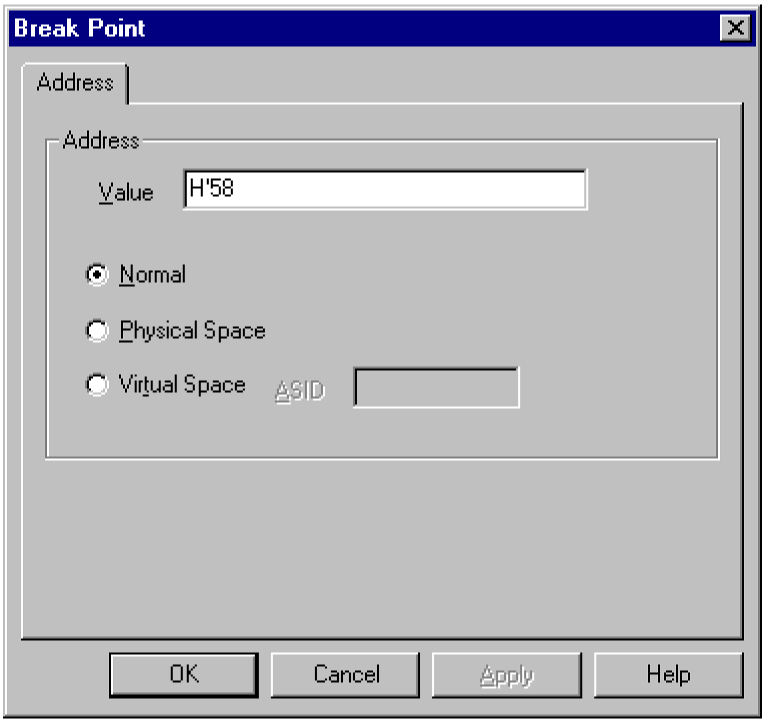


Figure 3.34   [Point] Page ([Break] Dialog Box)

- Click the [Add...] button to display the [Break Point] dialog box.
- Enter ***H'58*** to the [Value] edit box.



Address

Value: H'58

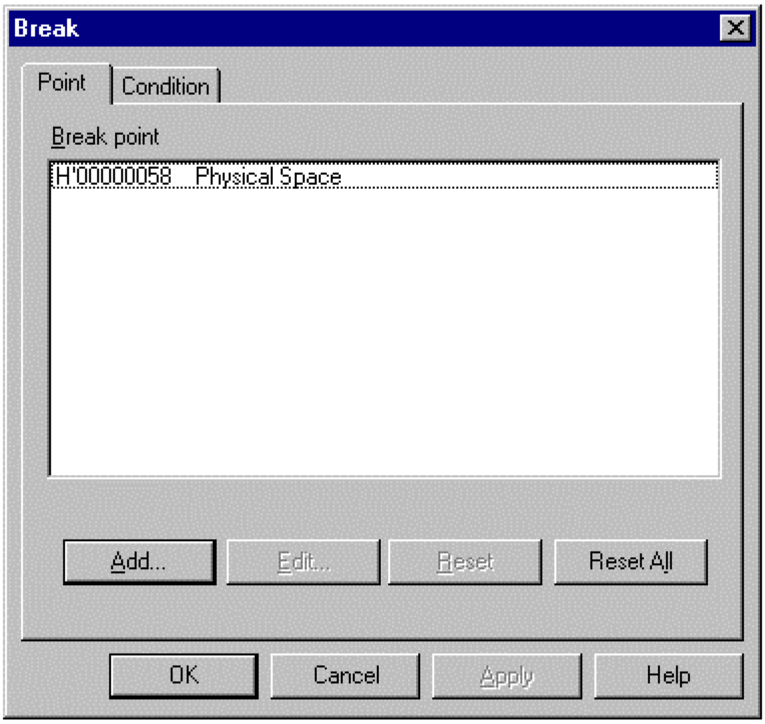
[✓ Normal] Physical Space Virtual Space

ASID:

Figure 3.35 [Break Point] Dialog Box

- Click the [OK] button.

The [Break] dialog box is displayed. The address set in the value field of [Break Point] and the memory space are displayed. "Physical Space" means that the breakpoint is set in the physical memory space.

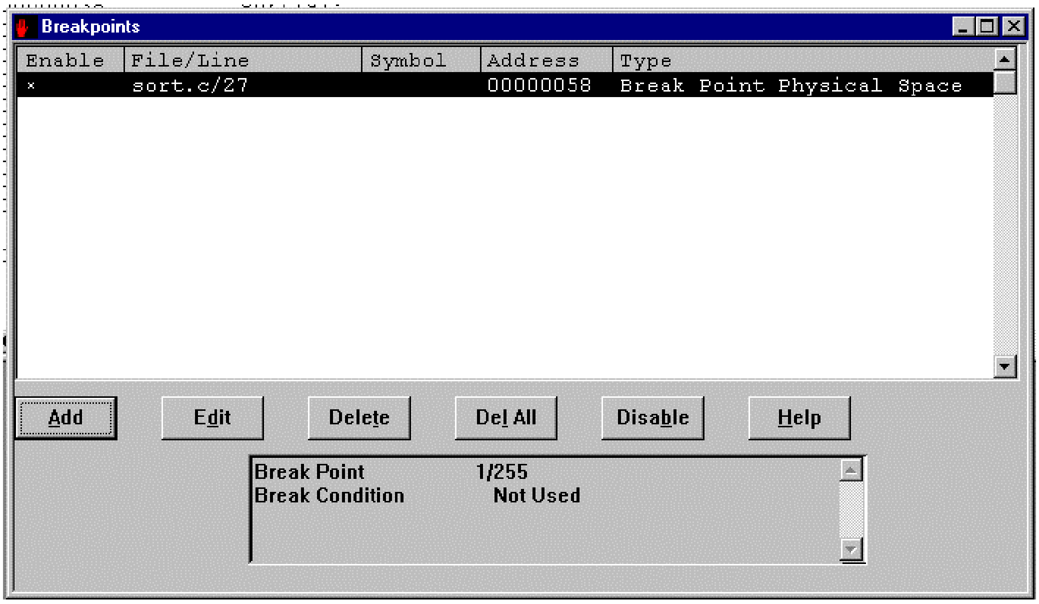


H'00000058   Physical Space

Figure 3.36   [Point] Page ([Break] Dialog Box) (After Software Breakpoint Setting)

- Click the [OK] button.
- Highlight the breakpoint in the [Breakpoints] window and click the [Delete] button.
- Close the [Breakpoints] window.

The software breakpoint that has been set is displayed in the [Breakpoints] window. The address space is displayed in [Type].



Enable	File/Line	Symbol	Address	Type
×	sort.c/27		00000058	Break Point Physical Space

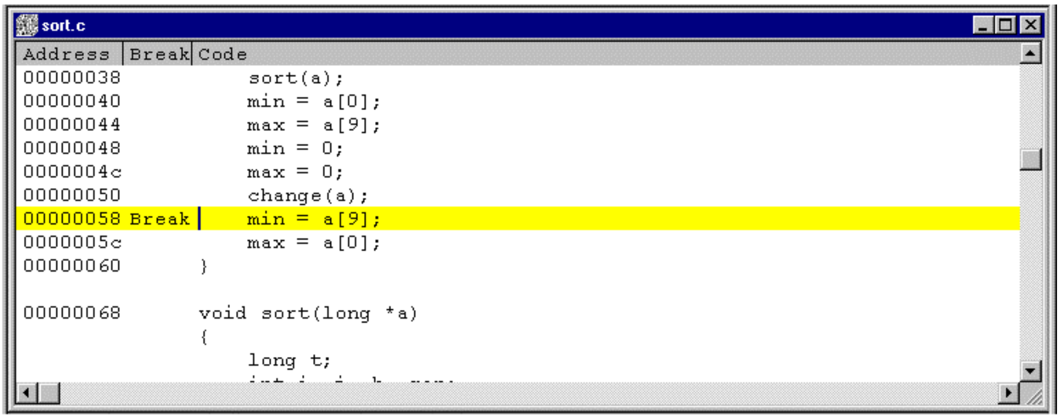
```
Break Point          1/255
Break Condition        Not Used
```

Figure 3.37   [Breakpoints] Window (Software Breakpoint Setting)

To delete breakpoints, the following procedure must be executed:

- Close the [Breakpoints] window.
- Set the program counter and stack pointer values that have been set in section 3.8, Setting Registers, in the [Registers] window. Click the [Go] button.

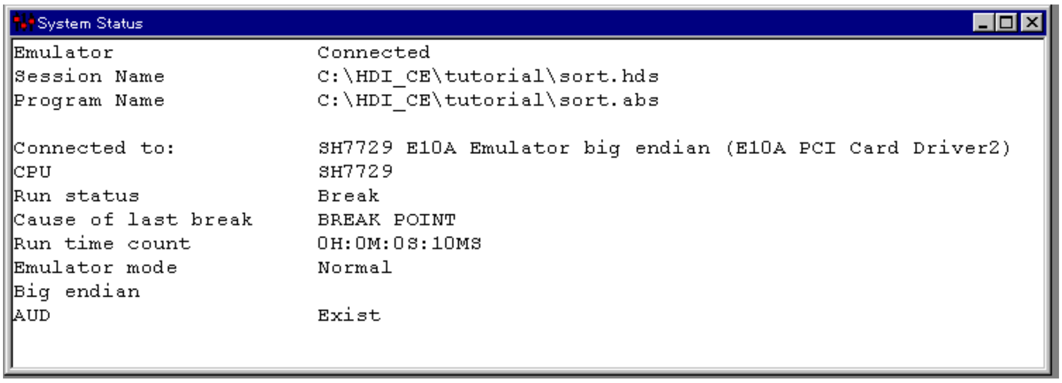
The program runs, and stops at the set breakpoint.



```
Address   Break Code
00000038            sort(a);
00000040            min = a[0];
00000044            max = a[9];
00000048            min = 0;
0000004c            max = 0;
00000050            change(a);
00000058 Break      min = a[9];
0000005c            max = a[0];
00000060        }

00000068        void sort(long *a)
                {
                    long t;
```

Figure 3.38 [Program] Window at Execution Stop (Software Break)

The [System Status] window displays the following contents.



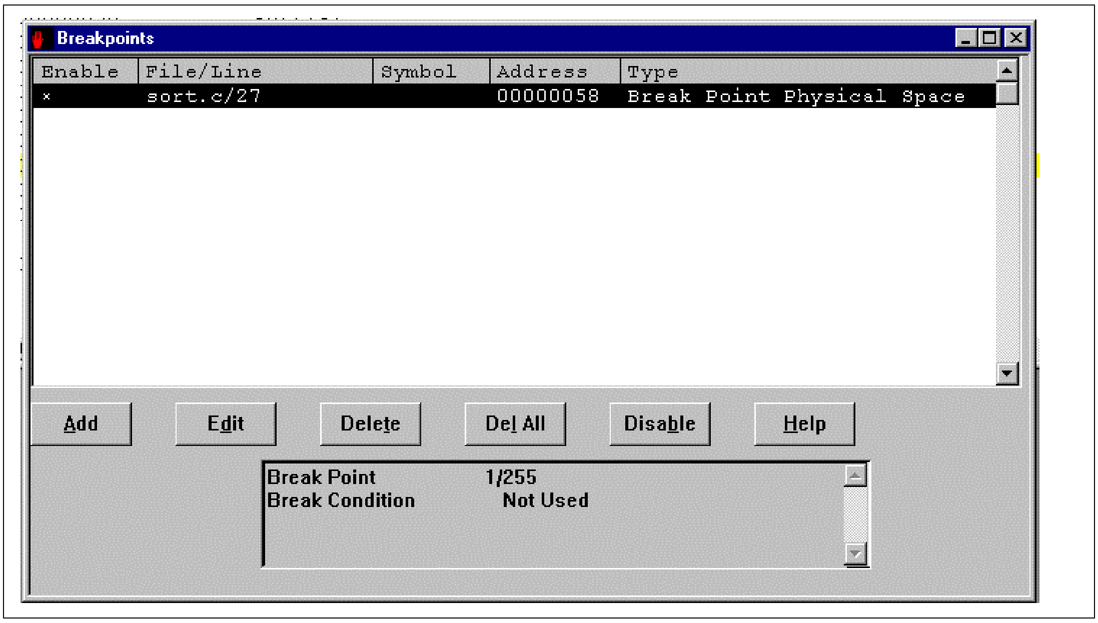
```
Emulator                 Connected
Session Name             C:\HDI_CE\tutorial\sort.hds
Program Name             C:\HDI_CE\tutorial\sort.abs

Connected to:            SH7729 E10A Emulator big endian (E10A PCI Card Driver2)
CPU                      SH7729
Run status               Break
Cause of last break      BREAK POINT
Run time count           0H:0M:0S:10MS
Emulator mode            Normal
Big endian
AUD                      Exist
```

Figure 3.39 Displayed Contents of the [System Status] Window (Software Break)

3.16 Hardware Break Function

An example is given below in which the address bus condition and the read cycles for the state condition are set under Break Condition 1 as hardware break conditions.

- Select [Breakpoint Window] from the [View] menu. The [Breakpoints] window is displayed.
- Click the [Del All] button to cancel all breakpoints that have been set.
- Click the [Add] button.



```
Enable  File/Line            Symbol    Address    Type
x       sort.c/27                      00000058   Break Point Physical Space
```

```
Break Point           1/255
Break Condition         Not Used
```

Figure 3.40 [Breakpoints] Window (Before Hardware Break Condition Setting)

The [Break] dialog box is displayed. To set hardware break conditions, select [Condition] in the [Break] dialog box to display the [Condition] page.

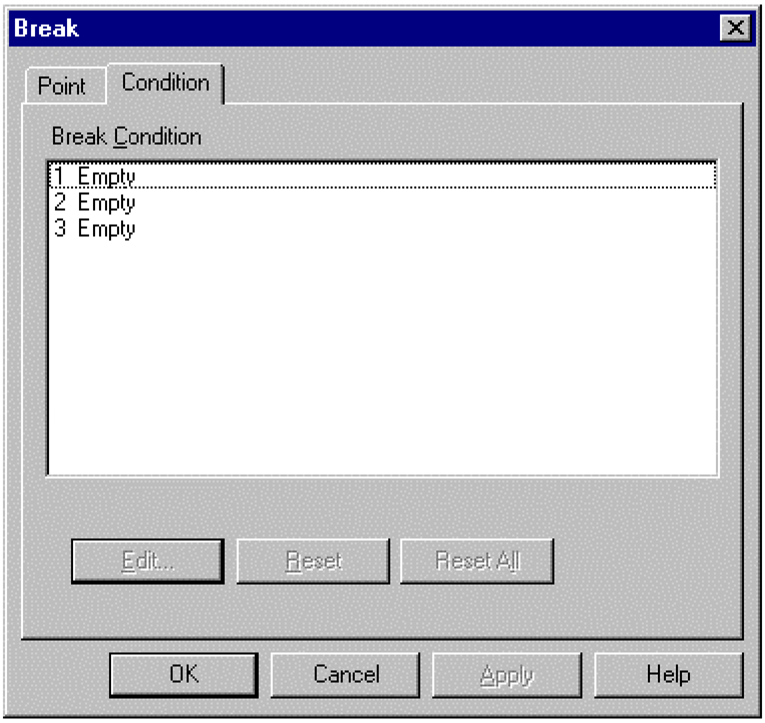
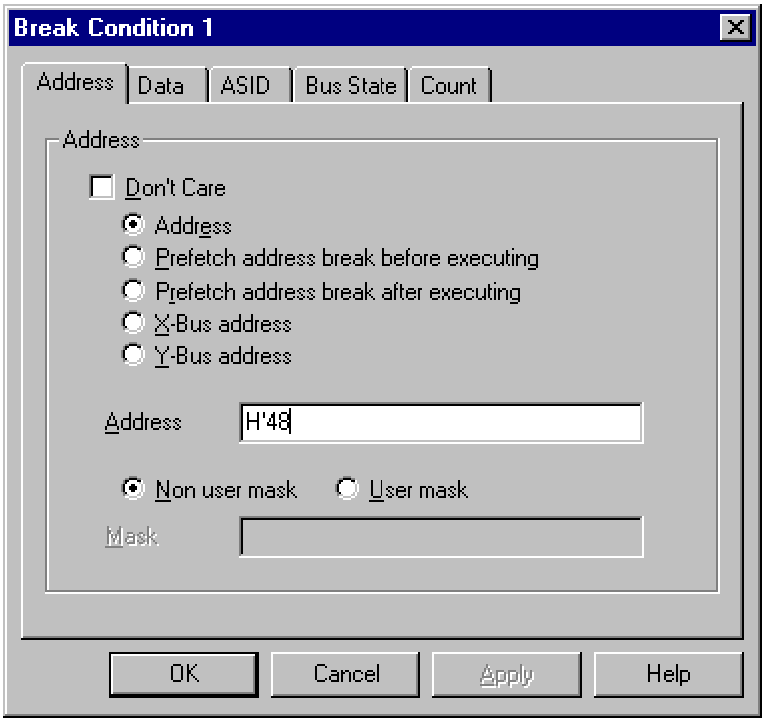


Figure 3.41   [Condition] Page ([Break] Dialog Box)

Up to three breakpoints can be set independently for the Break Condition hardware break condition. In this example, set the hardware break condition for Break Condition 1.

- Highlight the first point in the [Break Condition] list box.
- Click the [Edit...] button. The [Break Condition 1] dialog box is displayed.
- Clear the [Don't Care] check box in the [Address] page.
- Select the [Address] radio button and enter ***H'48*** as the value in the [Address] edit box.



Break Condition 1

Address | Data | ASID | Bus State | Count

Address

[ ] Don't Care

[✓ Address]
Prefetch address break before executing
Prefetch address break after executing
X-Bus address
Y-Bus address

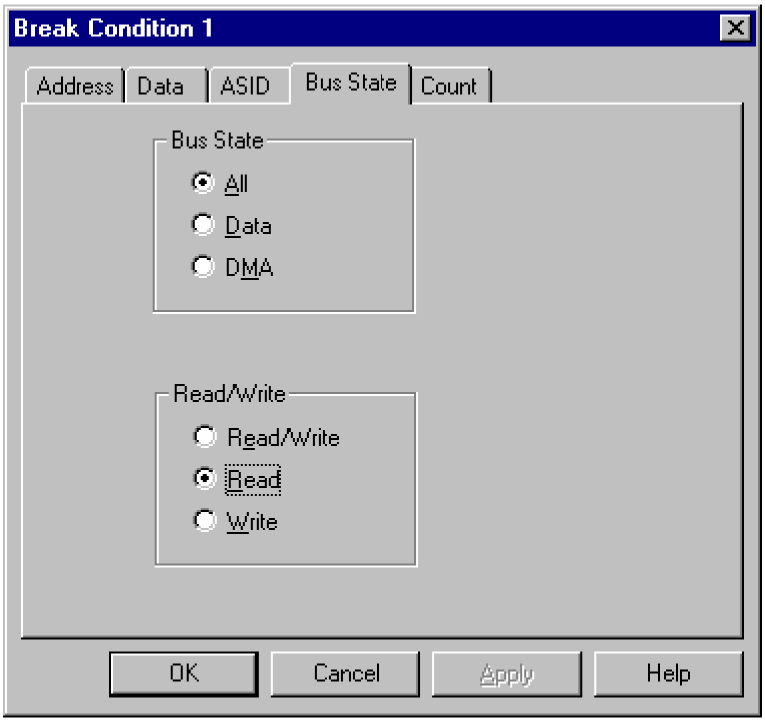
Address: H'48

[✓ Non user mask]   User mask

Mask:

Figure 3.42 [Address] Page ([Break Condition 1] Dialog Box)

- Select [Bus State] to display the [Bus State] page.
- Select the [Read] radio button in the [Read/Write] group box.

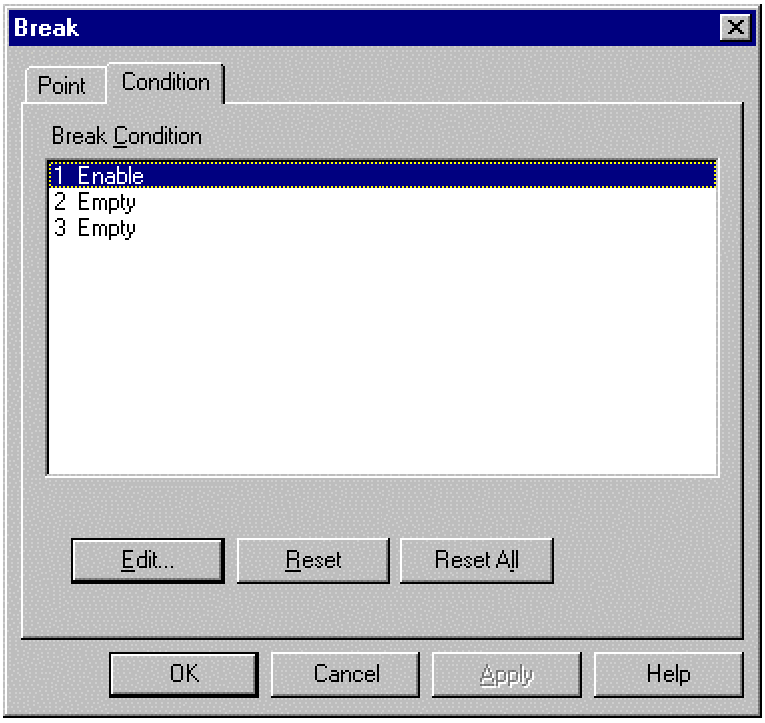


Bus State
- [✓ All]
- Data
- DMA

Read/Write
- Read/Write
- [✓ Read]
- Write

Figure 3.43 [Bus State] Page ([Break Condition 1] Dialog Box)

- Click the [OK] button.
- The [Break] dialog box is displayed, and the first point display in the [Break Condition] list box changes from `Empty` to `Enable`.



Break Condition

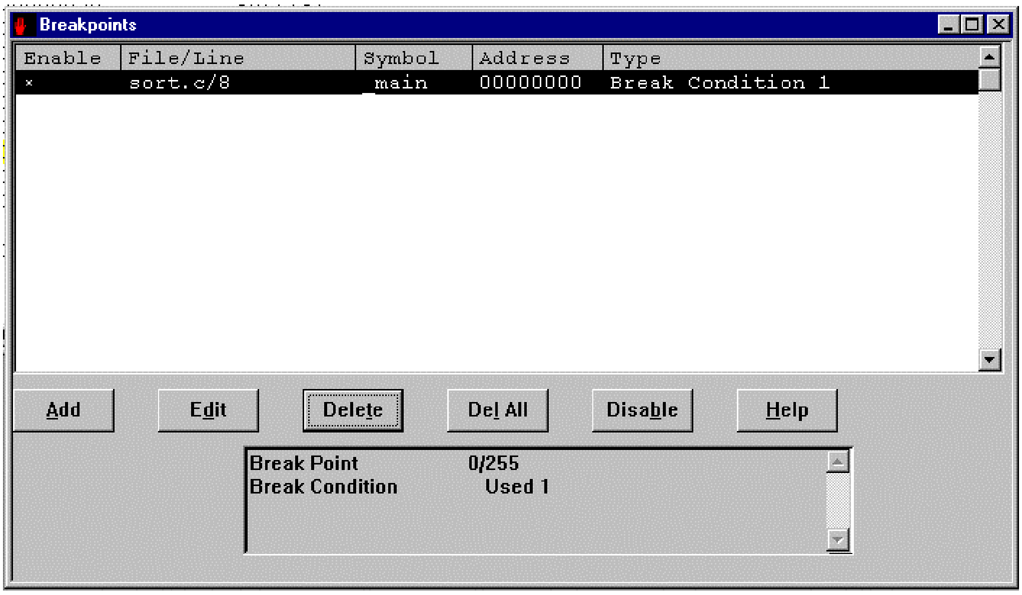
1 Enable
2 Empty
3 Empty

Figure 3.44 [Break] Dialog Box (After Hardware Break Condition Setting)

- Click the [OK] button.

The newly set hardware breakpoint is displayed in the [Breakpoints] window. With this setting, `Break Condition 1` is displayed in [Type] in the [Breakpoints] window.

This completes the setting of the Break Condition 1 hardware break condition. When the program is executed, a break will occur when address H'48 is accessed in a read cycle.



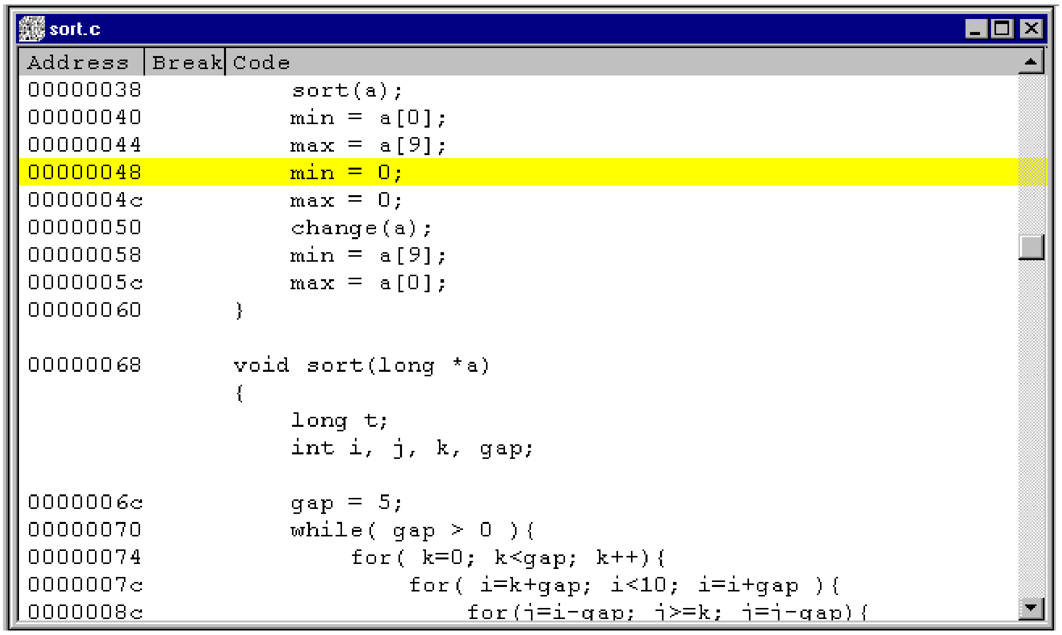
Enable	File/Line	Symbol	Address	Type
×	sort.c/8	main	00000000	Break Condition 1

```
Break Point          0/255
Break Condition        Used 1
```

Figure 3.45   [Breakpoints] Window ([Break Condition 1] Setting)

- Close the [Breakpoints] window.
- Set the program counter and stack pointer values that were set in section 3.8, Setting Registers, in the [Registers] window. Click the [Go] button.

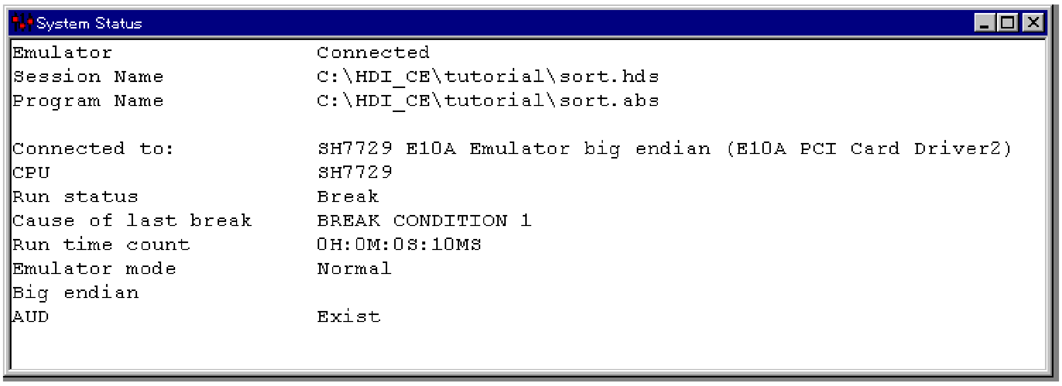
The program runs then stops at the condition specified under Break Condition 1.



```
sort.c
Address  Break Code
00000038          sort(a);
00000040          min = a[0];
00000044          max = a[9];
00000048          min = 0;
0000004c          max = 0;
00000050          change(a);
00000058          min = a[9];
0000005c          max = a[0];
00000060      }

00000068      void sort(long *a)
              {
                  long t;
                  int i, j, k, gap;

0000006c          gap = 5;
00000070          while( gap > 0 ){
00000074              for( k=0; k<gap; k++){
0000007c                  for( i=k+gap; i<10; i=i+gap ){
0000008c                      for(j=i-gap; j>=k; j=j-gap){
```

Figure 3.46 [Program] Window at Execution Stop (Break Condition 1)

The [System Status] window displays the following contents.



```
System Status
Emulator                 Connected
Session Name             C:\HDI_CE\tutorial\sort.hds
Program Name             C:\HDI_CE\tutorial\sort.abs

Connected to:            SH7729 E10A Emulator big endian (E10A PCI Card Driver2)
CPU                      SH7729
Run status               Break
Cause of last break      BREAK CONDITION 1
Run time count           0H:0M:0S:10MS
Emulator mode            Normal
Big endian
AUD                      Exist
```

Figure 3.47 Displayed Contents of the [System Status] Window (Break Condition 1)

3.16.1 Setting the Sequential Break Condition

The emulator has sequential break functions. When the hardware break conditions listed in table 3.5 are satisfied, program execution is halted. This mode is called sequential break.

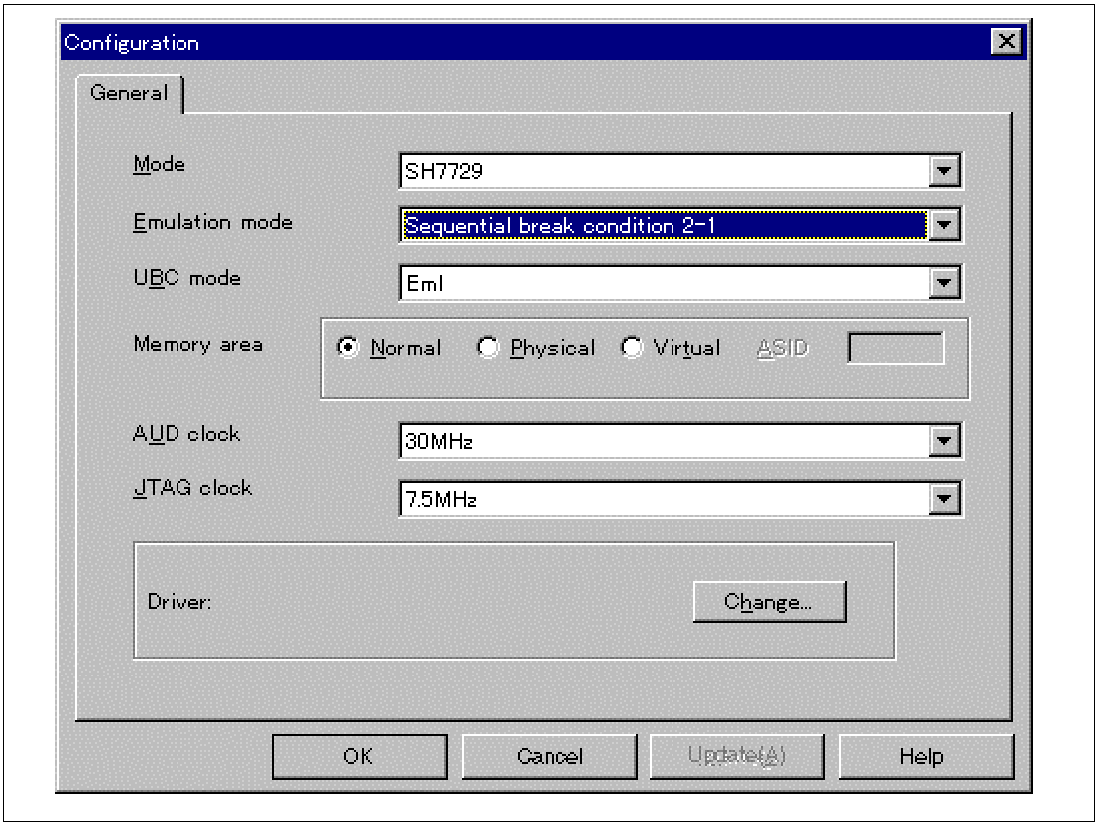
Table 3.5 Sequential Break Conditions

Break Condition	Description
Sequential break condition 2-1	Program is halted when Break Condition 2 and Break Condition 1 are satisfied in that order.

Sequential break condition 2-1 is described below as an example.

Before executing the program, change setting in the [Configuration] dialog box. Otherwise, the sequential break does not function.

- Select [Configure Platform...] from the [Setup] menu. The [Configuration] dialog box is displayed.
- Select Sequential break condition 2-1 from the [Emulation mode] combo box.



Mode: SH7729
Emulation mode: Sequential break condition 2-1
UBC mode: Eml
Memory area: [✓ Normal] Physical Virtual ASID
AUD clock: 30MHz
JTAG clock: 7.5MHz
Driver:

Figure 3.48 [Configuration] Dialog Box (Sequential Break Setting)

Click the [OK] button and close the [Configuration] dialog box.

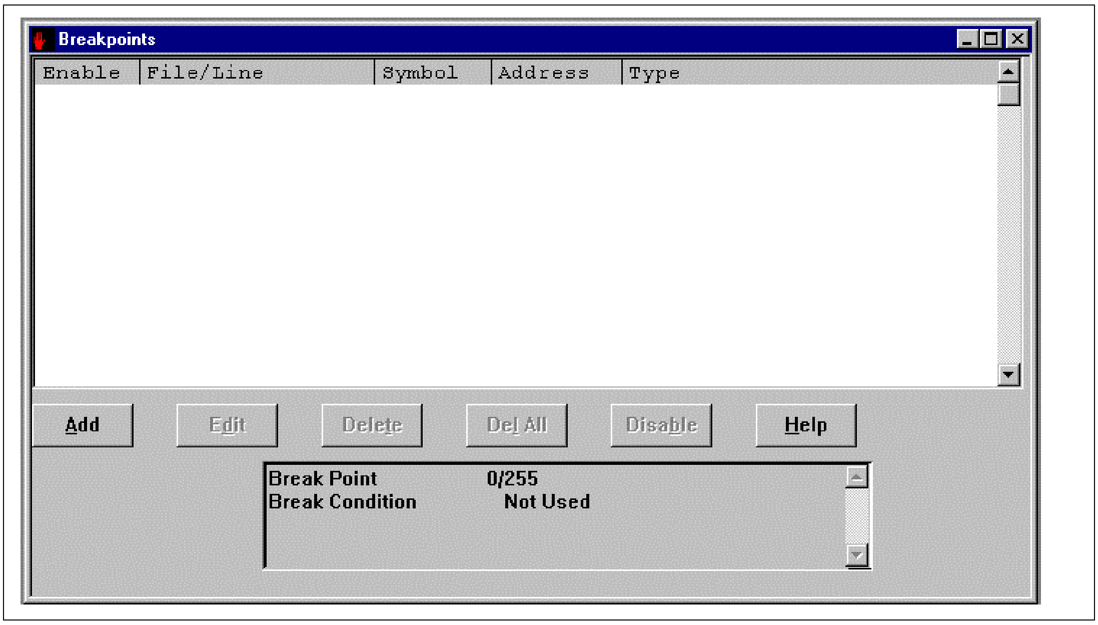
Set break conditions as follows:

Break condition 1: When address H'58 is accessed in a read cycle, a break condition is satisfied (Break Condition 1 is set).

Break condition 2: When address H'48 is accessed in a read cycle, a break condition is satisfied (Break Condition 2 is set).

Program execution is halted when break conditions 2 and 1 are satisfied in that order. Set the sequential break condition.

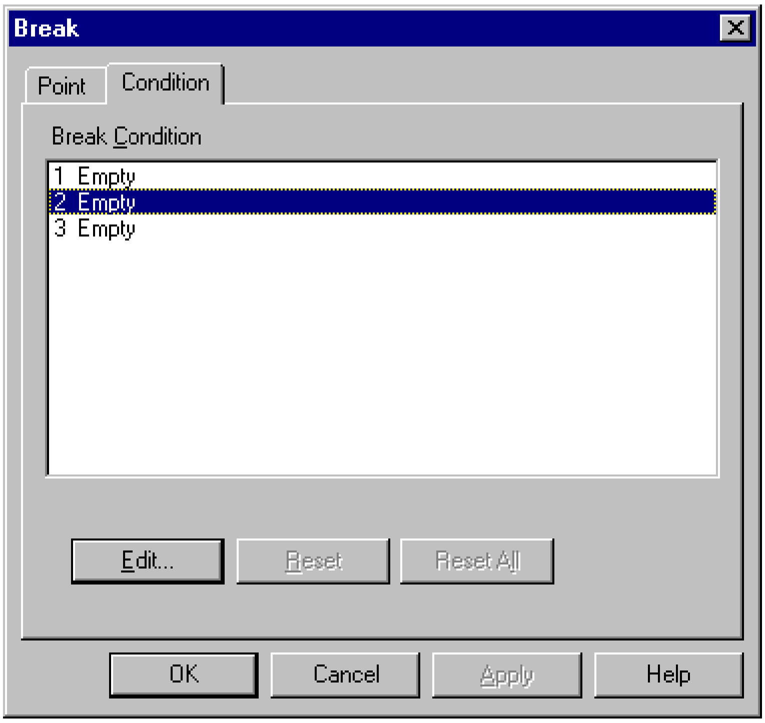
- Select [Breakpoint Window] from the [View] menu. The [Breakpoints] window is displayed.
- Click the [Del All] button to cancel all the set breakpoints.
- Click the [Add] button.



```
Break Point          0/255
Break Condition        Not Used
```

Figure 3.49 [Breakpoints] Window (Before Sequential Break Condition Setting)

The [Break] dialog box is displayed. To set sequential break conditions, select [Condition] in the [Break] dialog box to display the [Condition] page.

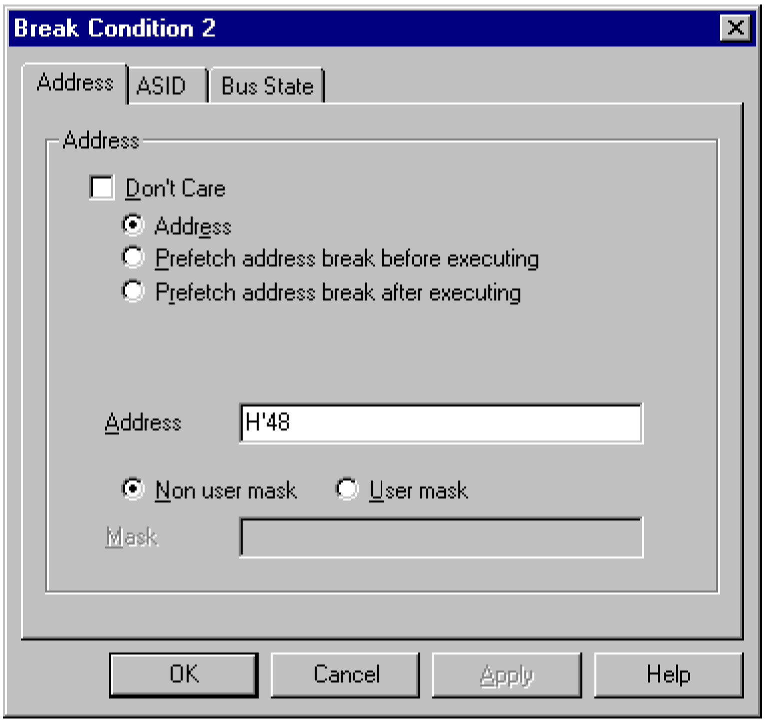


1 Empty
2 Empty
3 Empty

Figure 3.50   [Condition] Page ([Break] Dialog Box)

For the sequential break condition, set break condition 2 in the [Break Condition 2] dialog box, and set break condition 1 in the [Break Condition 1] dialog box.

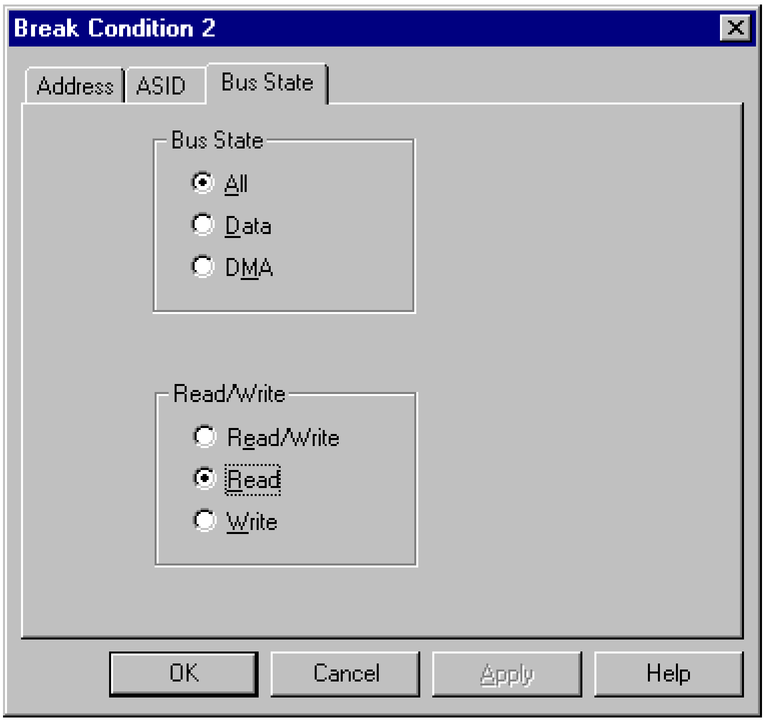
- Highlight the second point in the [Break Condition] list box.
- Click the [Edit...] button. The [Break Condition 2] dialog box is displayed.
- Clear the [Don't Care] check box in the [Address] page.
- Select the [Address] radio button and input address ***H'48*** as the value in the [Address] edit box.



Address: H'48

Figure 3.51 [Break Condition 2] Dialog Box (Condition 2 [Address] Page)

- Select [Bus State] to display the [Bus State] page.
- Select the [Read] radio button in the [Read/Write] group box.



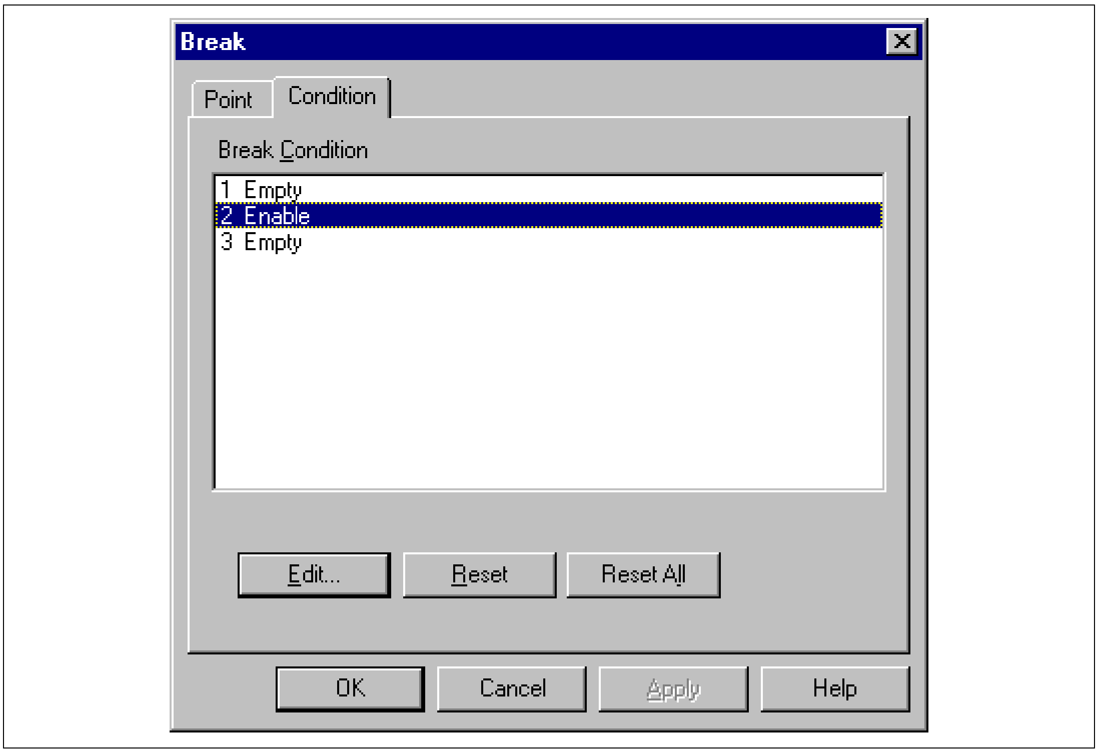
Bus State: [✓ All] Data DMA

Read/Write: Read/Write [✓ Read] Write

Figure 3.52 [Break Condition 2] Dialog Box (Condition 2 [Bus State] Page)

- Click the [OK] button.

- The [Break] dialog box is displayed, and the second point display in the [Break Condition] list box changes from `Empty` to `Enable`.



```
1  Empty
2  Enable
3  Empty
```

Figure 3.53 [Break] Dialog Box (After [Break Condition 2] Condition Setting)

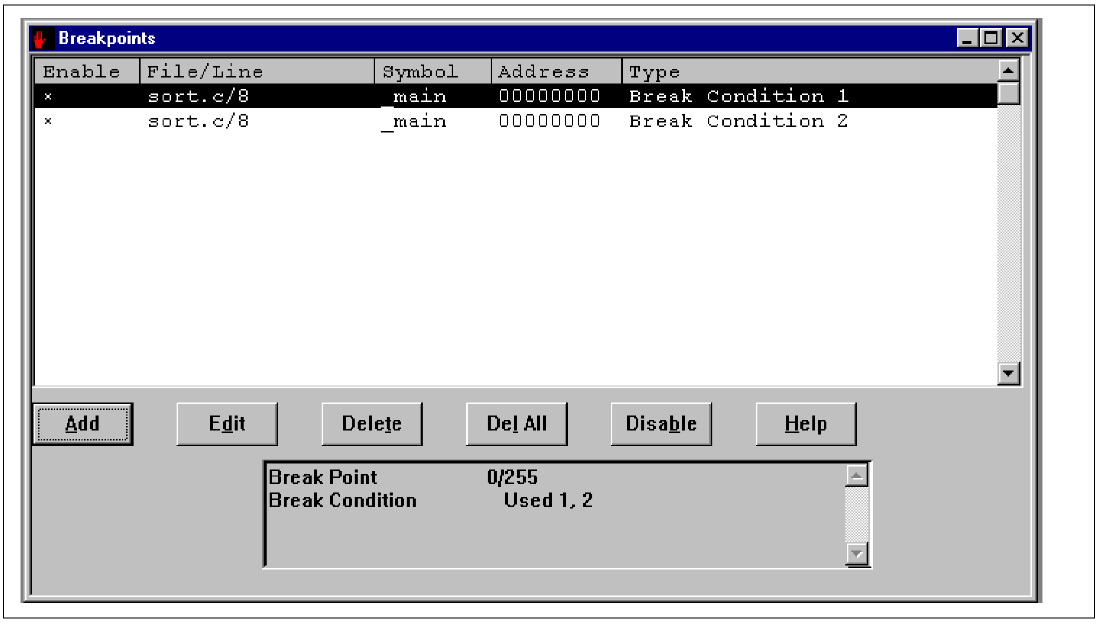
This completes the setting of break condition 2. Next, set break condition 1 as follows:

- Highlight the first point in the [Break Condition] list box.
- Click the [Edit...] button. The [Break Condition 1] dialog box is displayed.

The setting can then be made in the same way as for break condition 1.

- After setting break conditions 1 and 2, click the [OK] button.

`Break Condition 1` and `Break Condition 2` are displayed in [Type] in the [Breakpoints] window.



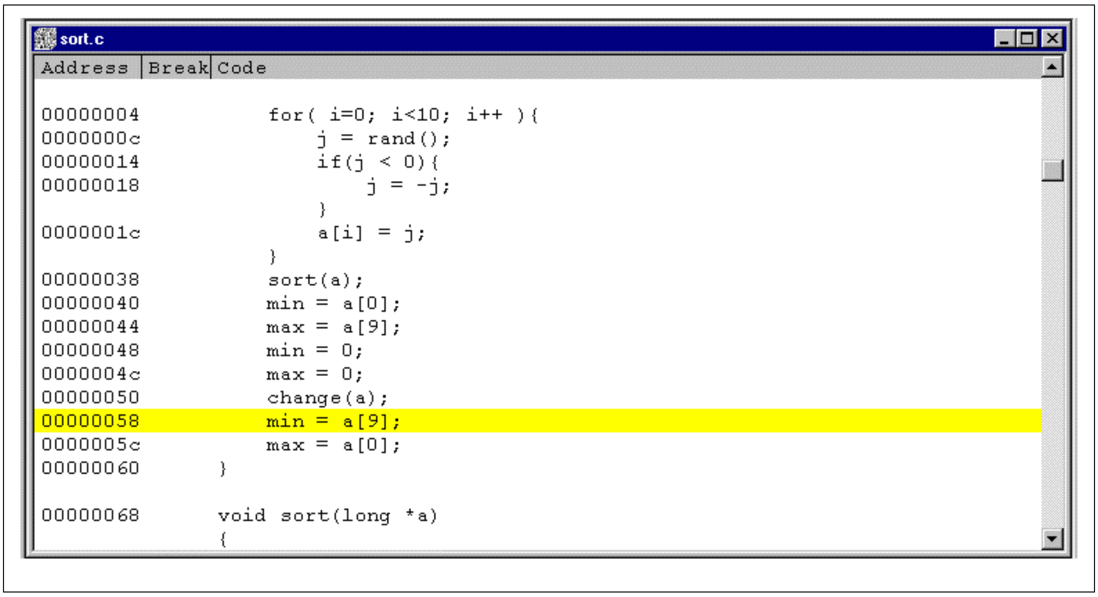
```
Enable    File/Line            Symbol     Address      Type
×         sort.c/8             _main      00000000     Break Condition 1
×         sort.c/8             _main      00000000     Break Condition 2
```

```
Break Point          0/255
Break Condition        Used 1, 2
```

Figure 3.54 [Breakpoints] Window (After Sequential Break Condition Setting)

- Close the [Breakpoints] window.
- Set the program counter and stack pointer values that were set in section 3.8, Setting Registers, in the [Registers] window. Click the [Go] button.

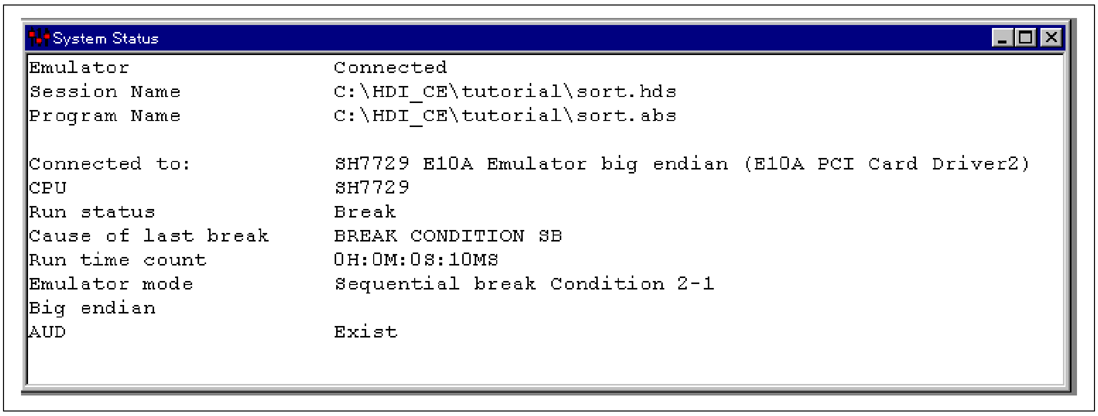
The program runs then stops at the condition specified under Break Condition 1.



```
sort.c
Address   Break Code

00000004           for( i=0; i<10; i++ ){
0000000c               j = rand();
00000014               if(j < 0){
00000018                   j = -j;
                       }
0000001c               a[i] = j;
                   }
00000038           sort(a);
00000040           min = a[0];
00000044           max = a[9];
00000048           min = 0;
0000004c           max = 0;
00000050           change(a);
00000058           min = a[9];
0000005c           max = a[0];
00000060       }

00000068       void sort(long *a)
               {
```

Figure 3.55 [Program] Window at Execution Stop (Sequential Break)

The [System Status] window displays the following contents.



```
System Status
Emulator                Connected
Session Name            C:\HDI_CE\tutorial\sort.hds
Program Name            C:\HDI_CE\tutorial\sort.abs

Connected to:           SH7729 E10A Emulator big endian (E10A PCI Card Driver2)
CPU                     SH7729
Run status              Break
Cause of last break     BREAK CONDITION SB
Run time count          0H:0M:0S:10MS
Emulator mode           Sequential break Condition 2-1
Big endian
AUD                     Exist
```

Figure 3.56 Displayed Contents of the [System Status] Window (Sequential Break)

3.17 Trace Functions

The HDI traces and displays the branch instructions. In the HDI, the acquired trace information can be displayed in the [Trace] window.

Table 3.6 shows the trace functions.

Table 3.6 Trace Functions

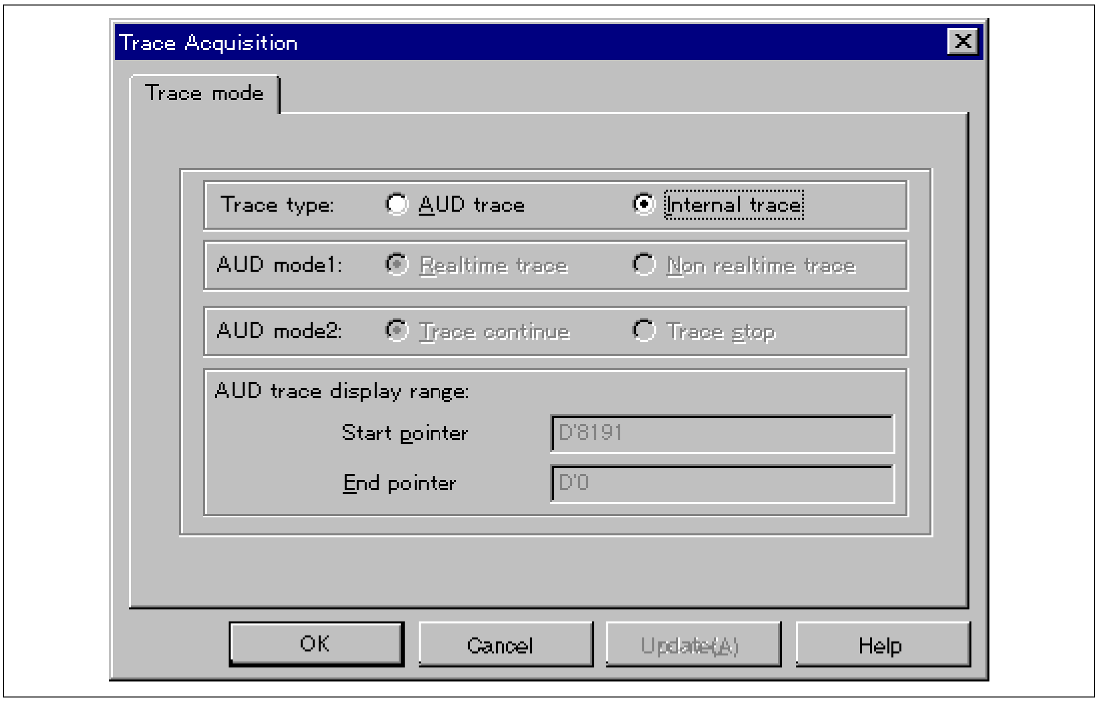
Function		Description
Internal trace		Branch instruction trace functions which are built into the MCU. This function displays the branch source and branch destination addresses and enables a realtime trace.
AUD trace	Realtime trace	This function is operational when the AUD pin is connected to the emulator. This function displays the branch source and branch destination addresses, and instruction words at the branch destination. When the next branch occurs while the trace information is being acquired, the information is overwritten and the latest trace information is acquired. The user program can be executed in realtime, but some trace information will not be acquired. • Trace continue function: When the trace buffer becomes full, this function always overwrites the oldest trace information to acquire the latest trace information. • Trace stop function: After the trace buffer becomes full, the trace information is not acquired. (The user program is continuously executed.)
	Non realtime trace	This function is operational when the AUD pin is connected to the emulator. This function displays the branch source and branch destination addresses, and instruction words at the branch destination. When the next branch occurs while the trace information is being acquired, the CPU stops operations until the information is acquired. The user program is not executed in realtime. • Trace continue function: When the trace buffer becomes full, this function always overwrites the oldest trace information to acquire the latest trace information. • Trace stop function: After the trace buffer becomes full, the trace information is not acquired. (The user program is continuously executed.)

3.17.1 Internal Trace Function

The branch source addresses and branch destination addresses for the eight latest branch instructions are displayed.

The following is a procedure to set the internal trace function:

1. Select [Trace window] from the [View] menu.
2. Click the [Acquisition] button to display the [Trace Acquisition] window.
3. Select the [Internal trace] radio button in the [Trace type] group box.



Trace type: ( ) AUD trace [✓ Internal trace]

AUD mode1: [✓ Realtime trace] ( ) Non realtime trace

AUD mode2: [✓ Trace continue] ( ) Trace stop

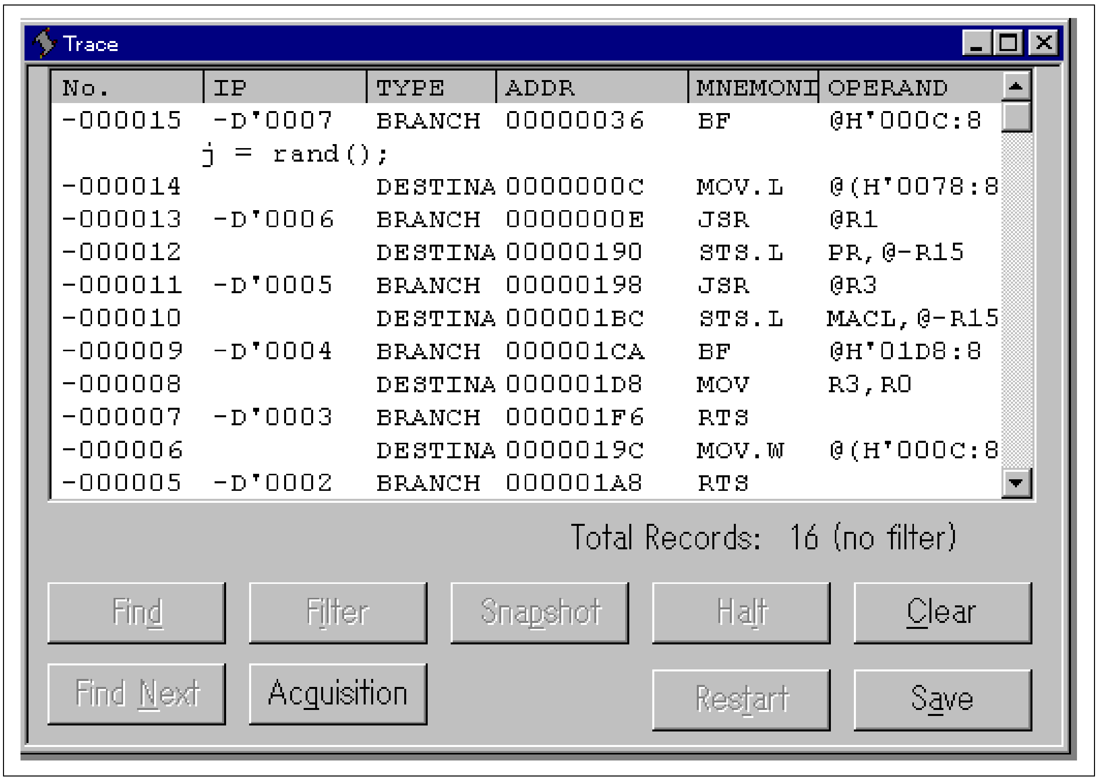
AUD trace display range:

Start pointer D'8191

End pointer D'0

Figure 3.57 [Trace mode] Window

Run the program as shown in the example of section 3.15.1, Software Break Function. The trace results are displayed in the [Trace] window after the program execution is completed.



```
No.      IP       TYPE     ADDR      MNEMONI OPERAND
-000015  -D'0007  BRANCH   00000036  BF      @H'000C:8
        j = rand();
-000014           DESTINA  0000000C  MOV.L   @(H'0078:8
-000013  -D'0006  BRANCH   0000000E  JSR     @R1
-000012           DESTINA  00000190  STS.L   PR,@-R15
-000011  -D'0005  BRANCH   00000198  JSR     @R3
-000010           DESTINA  000001BC  STS.L   MACL,@-R15
-000009  -D'0004  BRANCH   000001CA  BF      @H'01D8:8
-000008           DESTINA  000001D8  MOV     R3,R0
-000007  -D'0003  BRANCH   000001F6  RTS
-000006           DESTINA  0000019C  MOV.W   @(H'000C:8
-000005  -D'0002  BRANCH   000001A8  RTS
```

Total Records: 16 (no filter)

Figure 3.58 [Trace] Window

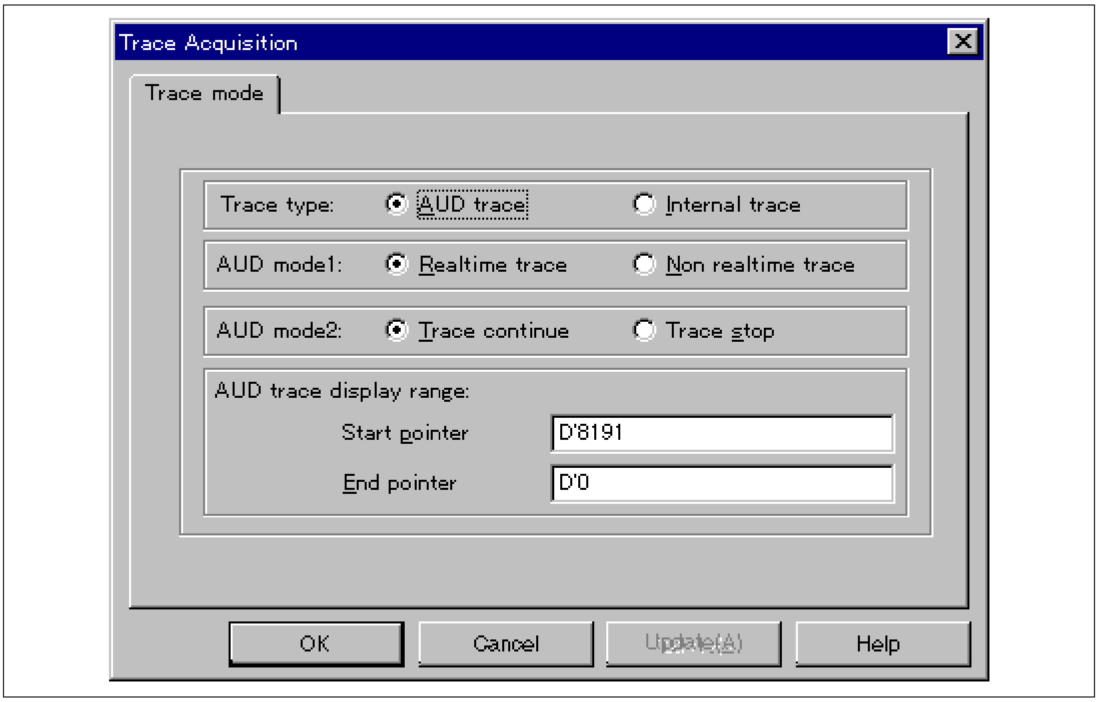
- If necessary, adjust the column width by dragging the header bar immediately below the title bar.

3.17.2 AUD Trace Function

This function is operational when the dedicated AUD pin is connected to the emulator. For available products, refer to table 6.7 in section 6, SHxxxx E10A Emulator Specifications.

The following is the procedure for setting the AUD trace function:

1. Select [Trace window] from the [View] menu.
2. Click the [Acquisition] button to display the [Trace Acquisition] window.
3. Select the [AUD trace] radio button in the [Trace type] group box.


Trace Acquisition

Trace mode

Trace type: [✓ AUD trace] Internal trace

AUD mode1: [✓ Realtime trace] Non realtime trace

AUD mode2: [✓ Trace continue] Trace stop

AUD trace display range:

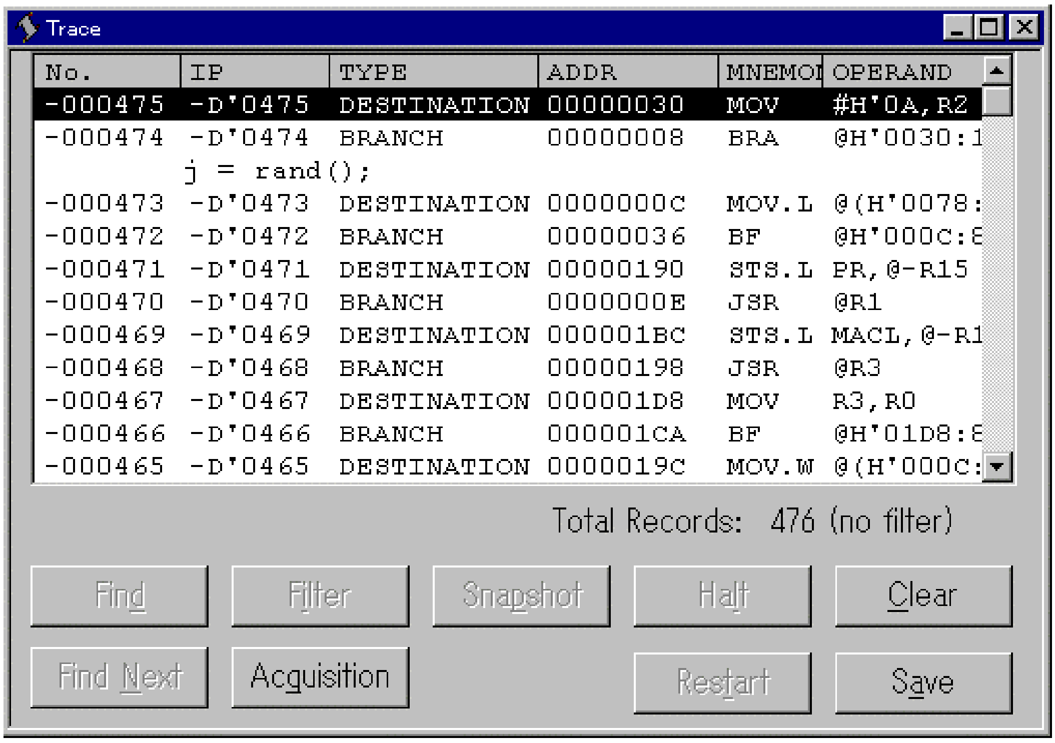
Start pointer D'8191

End pointer D'0

Figure 3.59 [Trace mode] Window

Note: For a description of each option, refer to table 3.6.

Run the program as shown in the example of section 3.15.1, Software Break Function. The trace results are displayed in the [Trace] window after the program execution is completed.



```
No.      IP       TYPE         ADDR      MNEMOI OPERAND
-000475  -D'0475  DESTINATION  00000030  MOV    #H'0A,R2
-000474  -D'0474  BRANCH       00000008  BRA    @H'0030:1
         j = rand();
-000473  -D'0473  DESTINATION  0000000C  MOV.L  @(H'0078:
-000472  -D'0472  BRANCH       00000036  BF     @H'000C:8
-000471  -D'0471  DESTINATION  00000190  STS.L  PR,@-R15
-000470  -D'0470  BRANCH       0000000E  JSR    @R1
-000469  -D'0469  DESTINATION  000001BC  STS.L  MACL,@-R1
-000468  -D'0468  BRANCH       00000198  JSR    @R3
-000467  -D'0467  DESTINATION  000001D8  MOV    R3,R0
-000466  -D'0466  BRANCH       000001CA  BF     @H'01D8:8
-000465  -D'0465  DESTINATION  0000019C  MOV.W  @(H'000C:
```

Total Records: 476 (no filter)

Figure 3.60 [Trace] Window

- If necessary, adjust the column width by dragging the header bar immediately below the title bar.

3.17.3 VP_MAP Translation

The MCU, which has an MMU, translates internal addresses (virtual addresses) to actual memory addresses (physical addresses). Address translation is performed according to the address translation table (translation look-aside buffer: TLB) in the MCU. The MMU operates during command input wait state as well as during user program execution. When a command for memory access is executed while the MMU address translation function is enabled, the address translated by the MMU is accessed. If the specified address is not within the TLB, a TLB miss occurs, and the TLB must be updated by the user program.

The emulator has address translation functions according to the VP_MAP tables. The VP_MAP tables are the address translation tables for the emulator created with the VPMAP_SET command.

The following shows an example of how to use the VP_MAP tables.

Example:

1. Create VP_MAP tables for translating virtual addresses H'10000 to H'10FFF to physical addresses H'4000000 to H'4000FFF and virtual addresses H'11000 to H'11FFF to physical addresses H'0 to H'FFF.

   ```
   >vs 10000 10FFF 4000000 (RET)
   >vs 11000 11FFF 0 (RET)
   >vd (RET)
   <VADDR_TOP>  <VADDR_END>  <PADDR_TOP>
   00010000     00010FFF     04000000
   00011000     00011FFF     00000000
   DISABLE
   ```

2. Then, enable the VP_MAP tables. (When the tables are disabled, addresses are not translated.)

   ```
   >ve ;enable (RET)
   >vd (RET)
   <VADDR_TOP>  <VADDR_END>  <PADDR_TOP>
   00010000     00010FFF     04000000
   00011000     00011FFF     00000000
   ENABLE
   ```

   Here, virtual addresses correspond to physical addresses as shown in figure 3.61.

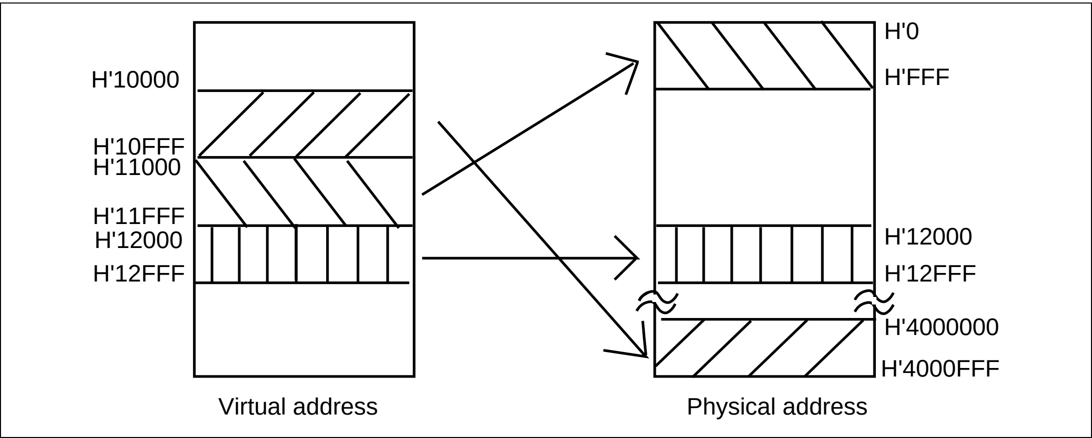


Figure 3.61 Address Translation according to VP_MAP Tables

How to translate addresses depends on the settings of the radio buttons of the memory area group in the [Configuration] dialog box. The following shows how to translate addresses in each setting state.

- When the Normal radio button is selected:

  The VP_MAP table has a priority over the TLB. When the VP_MAP table is enabled and the specified address is within the VP_MAP table settings, the emulator translates the address according to the VP_MAP table. If the specified address is outside the VP_MAP table settings even when the VP_MAP table is enabled, or when the VP_MAP table is disabled, the emulator translates the address according to the MMU state.
- When the Virtual radio button is selected:

  The address is translated according to the TLB. If the specified address is outside the TLB table settings, a TLB error will occur.
- When the Physical radio button is selected:

  The address is not translated.

Table 3.7 Address Translation Tables

	VP_MAP		MMU		
Radio Button*	**Enabled/ Disabled**	**Within/ Outside the range**	**Enabled/ Disabled**	**Within/Outside the TLB Range**	**Table Used for Translation**
Normal	Enabled	Within the Range	Enabled	Within the Range	Translated according to the VP_MAP table
				Outside the range	Translated according to the VP_MAP table
			Disabled	Within/outside the range	Translated according to the VP_MAP table
		Outside the Range	Enabled	Within the Range	Translated according to the TLB table
				Outside the range	TLB error
			Disabled	Within/outside the range	Not translated
	Disabled	Within/ outside the range	Enabled	Within the Range	Translated according to the TLB table
				Outside the range	TLB error
			Disabled	Within/outside the range	Not translated
Virtual	Enabled/ disabled	Within/ outside the range	Enabled	Within the Range	Translated according to the TLB table
				Outside the range	TLB error
			Disabled	Within the Range	Translated according to the TLB table
				Outside the range	TLB error
Physical	Enabled/ disabled	Within/ outside the range	Enabled/ disabled	Within/outside the range	Not translated

Note: Specified by the [Configuration] dialog box.

3.18 What Next?

This tutorial has described the major features of the emulator and the use of the HDI.

Sophisticated debugging can be carried out by using the emulation functions that the emulator offers. This provides for effective investigation of hardware and software problems by accurately isolating and identifying the conditions under which such problems arise.

Further details on the use of the HDI can be found in the separately issued Hitachi Debugging Interface User's Manual.

Section 4 Descriptions of Windows

4.1 HDI Windows

HDI window menu bars and the corresponding pull-down menus are listed in table 4.1. Where a description of a menu is included in the Hitachi Debugging Interface User's Manual or in this manual, a O mark or the relevant section number is shown. Related commands in the E10A Emulator User's Manual are also shown.

Table 4.1 HDI Window Menus and Related Manual Entries

Menu Bar	Pull-Down Menu	Hitachi Debugging Interface User's Manual	This Manual
File menu	Load Program…	O	3.6.1
	Save Memory…	O	—
	Verify Memory…	O	—
	Save Session	O	2.6
	Load Session…	O	—
	Save Session As…	O	—
	Initialize	O	—
	Exit	O	—
Edit Menu	Cut	O	—
	Copy	O	—
	Paste	O	—
	Find…	O	—
	Set Line…	O	—
	Fill Memory…	O	—
	Move Memory…	O	—
	Test Memory…	O	—
	Update Memory	O	—

Table 4.1 HDI Window Menus and Related Manual Entries (cont)

Menu Bar	Pull-Down Menu	Hitachi Debugging Interface User's Manual	This Manual
View Menu	Toolbar	O	—
	Status Bar	O	—
	Breakpoint Window	O	3.10, 3.15.1, 4.2.3, 6.4.5
	Command Line Window	O	—
	I/O Register Window	O	—
	Local Variable Window	O	3.14
	Memory Mapping Window	O	—
	Memory Window…	O	3.11
	Performance Analysis Window	O	—
	Program Window…	O	3.6.2
	Register Window	O	3.8
	Status Window	O	3.9, 3.15.1, 4.2.10
	Text Window	O	—
	Trace Window	O	4.2.8, 6.4.3, 6.4.7
	Watch Window	O	3.12
Run Menu	Go	O	3.9
	Go Reset	O	—
	Go to Cursor	O	—
	Run…	O	—
	Step In	O	3.13.1
	Step Over	O	3.13.3
	Step Out	O	3.13.2
	Step…	O	—
	Halt Program	O	—
	Set PC To Cursor	O	—
	Reset CPU	O	—

Table 4.1 HDI Window Menus and Related Manual Entries (cont)

Menu Bar	Pull-Down Menu	Hitachi Debugging Interface User's Manual	This Manual
Setup Menu	Options	O	—
	Radix	O	—
	Customise	O	—
	Select Platform…	O	2.5
	Configure Platform…	O	3.5, 4.2
Tools Menu	Symbols…	O	—
	Evaluate…	O	—
Window Menu	Cascade	O	—
	Tile	O	—
	Arrange Icons	O	—
	Close All	O	—
Help Menu	Index	O	—
	Using Help	O	—
	Search for Help on	O	—
	About HDI	O	—

4.2 Descriptions of Each Window

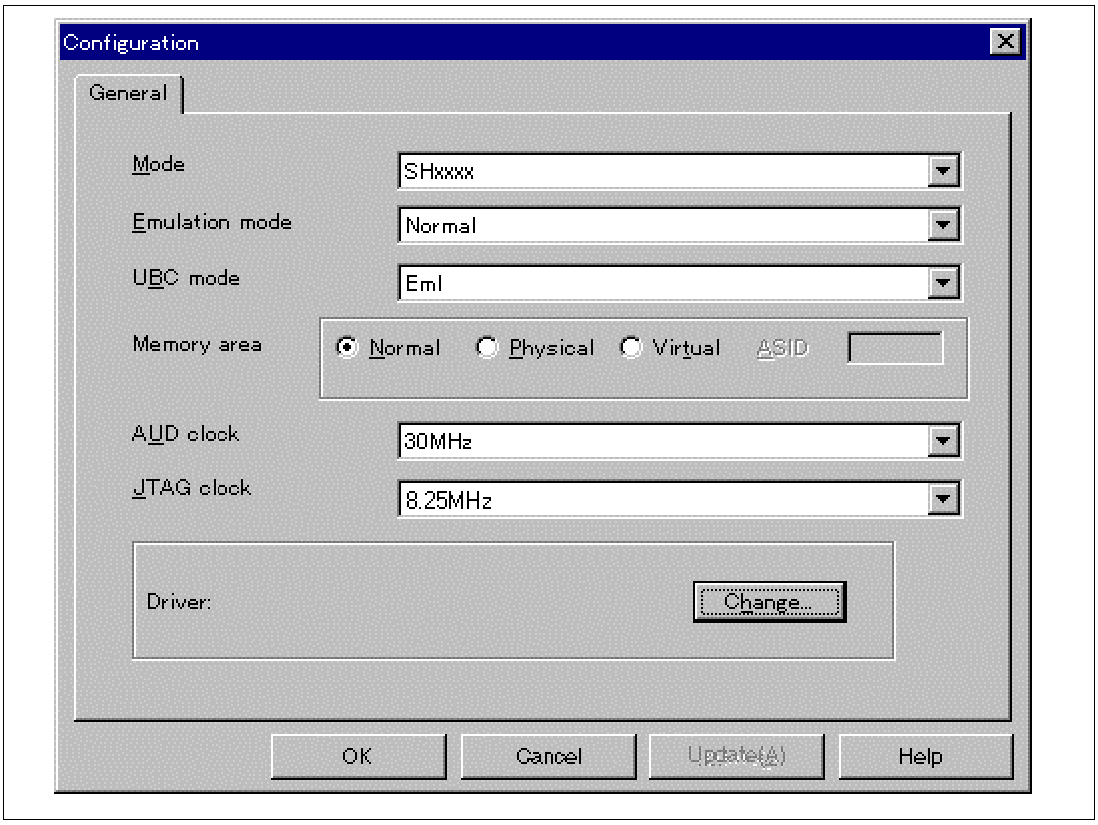
This section describes each window. Figures in this section are used as examples. Each E10A emulator type has explanatory notes. Read section 6, SHxxxx E10A Emulator Specifications.

4.2.1 [Configuration] Dialog Box

Function:

This dialog box sets the emulation conditions of the emulator.

Window:


Configuration

General

Mode: SHxxxx

Emulation mode: Normal

UBC mode: Eml

Memory area: [✓ Normal] Physical Virtual ASID

AUD clock: 30MHz

JTAG clock: 8.25MHz

Driver:

Figure 4.1 [Configuration] Dialog Box

Description:

The [Configuration] dialog box consists of the [General] page listed in table 4.2.

Table 4.2 [Configuration] Dialog Box Page

Page Name	Description
[General]	Sets and displays the emulation mode conditions.

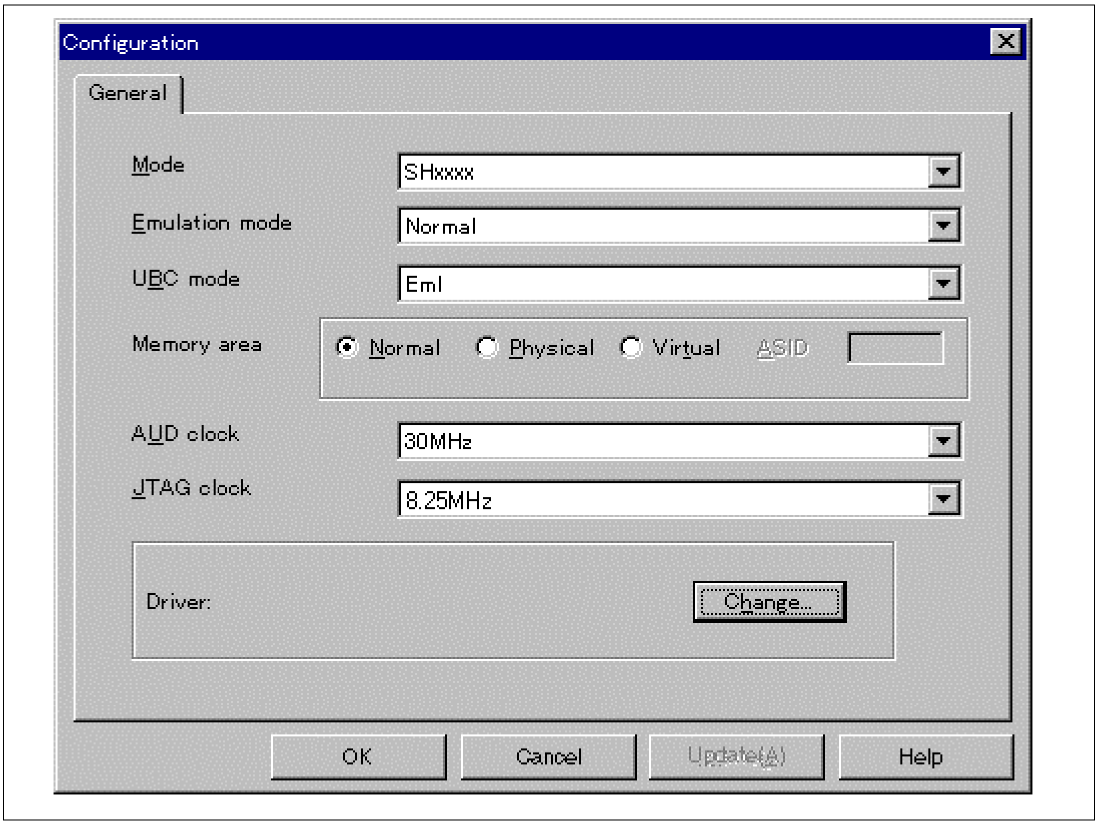
Clicking the [OK] button sets the emulation conditions. If the [Cancel] button is clicked, this dialog box is closed without setting the emulation conditions.

(1) [General] Page ([Configuration] Dialog Box)

Function:

This page sets the emulator operation conditions.

Window:



Mode: SHxxxx

Emulation mode: Normal

UBC mode: Eml

Memory area: [✓ Normal] Physical Virtual ASID

AUD clock: 30MHz

JTAG clock: 8.25MHz

Driver:

Figure 4.2 [General] Page ([Configuration] Dialog Box)

Description:

Table 4.3 [General] Page Options

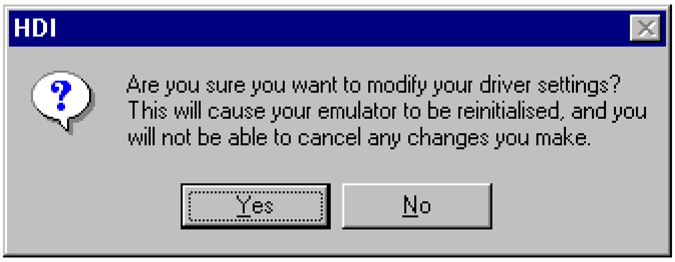
Option	Description
[Mode] combo box	Displays the MCU name.
[Emulation mode] combo box	Selects the execution mode. Select Normal to perform normal emulation. Select No Break to disable breakpoint settings. Select Sequential break Condition 2-1 to use the sequential break function*1. (For Sequential break Condition 2-1, execution stops when conditions are satisfied in the order of Break Condition 2 and Break Condition 1.)
[UBC mode] combo box	EML: The UBC is used as a Break Condition by the emulator. USER: The UBC is released for users. In this case, the [Break Condition] page becomes non-active.
[Memory area] group box	Sets the address setting mode to the memory area. The default is Normal. When the VP_MAP is enabled and the address is within the table range, address translation is done according to the VP_MAP table. For other cases, address translation is done according to the MMU state. Select Physical when setting with a physical address. Select Virtual when address translation is done by the TLB table.
[AUD clock] combo box	Selects the AUD clock*2. By default, the maximum frequency of each card emulator is displayed. The following frequencies can be selected: PCMCIA card: 7.5 MHz, 15 MHz, 30 MHz, and 60 MHz. PCI card: 10 MHz, 20 MHz, 30 MHz, 40 MHz, 50 MHz, 60 MHz, 70 MHz, 80 MHz, 90 MHz, 100 MHz, and CK/2.
[JTAG clock] combo box	Sets the Hitachi-UDI frequency*3.
[Driver] group box	Displays the driver currently selected.
[Change..] button	Displays the [E10A Driver Details] dialog box. Use when a driver currently connected is changed.

Notes: 1. When using the sequential break function, set the corresponding hardware break conditions.

2. The range of frequencies that the AUD operates under is different according to the MCUs used. For details, refer to section 6.4.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK).

3. The range of frequencies that the Hitachi-UDI operates at is different according to the MCUs used. For details, refer to section 6.4.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK).

When a driver is to be changed with the [Change..] button, the following message is displayed.


HDI

Are you sure you want to modify your driver settings? This will cause your emulator to be reinitialised, and you will not be able to cancel any changes you make.

Figure 4.3 Warning Message Box

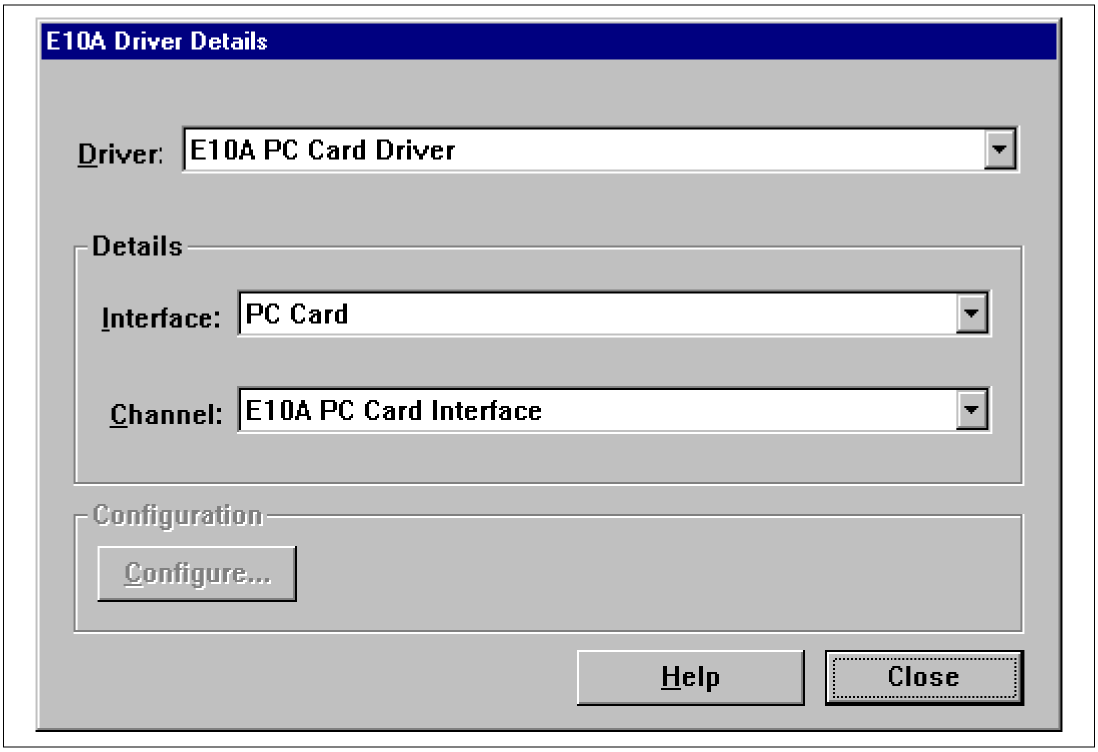
When the [Yes] button is clicked, the [E10A Driver Details] dialog box is displayed. When the [No] button is clicked, the display returns to the [Configuration] dialog box.

Related Command:

GO_OPTION command

4.2.2 [E10A Driver Details] Dialog Box

The [E10A Driver Details] dialog box is displayed when the [OK] button is clicked. Select the correct driver.



Driver: E10A PC Card Driver

Interface: PC Card

Channel: E10A PC Card Interface

Figure 4.4 [E10A Driver Details] Dialog Box

- With the [Driver] combo box, select the driver to connect the HDI with the emulator.
- [Interface] displays the interface name of the card to be connected, and [Channel] displays the interface to which the board is connected. Once the driver is selected in the [E10A Driver Details] dialog box, this dialog box is not displayed when the HDI is run next time.

  [Driver] combo box: Select [E10A PC Card Driver] to use the PCMCIA card.
  Select [E10A PCI Card Driver] to use the PCI card.
  [Interface] combo box: Select [PC Card] to use the PCMCIA card.
  Select [PCI] to use the PCI card. (If the driver is not installed, the
  [PC
  Card] or [PCI] is not displayed.)
- Click the [Close] button.

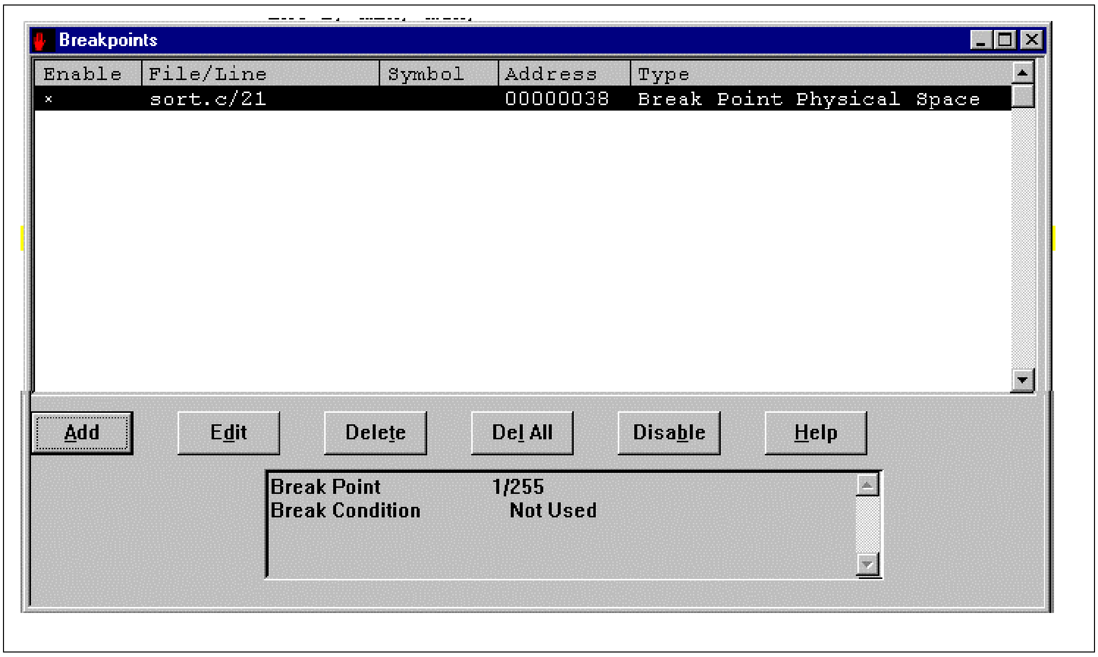
Note: When the HDI is not linked up even if the above procedure has been executed, the driver will not be set correctly. Use drivers provided with installation disk #3 (the \SETUP directory when the emulator setup program is provided with the CD-R) if necessary.

4.2.3 [Breakpoints] Window

Function:

This window lists all break conditions that have been set.

Window:



```
Enable  File/Line     Symbol  Address   Type
×       sort.c/21             00000038  Break Point Physical Space
```

```
Break Point         1/255
Break Condition       Not Used
```

Figure 4.5 [Breakpoints] Window

Description:

The [Breakpoints] window displays breakpoint setting information. The items listed in the following tables are displayed.

Table 4.4 [Breakpoints] Window Display Items

Item	Description
[Enable]	Displays whether the break condition is enabled or disabled. Symbol x indicates that the break condition is enabled.
[File/Line]	Displays the file name and line number where the breakpoint is set.
[Symbol]	Displays the symbol corresponding to the breakpoint address. If no symbol has been defined for the address, a blank is displayed.
[Address]	Displays the address where the breakpoint is set.
[Type]	Displays the break condition type as follows: Break Point Virtual Space ASID = D'xxx: Software breakpoint (Virtual address ASID value in decimal.) Break Point Physical Space: Software breakpoint (Physical address) Break Condition 1 to Break Condition 3: Hardware break condition

Buttons in the window can be used to set, change, and clear breakpoints, and to enable or disable break conditions. The button functions are described in the following table.

Table 4.5 [Breakpoints] Window Button Operation

Button Name	Description
[Add]	Sets break conditions. Clicking this button will display the [Break] dialog box, enabling break conditions to be set.
[Edit]	Changes break conditions. Select break conditions to be changed and click this button. The break condition setting dialog box will be displayed, enabling the break condition to be changed.
[Delete]	Clears break conditions. Select break conditions to be cleared and click this button.
[Del All]	Clears all break conditions.
[Disable] ([Enable])	Enables or disables break conditions. Select break conditions to be enabled or disabled and click this button.
[Help]	Displays help information.

Similar button operations can also be performed with the pop-up menu displayed by clicking the view area with the right mouse button.

Breakpoint setting conditions are displayed in the lower area of the window.

A software breakpoint is displayed as follows:

```
Break Point xxx/255
```

(`xxx` represents the number of breakpoints in decimal that have been set.)

A hardware break condition is displayed as follows:

`Break Condition   Used x, y, z...` (`x, y, z...` represent an integer from 1 to 3.)

When no breakpoint is set, the following is displayed:

```
Break Point 0/255
Break Condition   Not Used
```

4.2.4 [Break] Dialog Box

Function:

This dialog box displays the break condition settings.

Window:

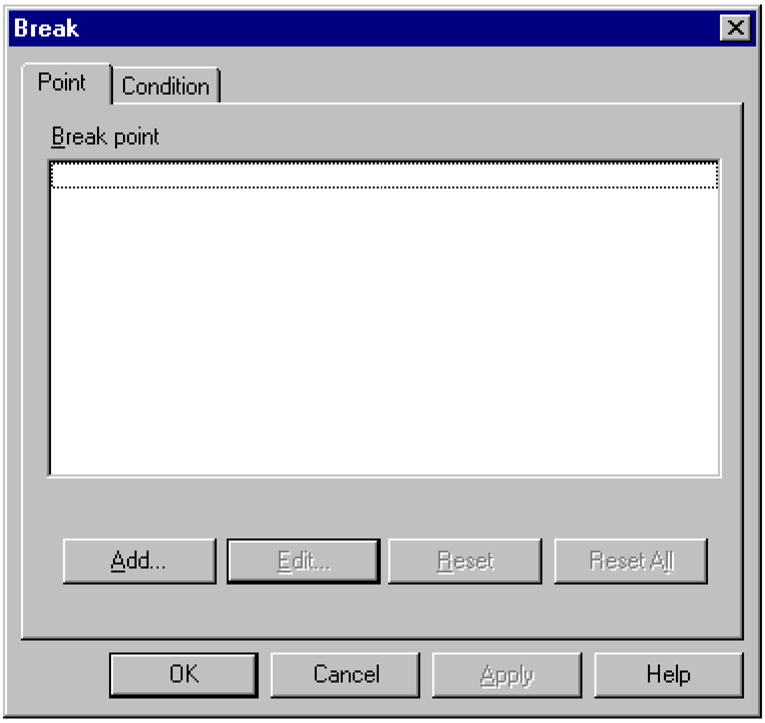


Figure 4.6 [Break] Dialog Box

Description:

The [Break] dialog box consists of the pages listed in table 4.6.

Table 4.6 [Break] Dialog Box Pages

Page Name	Description
[Point]	Displays software breakpoint settings.
[Condition]	Displays Break Condition settings.

The dialog boxes which set or modify break conditions can be displayed from the pages above.

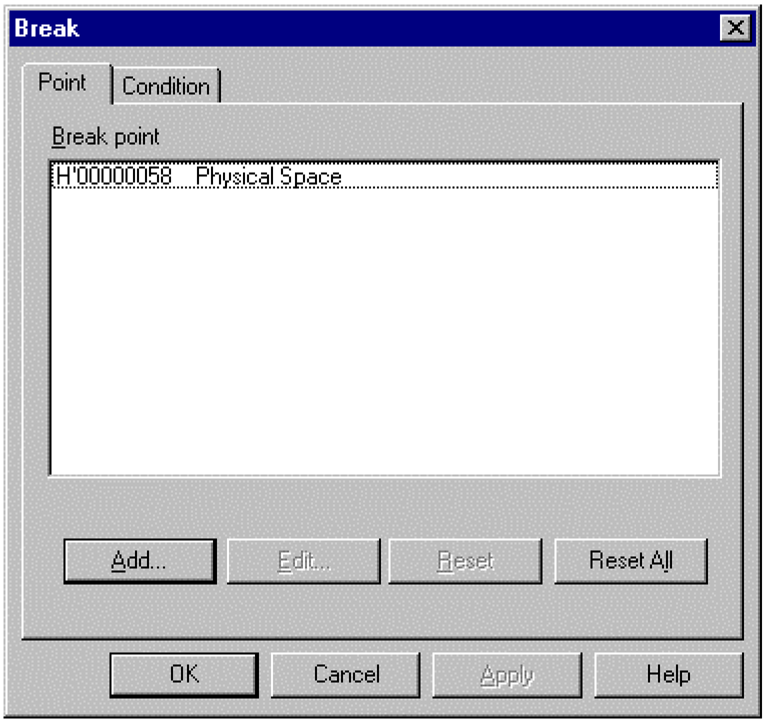
Clicking the [OK] button will close this dialog box.

(1) [Point] Page ([Break] Dialog Box)

Function:

This page displays software breakpoint settings. In this page, software breakpoints can be set, changed, and cleared.

Window:



H'00000058 Physical Space

Figure 4.7 [Point] Page ([Break] Dialog Box)

Description:

Table 4.7 [Point] Page Options

Option	Description
[Break point] list box	Lists the software breakpoints currently being set. The display contents are <breakpoint address> and <address space>. <address space> is displayed as follows: • Physical Space • Virtual Space ASID = D'xxx (xxx is the ASID value displayed in decimal form.)
[Add...] button	Sets software breakpoints. Clicking this button displays the [Break Point] dialog box.
[Edit...] button	Changes the software breakpoint selected in the [Break point] list box. Clicking this button displays the [Break Point] dialog box.
[Reset] button	Clears the software breakpoint selected in the [Break Point] list box.
[Reset All] button	Clears all software breakpoints displayed in the [Break Point] list box.

Related Commands:

BREAKPOINT command
BREAKPOINT_CLEAR command
BREAKPOINT_ENABLE command
BREAKPOINT_DISPLAY command

(2) [Condition] Page ([Break] Dialog Box)

Function:

This page displays the Break Condition 1 to Break Condition 3 settings. These conditions can also be set or cleared in this page.

Note: The function will be different according to the MCUs used.

Window:

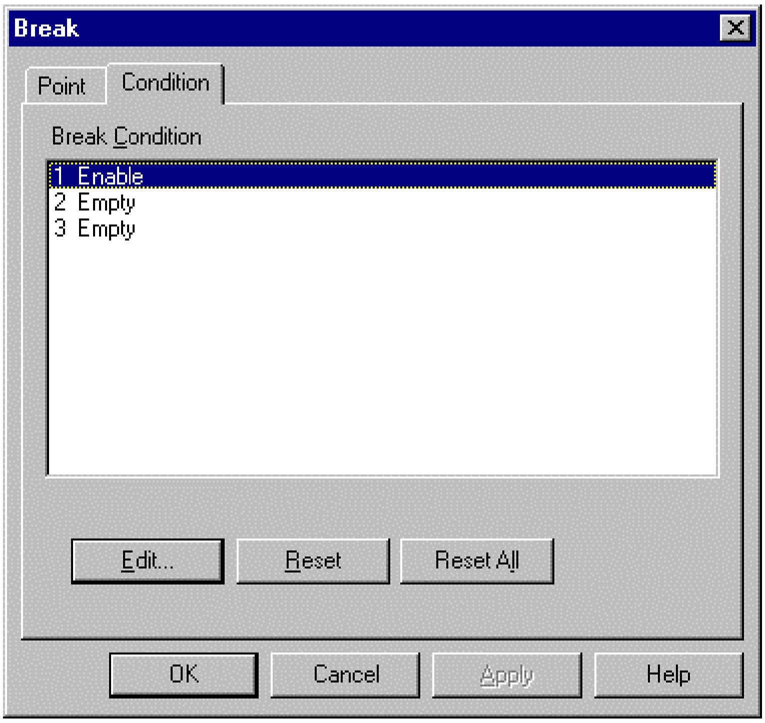


Figure 4.8   [Condition] Page ([Break] Dialog Box)

Description:

Table 4.8 [Condition] Page Options

Option	Description
[Break Condition] list box	Displays the Break Condition 1 to Break Condition 3 settings. The display at system initiation is as follows: When conditions are set, Enable is displayed. When no conditions are set, Empty is displayed. 1 Empty (setting of Break Condition 1) 2 Empty (setting of Break Condition 2) 3 Empty (setting of Break Condition 3)
[Edit...] button	Changes the Break Condition 1 to Break Condition 3 settings selected in the [Condition] list box. Clicking this button displays the [Break Condition 1] to [Break Condition 3] dialog boxes.
[Reset] button	Clears the Break Condition 1 to Break Condition 3 settings selected in the [Condition] list box.
[Reset All] button	Clears all Break Condition 1 to Break Condition 3 settings in the [Condition] list box.

Related Commands:

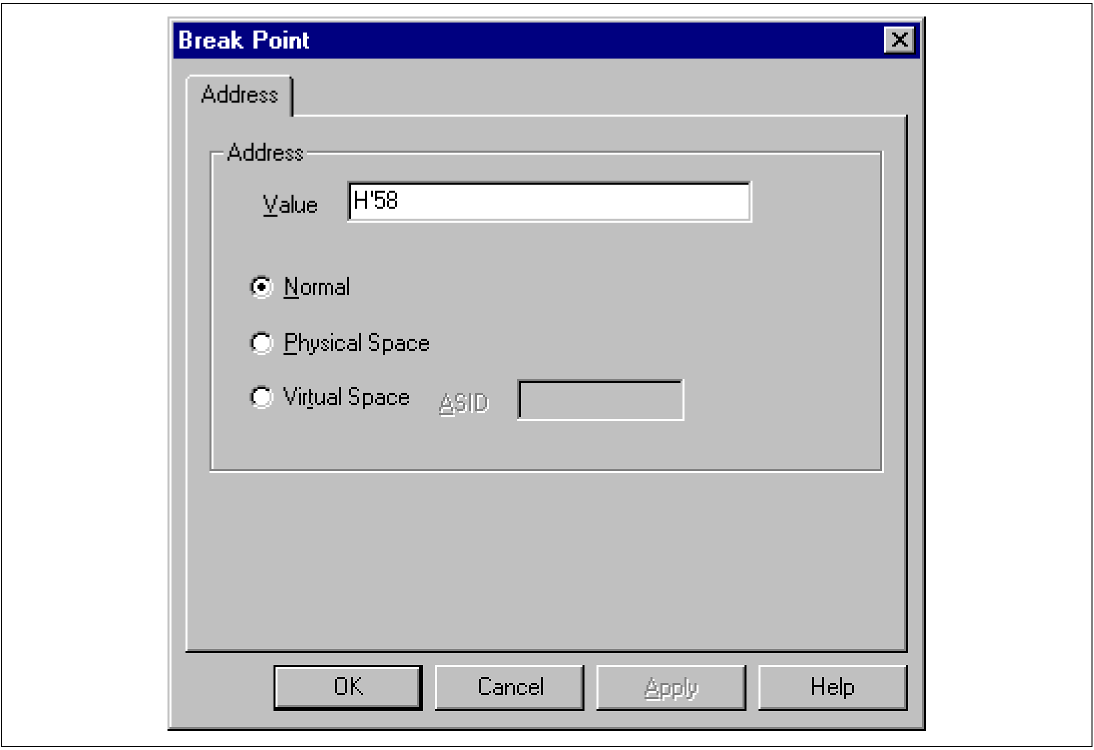
BREAKCONDITION_CLEAR command
BREAKCONDITION_DISPLAY command
BREAKCONDITION_ENABLE command
BREAKCONDITION_SET command

4.2.5 [Break Point] Dialog Box

Function:

This dialog box sets software breakpoints.

Window:



Address

Value: H'58

[✓ Normal] Physical Space Virtual Space

ASID:

Figure 4.9 [Break Point] Dialog Box

Description:

The [Break Point] dialog box consists only of the [Address] page. This dialog box sets address conditions and address areas. The [Address] page options are as follows:

Table 4.9   [Address] Page Options

Option	Description
[Value] edit box	Sets a breakpoint address with a number or a symbol.
[Normal] radio button	Does not set an address area
[Physical Space] radio button	Shows that the break condition is the physical area.
[Virtual Space] radio button	Shows that the break condition is the virtual area.
[ASID] edit box	Sets an ASID value when the breakpoint address is in the virtual area. Nothing is set as default.

Clicking the [OK] button enables breakpoints to be set. If the [Cancel] button is clicked, this dialog box is closed without setting breakpoints.

Related Commands:

BREAKPOINT command
BREAKPOINT_CLEAR command
BREAKPOINT_DISPLAY command
BREAKPOINT_SET command

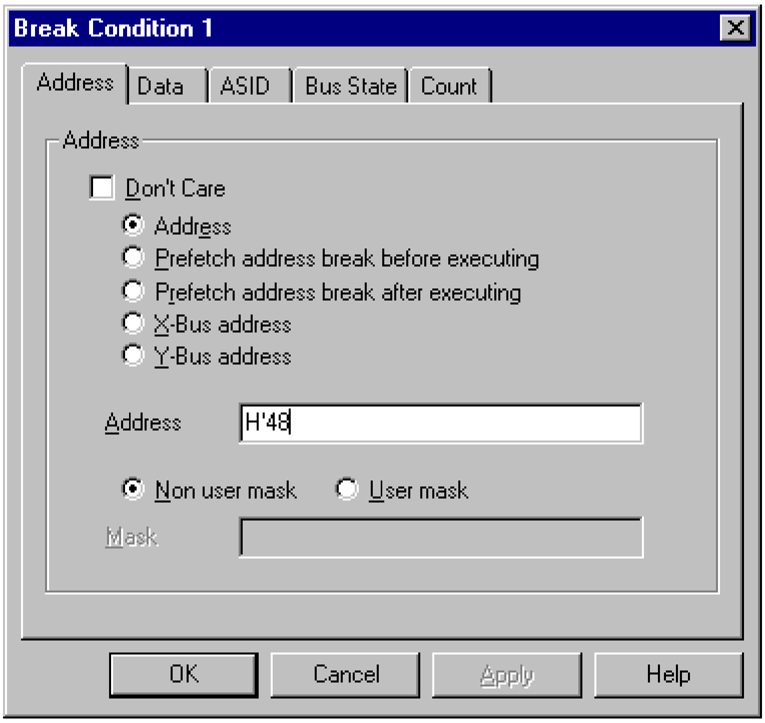
4.2.6 [Break Condition] Dialog Box

Function:

This dialog box sets hardware break conditions.

Note: The function will be different according to the MCUs used.

Window:



Address

[ ] Don't Care

[● Address] ○ Prefetch address break before executing ○ Prefetch address break after executing ○ X-Bus address ○ Y-Bus address

Address: H'48

[● Non user mask] ○ User mask

Mask:

Figure 4.10 [Break Condition] Dialog Box

Description:

The [Break Condition] dialog box is composed of the pages listed in table 4.10. Each page can set conditions that stop program execution.

Table 4.10 [Break Condition] Dialog Box Pages

Page Name	Function
[Address]	Sets the address conditions of Break Condition 1 and Break Condition 2.
[Data]	Sets the data conditions of Break Condition 1.
[ASID]	Sets the ASID conditions of Break Condition 1 and Break Condition 2.
[Bus State]	Sets the bus state conditions and read/write cycle conditions of Break Condition 1 and Break Condition 2.
[General]	Sets the conditions of Break Condition 3.

Contents to be set by each page are described in section 4.2.7, [Break Condition] Dialog Box Pages.

Clicking the [OK] button sets the hardware break conditions. If the [Cancel] button is clicked, the dialog box is closed without setting the hardware break conditions.

Related Commands:

BREAKCONDITION_CLEAR command
BREAKCONDITION_DISPLAY command
BREAKCONDITION_ENABLE command
BREAKCONDITION_SET command

4.2.7 [Break Condition] Dialog Box Pages

Function:

The [Break Condition] dialog box pages allow a number of hardware break conditions to be set. Table 4.11 shows all the [Break Condition] dialog box pages.

Note: The function will be different according to the MCUs used.

Table 4.11 [Break Condition] Dialog Box Pages

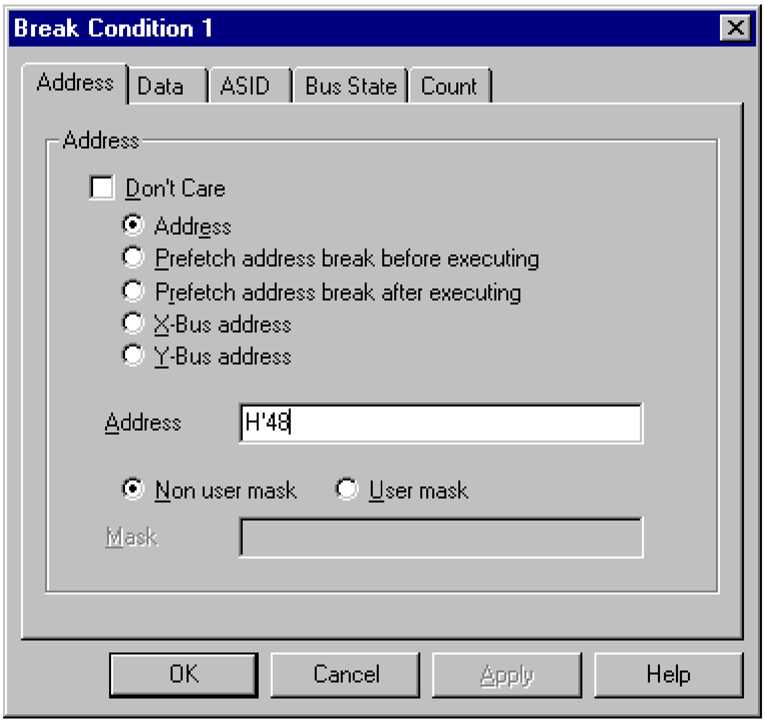
Page Name	Function
[Address]	Sets the address conditions of Break Condition 1 and Break Condition 2.
[Data]	Sets the data conditions of Break Condition 1. (Data conditions of Break Condition 2 and Break Condition 3 are not displayed.)
[ASID]	Sets the ASID conditions of Break Condition 1 and Break Condition 2.
[Bus State]	Sets the bus state conditions and read/write cycle conditions of Break Condition 1 and Break Condition 2.
[General]	Sets the conditions of Break Condition 3.

(1) [Address] Page ([Break Condition] Dialog Box)

Function:

This page sets the address bus conditions.

Window:



Don't Care [ ]

[✓ Address]

Prefetch address break before executing

Prefetch address break after executing

X-Bus address

Y-Bus address

Address: H'48

[✓ Non user mask] User mask

Mask:

Figure 4.11 [Address] Page ([Break Condition 1] Dialog Box)

Description:

Table 4.12 [Address] Page Options

Option	Description
[Don’t Care] check box	Does not set address conditions.
[Address] radio button	Sets use of the normal address bus as break conditions.
[Prefetch address break before executing] radio button	Sets a break before prefetched address execution as break conditions.
[Prefetch address break after executing] radio button	Sets a break after prefetched address execution as break conditions.
[X-bus address] radio button	Sets the X address bus as a break condition. Can be set only with Break Condition 1.
[Y-bus address] radio button	Sets the Y address bus as a break condition. Can be set only with Break Condition 1.
[Address] edit box	Sets the address bus value with a number or a symbol.
[Non user mask] radio button	Sets no mask conditions.
[User mask] radio button	Sets mask conditions.
[Mask] edit box	Sets the mask bits if [User mask] is selected. For masked bits, the break condition is satisfied regardless of the address values.

Note: This page is displayed when the conditions of Break Condition 1 and Break Condition 2 are set.

A page name to be displayed and the contents of an option that can be set will change depending on the radio button selected.

Table 4.13 Address Options

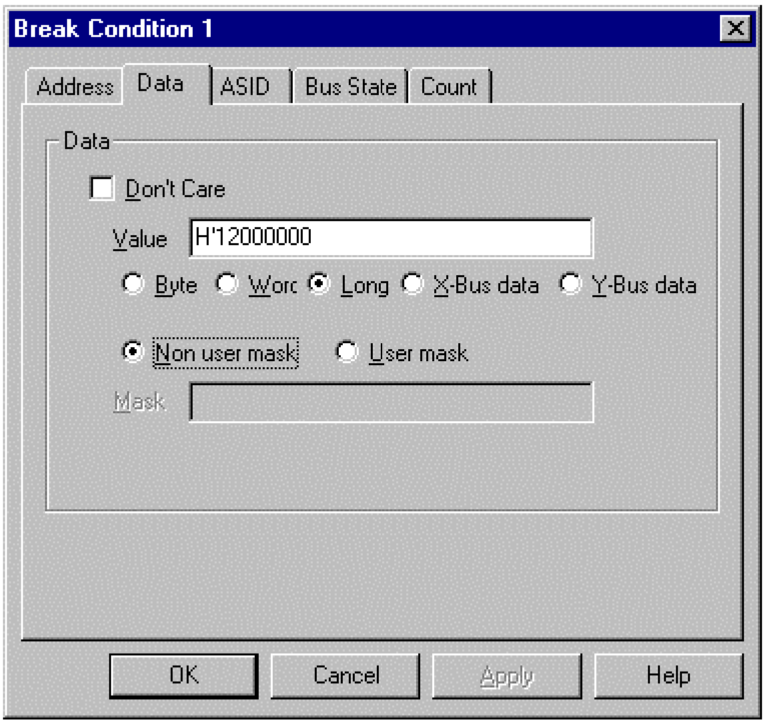
Option	Description
[Address] radio button	All pages and masks can be set.
[Only program fetched address] radio button	All pages can be set; however, no mask can be set.
[Only program fetched address after] radio button	Only the [ASID] page can be set.

(2) [Data] Page ([Break Condition] Dialog Box)

Function:

This page sets the data bus conditions.

Window:



Data

[ ] Don't Care

Value: H'12000000

Byte [✓ Long] Word X-Bus data Y-Bus data

[✓ Non user mask] User mask

Mask:

Figure 4.12 [Data] Page ([Break Condition 1] Dialog Box)

Description:

Table 4.14 [Data] Page Options

Option	Description
[Don’t Care] check box	Does not set data conditions.
[Value] edit box	Sets the data bus value with a number.
[Byte] radio button	Sets byte data access cycles.
[Word] radio button	Sets word data access cycles.
[Long] radio button	Sets longword data access cycles.
[X-bus data]	Sets X-bus data access cycles.
[Y-bus data]	Sets Y-bus data access cycles.
[Non user mask] radio button	Does not set mask conditions.
[User mask] radio button	Sets mask conditions.
[Mask] edit box	Sets the mask bits when [User mask] is selected. Mark a bit to be masked with *. For masked bits, the break conditions will be satisfied regardless of the data values.

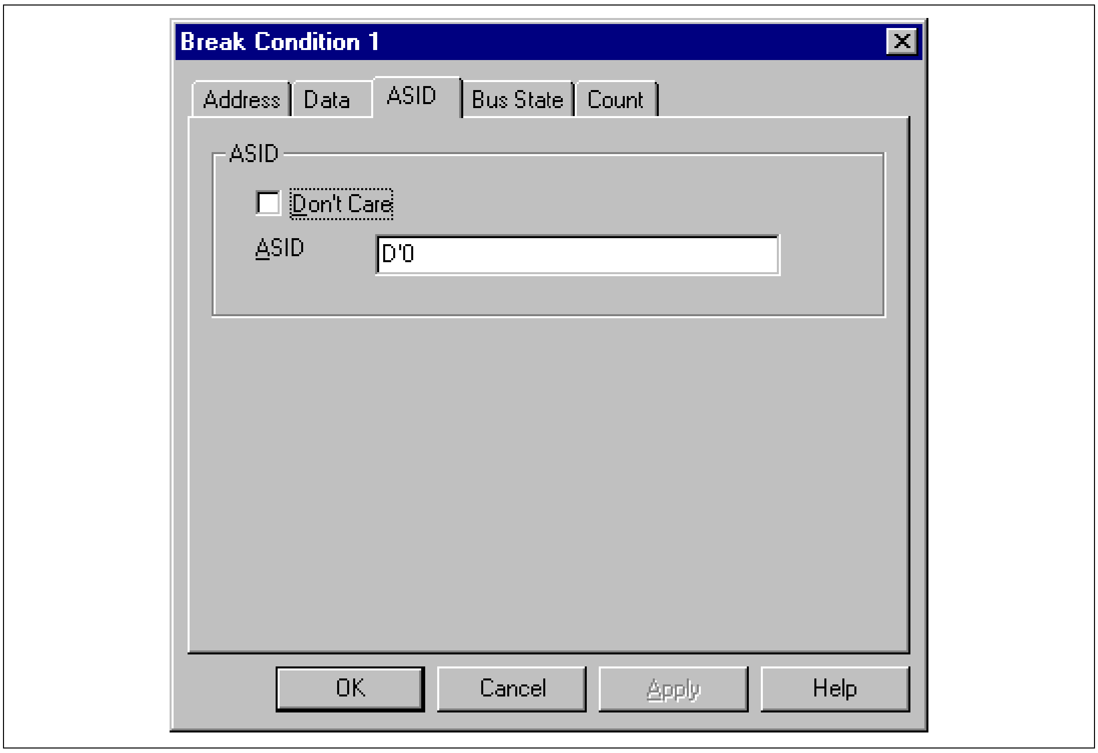
Note: This page is displayed when the conditions of Break Condition 1 are set.

(3) [ASID] Page ([Break Condition] Dialog Box)

Function:

This page sets the ASID conditions.

Window:



ASID D'0

Figure 4.13 [ASID] Page ([Break Condition] Dialog Box)

Description:

Table 4.15 [ASID] Page Options

Option	Description
[Don't Care] check box	Does not set ASID conditions.
[ASID] edit box	Sets the ASID condition value. The default is 0.

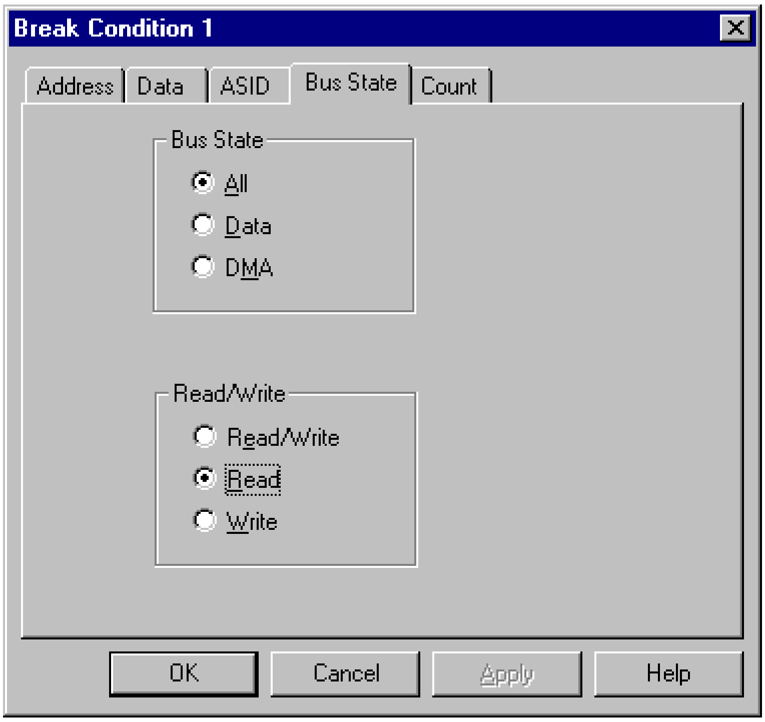
Note: This page is displayed when the conditions of Break Condition 1 and Break Condition 2 are set.

(4) [Bus State] Page ([Break Condition] Dialog Box)

Function:

This page sets bus state conditions and read/write cycle conditions.

Window:



Bus State

[✓ All] Data DMA

Read/Write

Read/Write [✓ Read] Write

Figure 4.14 [Bus State] Page ([Break Condition] Dialog Box)

Description:

Table 4.16 [Bus State] Page Options

Option	Description
[Bus State] group box	Sets the bus state conditions by the following options.
[All] radio button	Sets the bus state conditions as break conditions.
[Data] radio button	Sets the execution cycle as break conditions.
[DMA] radio button	Sets DMA cycles as a break condition.
[Read/Write] group box	Sets the read/write cycle conditions by the following options.
[Read/Write] radio button	Sets the read/write cycle conditions as break conditions.
[Read] radio button	Sets read cycles as break conditions.
[Write] radio button	Sets write cycles as break conditions.

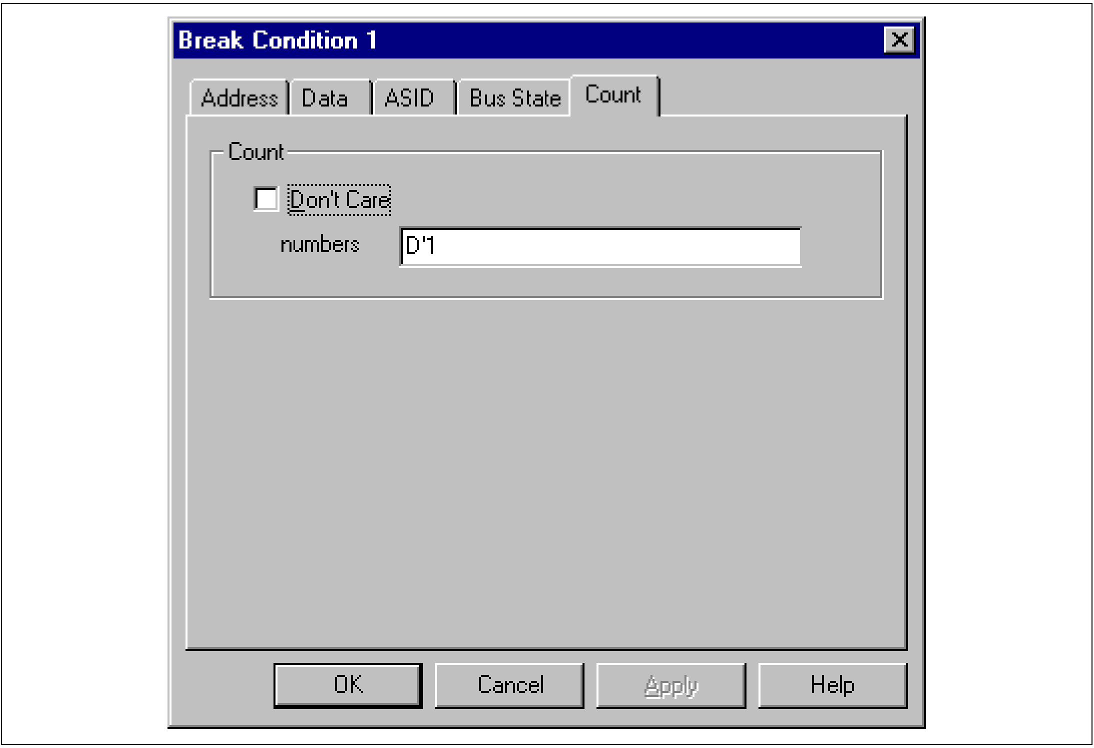
Note: This page is displayed when the conditions of Break Condition 1 and Break Condition 2 are set.

(5) [Count] Page ([Break Condition] Dialog Box)

Function:

This page sets the conditions for Break Condition 3.

Window:



Count

Don't Care: ☐

numbers: D'1

Figure 4.15 [Count] Page ([Break Condition] Dialog Box)

Table 4.17 [Count] Page Options

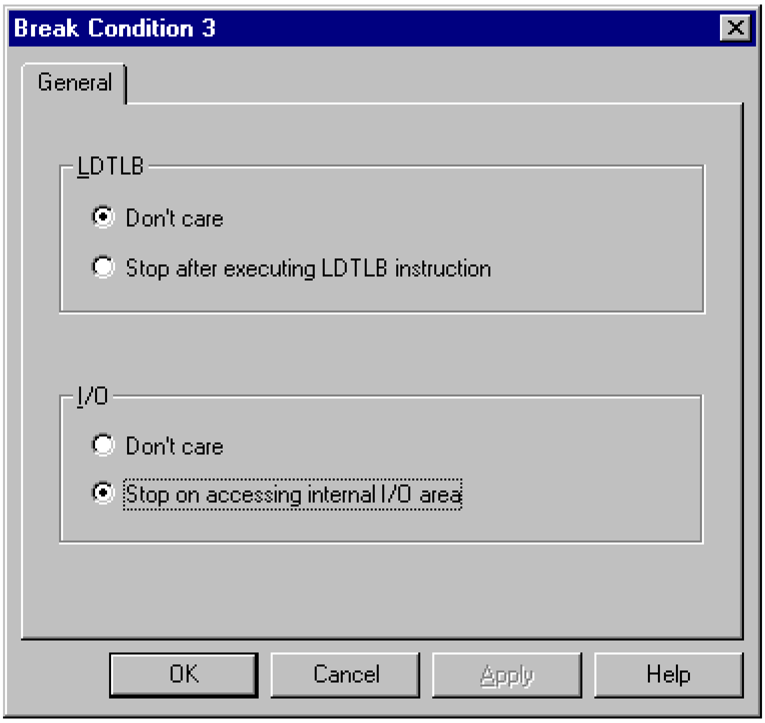
Option	Description
[Don't Care] check box	Sets no satisfaction count conditions.
Input area	Sets the satisfaction count as a break condition. The maximum count is 4,095. Breaks when the conditions set by the [Break Condition] dialog box for the specified times are satisfied. The default is D'1.

(6) [General] Page ([Break Condition] Dialog Box)

Function:

This page sets the conditions for Break Condition 3.

Window:


Break Condition 3

General

LDTLB
[✓ Don't care]
Stop after executing LDTLB instruction

I/O
Don't care
[✓ Stop on accessing internal I/O area]

Figure 4.16 [General] Page ([Break Condition] Dialog Box)

Description:

Table 4.18 [General] Page Options

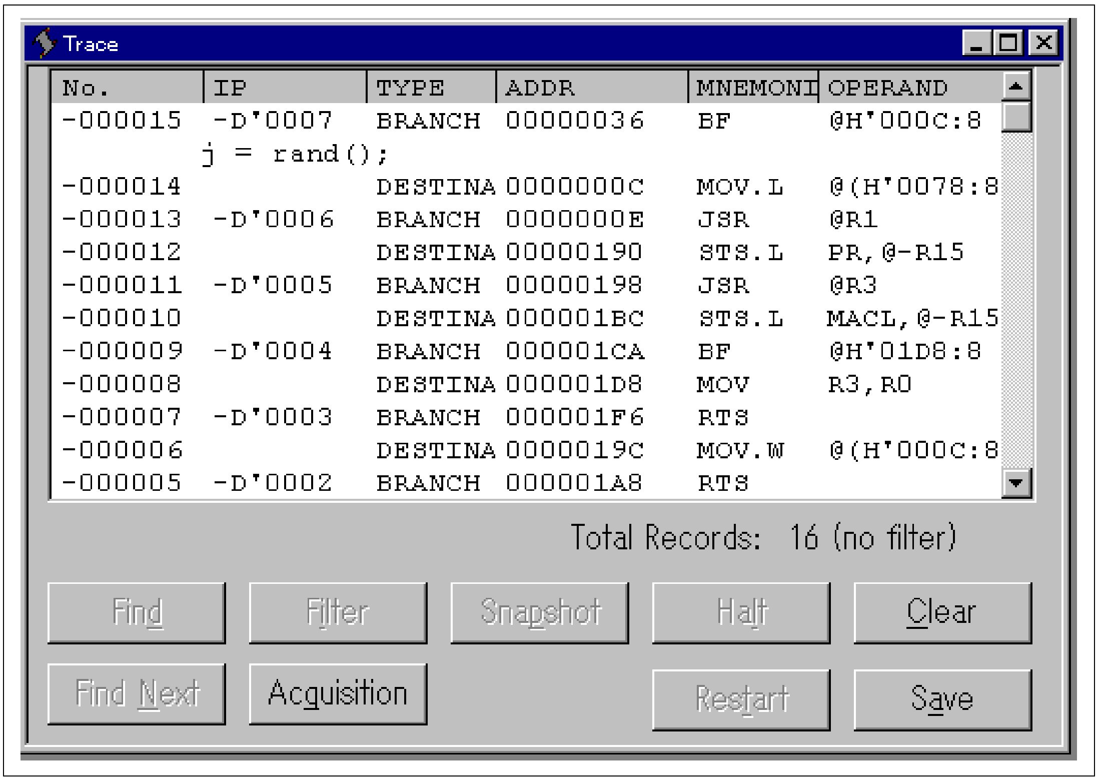
Option	Description
[LDTLB] group box	Sets the break conditions at LDTLB instruction execution with the following options.
[Don’t Care] radio button	Does not set break conditions when the LDTLB instruction is executed.
[Stop after executing LDTLB instruction] radio button	Sets the LDTLB instruction execution as break conditions.
[I/O] group box	Sets the break conditions at internal I/O-area access with the following options.
[Don’t Care] radio button	Does not set break conditions when the internal I/O area is accessed.
[Stop on accessing internal I/O area] radio button	Sets the internal I/O area access as break conditions.

4.2.8 [Trace] Window

Function:

This window displays the trace buffer contents.

Window:



```
No.      IP       TYPE    ADDR      MNEMONI OPERAND
-000015  -D'0007  BRANCH  00000036  BF      @H'000C:8
        j = rand();
-000014           DESTINA 0000000C  MOV.L   @(H'0078:8
-000013  -D'0006  BRANCH  0000000E  JSR     @R1
-000012           DESTINA 00000190  STS.L   PR,@-R15
-000011  -D'0005  BRANCH  00000198  JSR     @R3
-000010           DESTINA 000001BC  STS.L   MACL,@-R15
-000009  -D'0004  BRANCH  000001CA  BF      @H'01D8:8
-000008           DESTINA 000001D8  MOV     R3,R0
-000007  -D'0003  BRANCH  000001F6  RTS
-000006           DESTINA 0000019C  MOV.W   @(H'000C:8
-000005  -D'0002  BRANCH  000001A8  RTS
```

Total Records: 16 (no filter)

Figure 4.17 [Trace] Window

Description:

This window displays the trace buffer contents. The items listed in table 4.19 are displayed.

Table 4.19 [Trace] Window Display Items

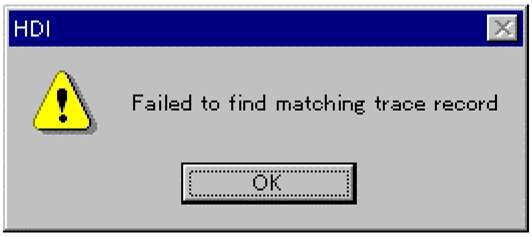
Item	Description
[No]	Displays the number in ascending order as the trace stop point is 0 (signed decimal).
[IP]	Displays the instruction pointer (signed decimal).
[TYPE]	For the branch instruction trace, displays the information type, that is, branch source or branch destination. BRANCH: Branch source DESTINATION: Branch destination.
[ADDR]	For the branch instruction trace, displays the branch source or branch destination address.
[MNEMONIC]	Displays the execution instruction mnemonic.
[OPERAND]	Displays the execution instruction operand.
[Total Records]	Displays the number of trace information displayed in the [Trace] window.
[Halt] button	Displays the trace during user program execution (no realtime operation).

Notes:

1. In some cases, the emulator address may be acquired by trace. In such a case, the following message will be displayed. Ignore this address because it is not a user program address.

```
        *** EML ***
```

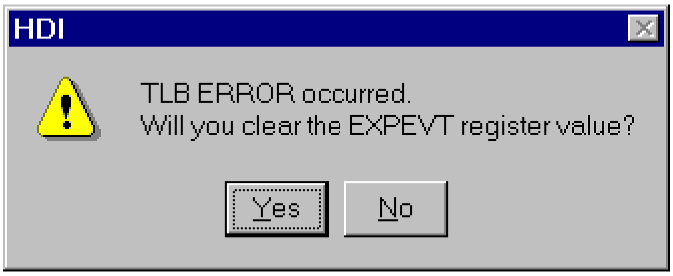
2. If no branch trace information is acquired, the following message will be displayed.


HDI

Failed to find matching trace record

Figure 4.18 Failed to find matching trace record Message Box

3. If a TLB error occurs while the acquired trace information is displayed, the following error message will be displayed:


HDI

TLB ERROR occurred.
Will you clear the EXPEVT register value?

Figure 4.19 TLB Error Message Box

4. The [Halt] button is active only when the [Trace] window is opened during user program execution.

Related Command:

TRACE_DISPLAY command

4.2.9 [Trace Acquisition] Dialog Box

Function:

This dialog box sets trace acquisition conditions.

Table 4.20 [Trace Acquisition] Page Options

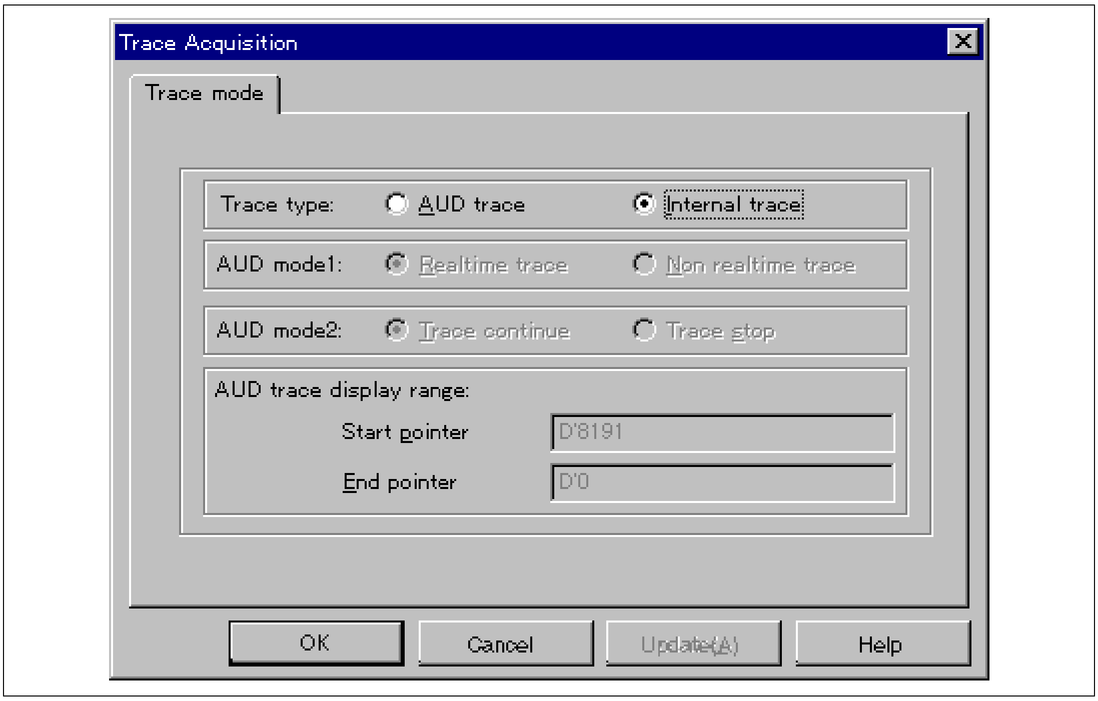
Page Name	Function
[Trace Mode]	Sets the conditions of trace mode.

(1) [Trace Mode] Page ([Trace Acquisition] Dialog Box)

Function:

This page sets the conditions for trace mode.

Window:



Trace Acquisition

Trace mode

Trace type: ( ) AUD trace [✓ Internal trace]

AUD mode1: [✓ Realtime trace] ( ) Non realtime trace

AUD mode2: [✓ Trace continue] ( ) Trace stop

AUD trace display range:

Start pointer: D'8191

End pointer: D'0

Figure 4.20 [Trace mode] Page ([Trace Acquisition] Dialog Box)

Description:

Table 4.21 [Trace mode] Page Options

Option	Description
[AUD trace] radio button	Uses AUD trace functions. By default, this box is not checked.
[Internal trace] radio button	Uses the internal trace functions. By default, this box is checked.
[Realtime trace] radio button	When the next branch occurs while the trace information is being acquired, the information is overwritten and the latest trace information is acquired. The user program can be executed in realtime, but some trace information will not be acquired. By default, this box is checked.
[Non realtime trace] radio button	When the next branch occurs while the trace information is being acquired, the CPU stops operations until the information is acquired. The user program is not executed in realtime. By default, this box is not checked.
[Trace continue] radio button	When the trace buffer becomes full, this function always overwrites the oldest trace information to acquire the latest trace information.
[Trace stop] radio button	When the trace buffer becomes full, the trace information is not acquired.
[AUD trace display range] group box	Inputs the start or end pointer value in the trace display range as numerical values. By default, the start pointer is –D’8191 and the end pointer is –D’0000. In the PCMCIA card, –D’8191 to D’0 can be set to the trace pointer. In the PCI card, –D’32767 to D’0 can be set.

Related Command:

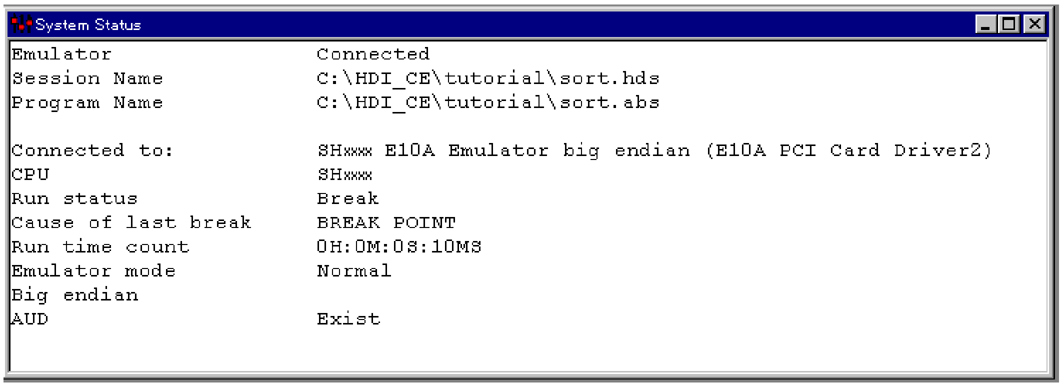
AUD_MODE command

4.2.10 [System Status] Window

Function:

This window lists information, such as conditions that have been set to the emulator and execution results.

Window:



```
Emulator                  Connected
Session Name              C:\HDI_CE\tutorial\sort.hds
Program Name              C:\HDI_CE\tutorial\sort.abs

Connected to:             SHxxxx E10A Emulator big endian (E10A PCI Card Driver2)
CPU                       SHxxxx
Run status                Break
Cause of last break       BREAK POINT
Run time count            0H:0M:0S:10MS
Emulator mode             Normal
Big endian
AUD                       Exist
```

Figure 4.21 [System Status] Window

Description:

The items listed in the following table are displayed in the [System Status] window.

Table 4.22 [System Status] Window Display Items

Item	Description
Emulator	Displays whether the emulator is connected.
Session Name	Displays a session file name.
Program Name	Displays the name of a connected load module.
Connected to	Displays the setting to be used.
CPU	Displays a target MCU.
Run status	Displays whether the emulator is in execution. During execution, RUNNING is displayed. During halt, Break is displayed.
Cause of last break	Displays the cause of a break.
Run time count	Displays a program execution time. The display format is as follows: D'0H:0M:0S:10MS (H: hour, M: minute, S: second, and MS: millisecond)
Emulator mode	Displays the operating mode of the emulator.
Big Endian/Little Endian	Displays the endian state (Big Endian or Little Endian). In this example, Big endian is displayed.
AUD	Displays whether the AUD function can be used. In this example, the AUD function can be used.

Section 5 Command-line Functions

5.1 Table and Symbol Description

This section describes the format used in section 5.2, Command Descriptions. The descriptions of some commands are given over two or more pages.

5.1.1 Format

The input format for each command is as follows. Characters shown in bold-italics are to be input.

[ ] : Parameters enclosed by [ ] can be omitted.

< > : Contents shown in < > are set.

< >=: The parameter to the left of the "=" sign is input in the format shown to the right.

| : This represents a non-exclusive selection.

| | : This represents an exclusive selection.

The command parameter details are described in the parameter table.

5.1.2 Parameter Input

Numerical Parameters:

A binary, octal, decimal, or hexadecimal value, a symbol, or a formula can be input. A symbol can contain up to 32 characters. Terms in a formula are separated with operators (such as + or –).

Keyword Parameters:

One of the bold characters given in the description column of the table can be input. If a character string not shown in the description is input, an error will occur.

Character-String Parameters:

Character-string parameters are used to input mask data or a file name. In the mask data, set a radix (H’: hexadecimal or B’: binary) at the top of a character string and set * at the digit to be masked.

5.1.3 Examples

These are actual input examples. For commands whose execution results in a specific display output, an example of the display is given.

5.1.4 Related Items

Related E10A HDI commands (abbreviations) and dialog boxes are shown. (Refer to section 4, Descriptions of Windows.)

5.2 Command Descriptions

The command list of the E10A HDI is shown below.

Table 5.1 E10A HDI Commands

No	Command	Abb.	Function
1	AUD_CLOCK	AUCL	Sets the AUD clock (AUDCK).
2	AUD_MODE	AUM	Sets AUD trace conditions.
3	AUD_TRACE	AUT	Displays trace information.
4	BREAKCONDITION_ CLEAR	BCC	Clears hardware breakpoints that have been set.
5	BREAKCONDITION_ DISPLAY	BCD	Displays hardware breakpoints settings.
6	BREAKCONDITION_ ENABLE	BCE	Enables or disables hardware breakpoints that have been set.
7	BREAKCONDITION_ SET	BCS	Sets hardware breakpoints.
8	BREAKPOINT	BP	Sets software breakpoints.
9	BREAKPOINT_CLEAR	BC	Clears software breakpoints that have been set.
10	BREAKPOINT_DISPLAY	BD	Displays software breakpoints that have been set.
11	BREAKPOINT_ENABLE	BE	Enables or disables software breakpoints that have been set.
12	DEVICE_TYPE	DE	Displays MCU type currently selected.
13	GO_OPTION	GP	Sets or displays the emulation mode during user program execution.
14	JTAG_CLOCK	JCK	Displays and sets a JTAG clock (TCK) frequency.
15	MEMORYAREA_SET	MAS	Displays and sets memory area at command input, such as load, verify, save, memory display, or memory change.

Table 5.1 E10A HDI Commands (cont)

No	Command	Abb.	Function
16	REFRESH	RF	Updates the HDI memory information to the latest contents.
17	RESTART	RST	Restarts the emulator.
18	STATUS	STS	Displays emulator state information.
19	TRACE_DISPLAY	TD	Displays acquired trace buffer information.
20	UBC_MODE	UM	Sets or displays UBC use states.
21	VPMAP_CLEAR	VC	Clears the emulator address translation (VP_MAP) table which has been set.
22	VPMAP_DISPLAY	VD	Displays the emulator address translation (VP_MAP) table.
23	VPMAP_ENABLE	VE	Enables or disables the emulator address translation (VP_MAP) table.
24	VPMAP_SET	VS	Sets emulator address translation (VP_MAP) table.

5.2.1 AUD_CLOCK:AUCL

Description:

Sets or displays the AUD clock (AUDCK) values that have been set.

Format:

```
aucl [<option>]
```

<option> = <aud_clock>

Table 5.2 AUD_CLOCK Command Parameter

Parameter	Type	Description
<aud_clock>	Numerical value	Sets values from 1 to 7.
		1: 5 MHz (PCI), 7.5 MHz (PCMCIA)
		2: 10 MHz (PCI), 15 MHz (PCMCIA)
		3: 20 MHz (PCI), 30 MHz (PCMCIA)
		4: 30 MHz (PCI), 60 MHz (PCMCIA)
		5: 40 MHz (PCI)
		6: 50 MHz (PCI)
		7: 60 MHz (PCI)

Notes: 1. When <option> is omitted, the AUD clock (AUDCK) values that have been set are displayed.

2. The range of frequencies that the AUD operates under is different according to the MCUs used. For details, refer to section 6.4.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK).

Examples:

To set AUD clock (AUDCK) to 15 MHz:

```
>AUD_CLOCK 2 (RET)
AUD CLOCK = 15MHz
```

The AUD clock (AUDCK) is displayed:

```
>AUD_CLOCK (RET)
AUD CLOCK = 15MHz
```

Related Item:

[Configuration] dialog box

5.2.2 AUD_MODE:AUM

Description:

Sets or displays AUD trace acquisition conditions.

Format:

```
aum [<option1>] [<option2>]
```

<option1> = mode<mode>
<option2> = full<full>

Table 5.3 AUD_MODE Command Parameter

Parameter	Type	Description
<mode>	Keyword	Selects the trace mode. **N**: Internal trace **F**: Non realtime trace **R**: Realtime trace
<full>	Keyword	Continues or stops emulation when the trace memory is full. **C**: Always overwrites the oldest information to acquire the latest information. **S**: When the memory is full, information acquisition stops.

Note: When <option1> and <option2> are omitted, the current setting conditions are displayed.

Examples:

To select realtime trace mode and set continue option:

```
>aum mode R full c (RET)
```

To display settings:

```
>aum (RET)
 mode = Realtime trace, continue
```

To use internal trace mode:

```
>aum (RET)
 mode = Internal trace
```

Related Item:

[Trace Acquisition] dialog box

5.2.3 AUD_TRACE:AUT

Description:

Displays the trace information.

Format:

```
aut [<option1>] [<option2>]
```

<option1> = start<start_pointer>
<option2> = end<end_pointer>

Table 5.4 AUD_TRACE Command Parameter

Parameter	Type	Description
<start_pointer>	Numerical value (-n)	Start pointer value for trace display.
<end_pointer>	Numerical value (-m)	End pointer value for trace display.

Note: In the PCMCIA card, –D'8191 to D'0 can be set to the trace pointer. In the PCI card, –D'32767 to D'0 can be set.

When the internal trace is selected, the AUT command displays the information that has been acquired by using the AUD function.

Example:

To display trace information according to the information acquired during user program execution:

```
>AUD_TRACE (RET)
IP            TYPE          ADDR          MNEMONIC      OPERAND
-D'xxxxxx     BRANCH        ******10
              DESTINATION   01000020      MOV.L         R1, @R1
(a)           (b)           (c)           (d)           (e)
```

(a) Instruction pointer (signed decimal)
(b) Types of branch source or branch destination
BRANCH: Branch source
DESTINATION: Branch destination
(c) Address of instruction word
(d) Instruction mnemonic
(e) Instruction operand

Related Item:

[Trace] dialog box

5.2.4 BREAKCONDITION_CLEAR: BCC

Description:

Clears hardware breakpoints that have been set.

Format:

```
bcc [<channel>]
```

<channel> = channel <channel_number>

Table 5.5 BREAKCONDITION_CLEAR Command Parameter

Parameter	Type	Description
<channel number>	Numerical value	Hardware break channel number from **1** to **3**

Note: When <channel> is omitted, all hardware breakpoints that have been set are canceled.

Examples:

To clear all hardware breakpoints:

```
        >bcc (RET)
```

To clear all hardware breakpoints set at channel 2:

```
        >bcc channel 2 (RET)
```

Related Items:

BCD, BCE, and BCS commands
[Breakpoints] window
[Break] and [Break Condition] dialog boxes

5.2.5 BREAKCONDITION_DISPLAY: BCD

Description:

Displays hardware breakpoints that have been set. The display contents include a hardware breakpoint channel number, enable or disable of the setting, and setting conditions.

Format:

```
bcd [<channel>]
```

<channel> = channel <channel_number>

Table 5.6 BREAKCONDITION_DISPLAY Command Parameter

Parameter	Type	Description
<channel_number>	Numerical value	Hardware breakpoint channel number from **1** to **3**

Note: When <channel> is omitted, all hardware breakpoints that have been set are displayed.

Examples:

To display all hardware breakpoints settings:

```
>bcd (RET)
```

```
Break Condition 1:Enable data 20 long
Break Condition 2:Disable address 126
Break Condition 3:Disable LDTLB break
```

To display the hardware breakpoint set at channel 1:

```
>bcd channel 1 (RET)
```

```
Break Condition 1:Enable data 20 long
```

Related Items:

BCC, BCE, and BCS commands
[Breakpoints] window
[Break] and [Break Condition] dialog boxes

5.2.6 BREAKCONDITION_ENABLE: BCE

Description:

Enables or disables hardware breakpoints that have been set.

Format:

```
bce [<channel>] <mode>
```

<channel> = channel <channel_number>

Table 5.7 BREAKCONDITION_ENABLE Command Parameters

Parameter	Type	Description
<channel_number>	Numerical value	Hardware break channel number from **1** to **3**
<mode>	Keyword	Enables or disables hardware break settings. Set either of the following: **enable**: Enables hardware break settings. **disable**: Disables hardware break settings.

Note: When <channel> is omitted, all hardware breakpoints that have been set are enabled or disabled.

Examples:

To enable all hardware breakpoints:

```
>bce enable (RET)
```

To disable the hardware breakpoints set at channel 1:

```
>bce channel 1 disable (RET)
```

Related Items:

BCC, BCD, and BCS commands
[Breakpoints] window
[Break] and [Break Condition] dialog boxes

5.2.7 BREAKCONDITION_SET: BCS

Description:

Sets hardware breakpoints.

Note: The function will be different according to the MCUs used.

Format:

```
bcs <channel>  <option> [<option> ... ]
```

<channel>	=	channel <channel_number>
<option>	=	[<addropt> | <dataopt> | <asidopt> | <r/wopt> | <accessopt>] | [<countopt>] || [<ldtlbopt> | <ioopt>]
<addropt>	=	address <address> [<addrcycle>]|| address mask <maskdata> <addrcycle>
<dataopt>	=	data <data> <datawidth> || data mask <maskdata> <datawidth>
<asidopt>	=	asid <asid>
<r/wopt>	=	direction <r/w>
<accessopt>	=	access <access>
<countopt>	=	count <count>
<ldtlbopt>	=	ldtlb <lbtlb>
<ioopt>	=	io <io>

Table 5.8 BREAKCONDITION_SET Command Parameters

Parameter	Type	Description
<channel_number>	Numerical value	Hardware break condition channel number from **1** to **3** Specifiable options change depending on the channel number. **1**: <addropt>, <dataopt>, <asidopt>, <r/wopt>, and <accessopt> can be set. **2** : <addropt>, <asidopt>, <r/wopt>, and <accessopt> can be set. **3**: <ldtlbopt> and <ioopt> can be set.
<address>	Numerical value	Virtual address as an address bus value
<addrcycle>	Keyword	Address bus access conditions for program fetch cycles Set either of the following keywords: **pc**: Breaks before the address set by the <address> parameter is executed. When this keyword is set, only the <addropt> and <asidopt> cannot be set as conditions. In addition, when pc is set, the <maskdata> parameter cannot be set. **pcafter**: Breaks after the address set by the <address> parameter is executed. When this keyword is set, only the <addropt> and <asidopt> cannot be set as conditions. When pcafter is not set, the address bus during data access cycles and program fetch cycles is targeted. **x:** X-Bus address bus access **y:** Y-Bus address bus access
<maskdata>	Character string	Mask specification for desired address bus and data bus Set a radix (**H'** for hexadecimal or **B'** for binary) at the top of a character string and set * in the bit to the masked. Conditions are satisfied regardless of the values of masked bits.
<data>	Numerical value	Data bus value

Table 5.8 BREAKCONDITION_SET Command Parameters (cont)

Parameter	Type	Description
<datawidth>	Keyword	Data bus access conditions Set one of the following keywords: **byte**: byte access **word**: word access **long**: longword access **x:** X-Bus data bus access **y:** Y-Bus data bus access
<asid>	Numerical value	ASID value from 1 to H'FF.
<r/w>	Keyword	Bus cycle read/write conditions Set either of the following keywords: **read**: read cycles **write**: write cycles
<access>	Keyword	Bus cycle access type **dat**: execution cycles
<count>	Numerical value	Set satisfaction count from 1 to H'FFFF
<ldtlb>	Keyword	Set LDTLB instruction execution break **break:** Breaks when the LDTLB instruction is executed.
<io>	Keyword	Set internal I/O access condition as a break condition. **break:** Breaks when the internal I/O area is accessed.

Examples:

To set the following conditions for channel 1 hardware breakpoints:

<addropt> item: An address bus value of H'1000000,
<dataopt> item: D0 bit of the byte access data is 0,
<r/wopt> item: write cycle.

```
>bcs channel 1 address H'1000000 data mask B'*******0 byte
direction write (RET)
```

To set the following conditions for channel 2 hardware breakpoints:

<addropt> item: Deletes an address bus value of H'1000000 during the program fetch cycles, and breaks before execution.
<asidopt> item: The ASID value is H’0.

```
>bcs channel 2 address H'1000000 pc asid H'0 (RET)
```

To set the following conditions for channel 1 hardware breakpoints:

<addropt> item: Deletes an address bus value of H'1000000 during the program fetch cycles with a mask set to the lower 10 bits, and breaks after execution,
<asidopt> item: H'10 to the ASID value.

```
>bcs channel 1 address H'1000000 pcafter m1 asid H'10 (RET)
```

To set the following conditions for channel 2 hardware breakpoints:

<accessopt> item: Execution cycles,
<r/wopt> item: Read cycles.

```
>bcs channel 2 access dat direction read (RET)
```

To set the following conditions for channel 3 hardware breakpoints:

<ldtlbopt> item: Breaks during LDTLB instruction execution,
<ioopt> item: Breaks when the internal I/O area is accessed.

```
>bcs channel 3 ldtlb break io (RET)
```

Related Items:

BCC, BCD, BCE, and TM commands
[Breakpoints] window
[Break] and [Break Condition] dialog boxes

5.2.8 BREAKPOINT: BP

Description:

Sets software breakpoints.

Note: The function will be different according to the MCUs used.

Format:

```
bp <address> [<address_space> [<asidopt>]]
```

<address_space> = space <space>
<asidopt> = asid <asid>

Table 5.9 BREAKPOINT Command Parameters

Parameter	Type	Description
<address>	Numerical value	Breakpoint address When an odd address is set, the address is rounded down to an even address.
<space>	Keyword	Breakpoint address area Set either of the following: **physical**: physical address **virtual**: virtual address
<asid>	Numerical value	ASID value of a breakpoint when virtual is set to the <space> parameter.

Note: When virtual is set and the <asidopt> item is omitted in the <address_space> item, a breakpoint is set to a virtual address corresponding to the ASID value at command input.

Examples:

To set a software breakpoint at physical address H'10002C8:

```
>bp H'10002C8 space physical (RET)
```

To set a software breakpoint at logical address H'1000000, whose ASID value is H'10:

```
>bp H'1000000 space virtual asid H'10 (RET)
```

Related Items:

BC, BD, BE, VC, VD, VE, and VS commands
[Breakpoints] window
[Break] dialog box

5.2.9 BREAKPOINT_CLEAR: BC

Description:

Cancels software breakpoints that have been set.

Format:

```
bc [<address> [<address_space> [<asidopt>]]]
```

<address_space> = space <space>
<asidopt> = asid <asid>

Table 5.10 BREAKPOINT_CLEAR Command Parameters

Parameter	Type	Description
<address>	Numerical value	Breakpoint address
<space>	Keyword	Address area of a breakpoint Set either of the following: **physical**: physical address **virtual**: virtual address
<asid>	Numerical value	ASID value of a breakpoint when virtual is set to the <space> parameter.

Notes: 1. When no parameters are set, all software breakpoints are canceled.
2. When <address_space> and <asidopt> are not set, all software breakpoints that match the specified address are canceled.

Examples:

To cancel all breakpoints:

```
>bc (RET)
```

To cancel all software breakpoints whose address value is H'1000000:

```
>bc H'1000000 (RET)
```

To cancel a software breakpoint whose virtual address is H'1000000, according to the ASID value at command input:

```
>bc H'1000000 space virtual (RET)
```

To cancel the software breakpoint at virtual address H’1000000, whose ASID value is H'10:

```
>bc H'1000000 space virtual asid H'10 (RET)
```

Related Items:

BP, BD, BE, VC, VD, VE, and VS commands
[Breakpoints] window
[Break] dialog box

5.2.10 BREAKPOINT_DISPLAY: BD

Description:

Displays software breakpoints that have been set.

Format:

```
bd
```

Table 5.11 BREAKPOINT_DISPLAY Command Parameter

Parameter	Type	Description
None		

Example:

To display the software breakpoints that have been set:

>***bd (RET)***

```
  H'00000110 physical enable
  H'0000011c virtual asid H'0 disable
  H'00000250 physical enable
```

Related Items:

BP, BC, and BE commands
[Breakpoints] window
[Break] dialog box

5.2.11 BREAKPOINT_ENABLE: BE

Description:

Enables or disables software breakpoints that have been set.

Format:

```
be <address> <address_space> <asidopt> <mode>
```

<address_space> = space <space>
<asidopt> = asid <asid>

Table 5.12 BREAKPOINT_ENABLE Command Parameters

Parameter	Type	Description
<address>	Numerical value	Breakpoint address
<space>	Keyword	Address area Set either of the following: **physical**: physical address **virtual**: virtual address
<asid>	Numerical value	ASID value of a breakpoint when virtual is set to the <space> parameter.
<mode>	Keyword	Enables or disables breakpoints. Set either of the following: **enable**: Enables breakpoints. **disable**: Disables breakpoints.

Examples:

To enable a software breakpoint at physical address H'1002:

```
>be H'1002 space physical enable (RET)
```

To enable a software breakpoint at logical address H'1000000, whose ASID value is H'10:

```
>be H'1000000 space virtual asid H'10 enable (RET)
```

Related Items:

BC, BD, BP, VC, VD, VE, and VS commands
[Breakpoints] window
[Break] dialog box

5.2.12 DEVICE_TYPE: DE

Description:

Displays the currently selected MCU.

Format:

```
de
```

Table 5.13 DEVICE_TYPE Command Parameter

Parameter	Type	Description
None		

Example:

To display the currently selected MCU:

```
>de (RET)
 Current device = SHxxxx
```

5.2.13 GO_OPTION: GP

Description:

Displays or sets the emulation mode.

Format:

Displays emulation mode.

```
gp
```

Sets emulation mode.

```
gp <eml_opt>
```
<eml_opt> = eml_mode <eml_mode>

Table 5.14 GO_OPTION Command Parameter

Parameter	Type	Description
<eml_mode>	Keyword	Specifies the emulation mode. **normal**: Normal execution **sequence1**: Stops the user program only when the conditions are satisfied in the order of hardware breakpoints 2 to 1. Hardware breakpoints 1 and 2 must be set. **no_break**: Makes software breakpoints and hardware breakpoints temporarily invalid and executes the user program.

Examples:

To display the currently set emulation mode for user program execution:

```
>gp (RET)

Emulator execution mode = Sequential break Condition 2-1
```

To set the normal emulation mode for user program execution:

```
>gp eml_mode normal (RET)
```

Related Items:

BCS and BS commands,
[Breakpoints] window,
[Break], [Break Condition], and [Configuration] dialog boxes

5.2.14 JTAG_CLOCK: JCK

Description:

Displays or sets the JTAG clock (TCK) frequency.

Format:

Displays the JTAG clock (TCK) frequency.

```
jck
```

Sets the JTAG clock (TCK).

```
jck <jck_opt>
```

Table 5.15 JTAG_CLOCK Command Parameter

Parameter	Type	Description
<jck_opt>	Numerical value	Sets one of the JTAG clock (TCK) frequency. (PCMCIA used: 3.75 MHz, 7.5 MHz, or 15 MHz) **3**: 3.75 MHz **7**: 7.5 MHz **15**: 15 MHz (PCI used: 4.125 MHz, 8.25 MHz, or 16.5 MHz) **4**: 4.125 MHz **8**: 8.25 MHz **16**: 16.5 MHz

Note: The range of frequencies that the Hitachi-UDI operates at is different according to the MCUs used. For details, refer to section 6.4.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK).

Examples:

(when PCMCIA used):

To set the JTAG clock (TCK) frequency:

```
>jck 15 (RET)

JTAG Clock   15MHz
```

To display the JTAG clock (TCK) frequency:

```
>jck (RET)

JTAG Clock   15MHz
```

(when PCI used):

To set the JTAG clock (TCK) frequency:

```
>jck 16 (RET)

JTAG Clock   16.5MHz
```

To display the JTAG clock (TCK) frequency:

```
>jck (RET)

JTAG Clock   16.5MHz
```

5.2.15 MEMORYAREA_SET: MAS

Description:

Displays and sets memory area at command input, such as load, verify, save, memory display, or memory change.

Format:

Displays memory area.

```
mas
```

Sets memory area.

```
mas <memory_area> [<asidopt>]
```

<asidopt> = **asid** <asid>

Table 5.16 MEMORYAREA_SET Command Parameters

Parameter	Type	Description
<memory_area>	Keyword	Sets memory area. **normal**: Does not set memory area. **physical**: Sets physical address area. **virtual**: Sets virtual address area.
<asid>	Numerical value	Sets an ASID value from 1 to H'FF when virtual is set to the <memory_area> parameter.

Notes: 1. When virtual is set and <asid> is omitted in <memory_area>, a virtual address corresponding to the ASID value at command input is accessed.

2. When a memory is accessed, the contents in the instruction cache are disabled after this command is executed.

Examples:

To display a memory area for command input, such as load, verify, save, memory display, and memory change:

```
>mas (RET)
memoryarea_set virtual asid H'10
```

To set a memory area for command input, such as load, verify, save, memory display, and memory change, to a physical address area:

```
>mas physical (RET)
```

To set a memory area for command input, such as load, verify, save, memory display, and memory change, to a virtual address area whose ASID value is H'10:

```
>mas virtual asid H'10 (RET)
```

5.2.16 REFRESH: RF

Description:

Updates the HDI memory information.

Format:

```
rf
```

Table 5.17 REFRESH Command Parameter

Parameter	Type	Description
None		

Example:

To update the HDI memory information:

```
        >rf (RET)
```

5.2.17 RESTART: RST

Description:

Restarts the emulator. The settings of breakpoints or trace acquisition conditions are not reset here.

Format:

```
rst
```

Table 5.18 RESTART Command Parameter

Parameter	Type	Description
None		

Example:

To restart the emulator:

```
        >rst (RET)
```

5.2.18 STATUS: STS

Description:

Displays status information of the emulator.

Format:

```
sts
```

Table 5.19 STATUS Command Parameter

Parameter	Type	Description
None		

Example:

To display status information of the emulator:

```
>sts (RET)
Emulator Status
Connected to:          SHxxxx E10A Emulator big endian (E10A PCI
                        Card Driver2)
CPU                    SHxxxx
Run status             Break
Cause of last break    BREAK POINT
Run time count         0H:0M:0S:10MS
Emulator mode          Normal
Big endian
AUD                    Exist
```

5.2.19 TRACE_DISPLAY: TD

Description:

Displays the acquired trace information. The information to be acquired is the branch source and branch destination addresses when a branch is made during the user program execution.

Format:

```
td
```

Table 5.20 TRACE_DISPLAY Command Parameter

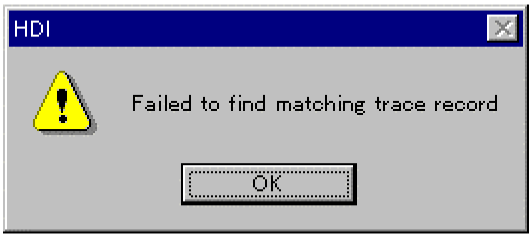
Parameter	Type	Description
None		

Notes

In some cases, the emulator address may be acquired. In such a case, the following message will be displayed at the place where the mnemonic or operand is displayed. Ignore this address because it is not a user program address.

*** EML ***

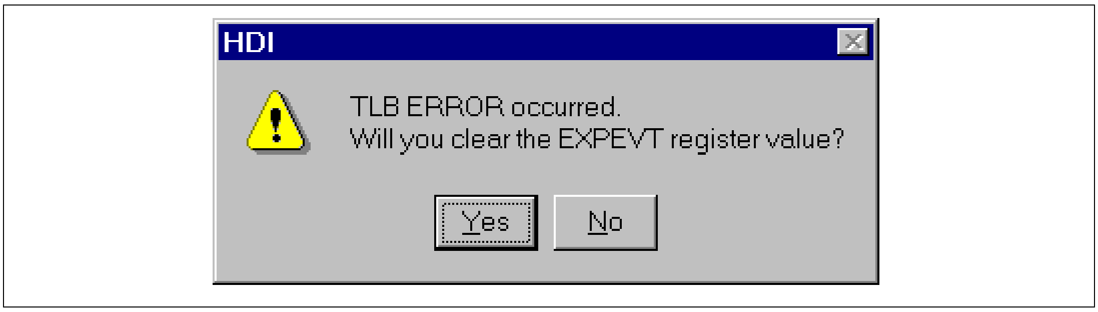
If no branch trace information is acquired, `Failed to find matching trace record` will be displayed.



Failed to find matching trace record

Figure 5.1 Failed to find matching trace record Message Box

If a TLB error occurs while acquired trace information is displayed, the following error message will be displayed:


HDI

TLB ERROR occurred.
Will you clear the EXPEVT register value?

Figure 5.2 TLB Error Message Dialog

Example:

To display trace information according to information acquired during user program execution:

```
>td (RET)
IP            TYPE          ADDR          MNEMONIC      OPERAND
-D'xxxxxx     BRANCH        01000010      JSR           @R0
              DESTINATION   01000020      MOV.L         R1, @R1
(a)           (b)           (c)           (d)           (e)
```

(a) Instruction pointer (signed decimal)
(b) Types of branch source or branch destination
BRANCH: Branch source
DESTINATION: Branch destination
(c) Address of instruction word
(d) Instruction mnemonic
(e) Instruction operand

Related Items:

BCS and TM commands
[Trace] window
[Break], [Break Condition], and [Trace Acquisition] dialog boxes

5.2.20 UBC_MODE:UM

Description:

Sets or displays the current UBC state.

Format:

```
um [<ubc_mode>]
```

Table 5.21 UBC_MODE Command Parameter

Parameter	Type	Description
<ubc_mode>	Keyword	Selects the UBC mode. **EML**: Used as Break Condition by the emulator. **USER**: Releases the UBC to the user. (Break Condition cannot be used.)

Note: When <option> is omitted, the current setting conditions are displayed.

Examples:

To release the UBC to the user:

```
    >UBC_mode user (RET)
    UBC_mode = USER
```

To display the current UBC state:

```
    >UBC_mode (RET)
    UBC_mode = EML
```

Related Item:

[Configuration] dialog box

5.2.21 VPMAP_CLEAR: VC

Description:

Clears the address translation (VP_MAP) table that is set in the emulator.

Format:

```
vc [<address>]
```

Table 5.22 VPMAP_CLEAR Command Parameter

Parameter	Type	Description
<address>	Numerical value	Sets the virtual start address of the VP_MAP table range to be cleared.

Note: All contents in the VP_MAP table are cleared if <address> is omitted.

Examples:

To clear all the contents in the VP_MAP table:

```
        >vc (RET)
```

To clear the contents in the VP_MAP table range starting from virtual address H'4000:

```
        >vc H'4000 (RET)
```

Related Items:

VD, VE, and VS commands

5.2.22 VPMAP_DISPLAY: VD

Description:

Displays the address translation (VP_MAP) table set in the emulator.

Format:

```
vd
```

Table 5.23 VPMAP_DISPLAY Command Parameter

Parameter	Type	Description
None		

Example:

To display the VP_MAP table:

```
>vd (RET)
<VADDR_TOP>         <VADDR_END>         <PADDR_TOP>
01000000            0100ffff            02000000
01010000            0101ffff            03000000
ENABLE
```

<VADDR_TOP>, <VADDR_END>, and <PADDR_TOP> represent the virtual start address, the virtual end address, and the physical start address, respectively. ENABLE or DISABLE in the last line indicates that the VP_MAP table is valid or invalid.

Related Items:

VC, VE, and VS commands

5.2.23 VPMAP_ENABLE: VE

Description:

Enables or disables the setting of the address translation (VP_MAP) table in the emulator.

Format:

```
ve [<enable>]
```

Table 5.24 VPMAP_ENABLE Command Parameter

Parameter	Type	Description
<enable>	Keyword	Enables or disables the setting of the VP_MAP table. **enable**: Enables the setting of the VP_MAP table. **disable**: Disables the setting of the VP_MAP table.

Note: The setting of the VP_MAP table is disabled at the emulator initiation.

Example:

To enable the setting of the VP_MAP table:

```
        >ve enable (RET)
```

Related Items:

VC, VD, and VS commands

5.2.24 VPMAP_SET: VS

Description:

Sets the address translation (VP_MAP) table in the emulator.

Format:

```
vs <lsaddress> <leaddress> <paddress>
```

Table 5.25 VPMAP_SET Command Parameters

Parameter	Type	Description
<lsaddress>	Numerical value	Specifies the virtual start address to be set in the VP_MAP table in the MCU page size units (1 kbyte or 4 kbytes). Setting a physical fixed area or an internal I/O area as a virtual address will result in an error.
<leaddress>	Numerical value	Specifies the virtual end address to be set in the VP_MAP table in the MCU page size units (1 kbyte or 4 kbytes). Setting a physical fixed area or an internal I/O area as a virtual address will result in an error.
<paddress>	Numerical value	Specifies the physical start address to be set in the VP_MAP table.

Note: The virtual address range to be newly set cannot overlap a virtual address that has already been set. Cancel the previous set range when making a new setting.

Example:

To set the virtual address range H'4000 to H'4FFF to be translated into the physical address range H'400000 to H'400FFF:

```
>vs H'4000 H'4fff H'400000 (RET)
```

Related Items:

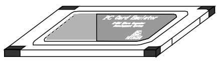
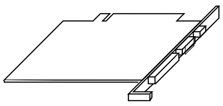
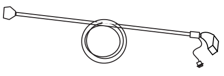
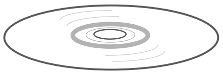
VC, VD, and VE commands

Section 6 SH7612 E10A Emulator Specifications

6.1 Components of the Emulator

The SH7612 E10A emulator supports the SH7410 and SH7612. Table 6.1 lists the components of the emulator.

Table 6.1 Components of the Emulator (HS7612KCM01H or HS7612KCI01H)

Classi-fication	Component	Appearance	Quan-tity	Remarks
Hard-ware	Card emulator (MODEL name: HS0005KCM03H or HS0005KCI03H)	 or 	1	HS0005KCM03H (PCMCIA: 14-pin type): Depth: 85.6 mm, Width: 54.0 mm, Height: 5.0 mm, Weight: 27.0 g HS0005KCI03H (PCI: 14-pin type): Depth: 144.0 mm, Width: 105.0 mm, Weight: 93.0 g
	User system interface cable		1	HS0005KCM03H (PCMCIA: 14-pin type): Length: 50 cm, Weight: 33.0 g HS0005KCI03H (PCI: 14-pin type): Length: 150 cm, Weight: 86.0 g
Soft-ware	SH7612 E10A emulator setup program, SH7612 E10A Emulator User's Manual, and Hitachi Debugging Interface User's Manual		1	HS7612KCM01SR, HS7612KCM01HJ (B), HS7612KCM01HE (B), HS6400DIIW2SJ, and HS6400DIIW2SE (provided on a CD-R)

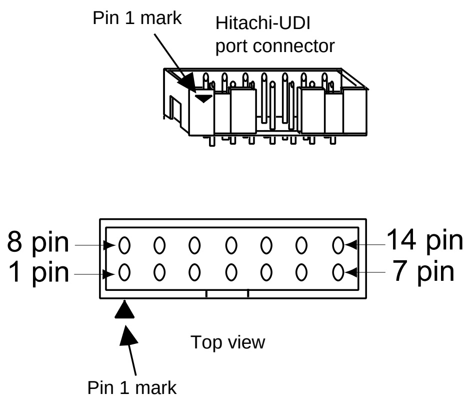
Note: The SH7612 E10A emulator does not operate on the actual chip. The following debugging chips are available:
When the SH7410 is used, ask Hitachi's sales department.
When the SH7612 is used, ask Hitachi's sales department. Note that this chip is applied only for the FP-176C (LQFP-176) product.

6.2 Pin Arrangement of the Hitachi-UDI Port Connector

Figure 6.1 shows the pin arrangement of the Hitachi-UDI port connector (14 pins).

CAUTION

Note that the pin number assignment of the Hitachi-UDI differs from that of the connector manufacturer.



Pin No.	Signal	Input/Output*[1] SH7410	Input/Output*[1] SH7612	Pin No. SH7410	Pin No. SH7612
1	TCK	Input	Input	112	150
2*[2]	/TRST	Input	Input	109	149
3	TDO	Output	Output	99	156
4 and 11	Not connected	—	—	—	—
5	TMS	Input	Input	103	152
6	TDI	Input	Input	101	154
7*[2]	/RESET	Output	Output	117	2
8 to 10 and 12 to 13	GND	—	—	—	—
14*[3]	GND	Output	Output	—	—

Notes: 1. Input to or output from the user system.
2. The slash (/) means that the signal is low-active.
3. The emulator monitors the GND signal of the user system through the UCONNECT and detects whether or not the user system is connected.

Figure 6.1 Pin Arrangement of the Hitachi-UDI Port Connector (14 Pins)

Notes: 1. Handling of the TCK, TMS, TDI, TDO, and /TRST pins depends on the use conditions of the Hitachi-UDI as follows:

(a) When the user system is used by connecting the emulator, the TCK, TMS, TDI, TDO, and /TRST pins must be pulled up by a resistance of several kilo-ohms. The ASEMD0 pin must be grounded.

(b) When the user system is independently used without using the emulator and Hitachi-UDI, the TCK, TMS, TDI, TDO, and /TRST pins must be pulled up by a resistance of several kilo-ohms. The ASEMD0 pin must be pulled up by a resistance of several kilo-ohms.

2. The /RESET signal in the user system is input to pin 117 of the SH7410 and pin 2 of the SH7612. Connect this signal to the H-UDI port connector as an output from the user system.

6.3 Differences between the SH7410 and SH7612 and the Emulator

1. When the emulator system is initiated, it initializes the general registers and part of the control registers as shown in table 6.2.

Table 6.2 Register Initial Values at Emulator Power-On

Register	When Using SH7612	When Using SH7410
R0 to R14	H'00000000	H'00000000
R15 (SP)	Value of the SP in the vector address table	Value of the power-on reset vector
PC	Value of the PC in the vector address table	Value of the power-on reset vector
SR	H'000000F0	H'000000F0
GBR	H'00000000	H'00000000
VBR	H'00000000	H'00000000
MACH	H'00000000	H'00000000
MACL	H'00000000	H'00000000
PR	H'00000000	H'00000000
RS	H'00000000	H'00000000
RE	H'00000000	H'00000000
MOD	H'00000000	H'00000000
A0G, A1G	H'00000000	H'00000000
A0, A1	H'00000000	H'00000000
X0, X1	H'00000000	H'00000000
Y0, Y1	H'00000000	H'00000000
M0, M1	H'00000000	H'00000000
DSR	H'00000000	H'00000000

2. The emulator uses the Hitachi-UDI; do not access the Hitachi-UDI.

3. Low-Power Mode (Sleep and Standby)

   For low-power consumption, the SH7410 and SH7612 have sleep and standby modes.

   The sleep and standby modes are switched using the SLEEP instruction. The sleep mode can be cleared by either normal clearing or by the satisfaction of a break condition (including (BREAK) or (CTRL) + C key input). In the latter case, the user program breaks. The standby mode can be cleared with the normal clearing function or (BREAK) or (CTRL) + C key input, and after the standby mode is cleared, the user program operates correctly. Note that, however, if a command has been entered in standby mode, no commands can be used from the emulator after the standby mode is cleared.

Notes: 1. After the sleep mode is cleared by a break, execution restarts at the instruction following the SLEEP instruction.

2. If the memory is accessed or modified in sleep mode, the sleep mode is cleared and execution starts at the instruction following the SLEEP instruction.

4. RES signal

   The SH7612 RES signal is only valid during user program execution started with clicking the GO or STEP-type button. If this signal is input from the user system in command input wait state, it is not sent to the SH7612.

Note: Do not start user program execution or access the memory while control input signals (RES, WAIT, and BREQ for SH7410, and RES, WAIT, and BRLS for SH7612) are being low. A TIMEOUT error will occur.

5. Direct Memory Access Controller (DMAC)

   The DMAC operates even while the emulation is executed or while the emulator is waiting for command input. When a data transfer request is generated, the DMAC executes DMA transfer.

6. Interrupts

   While the emulator is executing the user program, any interrupt to the SH7410 and SH7612 can be used. While the emulator is waiting for command input, interrupts are not processed. However, if an edge sensitive interrupt occurs in command input wait state, the emulator holds the interrupt and executes the interrupt processing routine when the GO command is entered.

7. 16-Bit Free-Running Timer (FRT)

   The FRT operates in command input wait state as well as during emulation. Even after the user program execution started with the GO command stops at a break, the FRT continues counting. Therefore, the timer signals are valid even after user program execution has stopped. The timer register contents can be modified with the [I/O Registers] window.

8. Serial Communication Interface (SCI)

   The SCI is valid in command input wait state as well as during emulation. For example, when data is written to the transmit data register (SCTDR) in the input ready state of the SCI and then the transmit data empty bit (TDRE) of the status register (SCSSR) is cleared, the data is output to the TxD pin.

9. Memory Space

   The SH7410 has the 48-kbyte internal ROM area and the 8-kbyte internal X-RAM and Y-RAM areas as the internal memory. The 48-kbyte internal ROM area of the SH7410 debug chip installed in the emulator can be used as the rewritable RAM area. The SH7612 debug chip has the 16-kbyte internal X-RAM and Y-RAM areas as the internal memory.

Notes: 1. When accessing the reserved memory area, use the [Memory] window; do not use other windows.

**2. The watchdog timer operates only while the user program is being executed.
If the value of the frequency change register is changed during watchdog timer operation, use the window other than the [I/O Registers] window or [Memory] window.**

10. Memory Access during Emulation

    If the memory contents are referenced or modified during emulation, realtime emulation cannot be performed.

11. The emulator can rewrite the memory contents only to the RAM area. Therefore, an operation such as memory write, software break, or user program download should be performed only for the RAM area. Note that the flash memory write is not supported.

6.4 Specific Functions for the SH7612 E10A Emulator

The SH7612 E10A emulator does not support the following functions:

- AUD trace function
- Internal I/O access break function
- MMU-related functions (The SH7410 and SH7615 do not mount the MMU.)
- VPMAP command
- Virtual and Physical specification in the [Configuration] window or on the command line
- LDTLB instruction execution break function
- ASID value specification in each window

6.4.1 Emulator Driver Selection

Table 6.3 shows drivers which can be selected in the [E10A Driver Details] dialog box. The emulator has only the following two types since it does not support the AUD trace function.

Table 6.3 Type Name and Driver

Type Name	Driver
HS7612KCM01H	E10A PC Card Driver 3
HS7612KCI01H	E10A PCI Card Driver 3

6.4.2 Hardware Break Condition Specifications

Hardware Break Conditions: In the SH7612 E10A emulator, two break conditions can be set (Break Condition 1,2). Table 6.4 lists the items that can be specified for each.

Table 6.4 Hardware Break Condition Specification Items

Items	Description
Address bus condition (Address)	Breaks when the MCU address bus value matches the specified value.
Data bus condition (Data)	Breaks when the MCU data bus value matches the specified value. Byte, word, or longword can be specified as the access data size.
Read or write condition (Read or Write)	Breaks in the read or write cycle.
Access type	Breaks when the bus cycle is the specified cycle.
Count	Breaks when the conditions set are satisfied the specified number of times.

Table 6.5 lists the combinations of conditions that can be set under Break Condition 1,2.

Table 6.5 Dialog Boxes for Setting Hardware Break Conditions

	Type		
Dialog Box	Address Bus Condition (Address)	Data Bus Condition (Data)	Read or write condition (Read or Write)
[Break Condition 1] dialog box	O	O	O
[Break Condition 2] dialog box	O	X	O

Note: O: Can be set by checking the radio button in the dialog box.
X: Cannot be set in the dialog box.

Notes on Setting the [Break Condition] Dialog Box:

1. Setting of Break Condition 2 is disabled when the stop address of the GO function is specified to execute the user program or when the STEP function is used.
2. Setting of Break Condition 2 is disabled when an instruction to which a software breakpoint has been set is executed. Accordingly, do not set a software breakpoint to an instruction in which Break Condition 2 is satisfied.
3. If a PC break before execution is set to the slot instruction after a delayed branch instruction, user program execution cannot be terminated before the slot instruction execution; execution stops before the branch destination instruction.
4. The Break Condition 1,2 settings are implemented by the MCU's user break controller. Therefore, select the EML in the [UBC mode] combo box in the [Configuration] dialog box so that the user break controller is used by the emulator.

5. Settings of BREAKPOINT and Break Condition 1,2 are invalid while the STEP OVER function is being used.

6.4.3 Notes on Setting the [Breakpoint] Dialog Box

1. When an odd address is set, the address is rounded down to an even address.
2. A software break is accomplished by replacing instructions. Accordingly, it can be set only to the RAM area. However, a software break cannot be set to the following addresses:
   - An address whose memory content is H'0000
   - An area other than RAM
   - An area of address H’C0000000 and the followings when using the SH7410, and an area of address H’40000000 and the followings when using the SH7612
   - An instruction in which Break Condition 2 is satisfied
   - A slot instruction of a delayed branch instruction
   - A lower 16-bit address of the 32-bit DSP instruction
3. During step execution, a software breakpoint is disabled.
4. A condition set at Break Condition 2 is disabled immediately after starting execution when an instruction at a software breakpoint is executed. The GO command execution does not stop when a condition of Break Condition 2 command is satisfied immediately after starting the execution.
5. When execution resumes from the breakpoint address after the program execution stops at the breakpoint, single-step execution is performed at the address before execution resumes. Therefore, realtime operation cannot be performed.
6. When a software breakpoint is set to the slot instruction of a delayed branch instruction, the exceptions of the illegal slot instruction occur although the program does not stop. Accordingly, do not set a software breakpoint to the slot instruction of a delayed branch instruction.
7. If a software breakpoint is set at a part of the repeating instructions where the BRA instruction cannot be set, it is handled as the general illegal instruction. In addition, because of the instruction restriction in the repeating loop, a break may or may not occur. Before and after the start or end of the loop, interrupts may not be accepted.
8. Settings of BREAKPOINT and Break Condition 1,2 are invalid while the STEP OVER function is being used.

6.4.4 Notes on Using the JTAG Clock (TCK)

In the SH7410, set the JTAG clock (TCK) frequency to lower than the frequency of 1/4 of the peripheral module clock. In the SH7612, set the JTAG clock (TCK) frequency to lower than the frequency of the peripheral module clock.

6.4.5 Trace Function

The trace function in the emulator uses the branch-instruction trace function in the MCU. It displays the branch-source and branch-destination addresses, the mnemonic, operand, and trace information can be acquired in realtime.

Note: When the SH7410 is used, the trace information on the four latest branch instructions can be acquired. This includes the information when execution branches from the emulator program to the user program or from the user program to the emulator program after the user program execution has been stopped. When the SH7612 is used, the trace information on the four latest branch instructions can be acquired. This includes the information when execution branches from the emulator program to the user program. Therefore, when four or more branches occur, the four latest branch instructions are acquired; when three or less branches occur, the information on the branch from the emulator program to the user program is displayed.

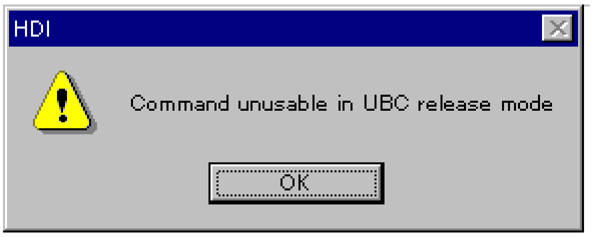
6.4.6 Notes on Setting the [Trace] Window

The emulator address may be displayed in the [Trace] window at the first address when the user program is started or at the last address when the user program is stopped. In such a case, the following message will be displayed. Ignore this address because it is not a user-program-related address.

*** EML ***

6.4.7 Notes on Setting the UBC MODE Command

In the [Configuration] window, if User is set while the UBC MODE command has been set, the STEP, STEP OVER, or STEP OUT function that uses Break Condition 2 cannot be used. If these commands are used, the message box shown in figure 6.2 is displayed.



Command unusable in UBC release mode

Figure 6.2 [HDI] Message Box

6.4.8 Notes on HDI

1. Source-Level Execution

   — Source file

   Do not move the source file from the directory path where it was compiled or assembled. Otherwise, input the old path and the replaced path in File-Load Program.
   Do not display source files that do not correspond to the load module in the program window. For a file having the same name as the source file that corresponds to the load module, only its addresses are displayed in the program window. The file cannot be operated in the program window.

   — Step

   Even standard C libraries are executed. To return to a higher-level function, enter Step Out. In a for statement or a while statement, executing a single step does not move execution to the next line. To move to the next line, execute two steps.

2. Watch

   — Local variables at optimization

   Depending on the generated object code, local variables in a C source file that is compiled with the optimization option enabled will not be displayed correctly. Check the generated object code by switching the display of the program window to Mixed.
   If the allocation area of the specified local variable does not exist, displays as follows.

   Example: The variable name is asc.

   asc = ? - target error 2010 (xxxx)

   — Variable name specification

   When a name other than a variable name, such as a symbol name or function name, is specified, no data is displayed.

   Example: The function name is main.

   main =

   — Array display

   When array elements exceed 1000, elements from after 1000 will not be displayed.

   — Modification of variable contents

   When modifying variable contents, do not select the Japanese character string as the input data type. For Japanese character string input, use Localized Dump.

3. Line Assembly
   - Input radix
     Regardless of the Radix setting, the default for line assembly input is decimal. Specify H' or 0x as the radix for a hexadecimal input.
   - Address space size
     In absolute addressing mode, specify the size (16, etc.).
4. Command Line Interface
   - Batch file
     To display the message "Not currently available" while executing a batch file, enter the sleep command. Adjust the sleep time length which differs according to the operating environment.

     Example: To display "Not currently available" during memory_fill execution:

     sleep d'3000

     memory_fill 0 ffff 0
   - Overwrite file
     In Command Line Interface, a file having the same name as the output file is overwritten without asking the user.
5. Note on Initiating HDI
   When the emulator is initiated by using another card after it has been initiated by using the PCI card, delete the [TARGET] line from the C:\windows\HDI.INI file.
6. Usage after Another HDI
   If another HDI has been used and Load last session is checked on startup of Setup-Options, the following error message is displayed when this HDI is started:

   ```
   invalid target system: <recently used debug platform name>
   ```

   In this case, initiate this HDI with "Start Menu-Run" as follows and without using the session files.

   ```
   <Directory path name in which HDI is installed>\hdi /n (RET)
   ```

   /n initiates the HDI without loading the recently used session files.
7. About Hitachi Debugging Interface User's Manual
   Does not support section 9, Selecting Functions, in Hitachi Debugging Interface User's Manual when using this version of the HDI.

8. Usage with Another HDI

— Automatic load of session files

If another HDI has been used and Load last session is selected on startup of [Options] of the [Setup] menu, the following error message is displayed when this HDI is started:

```
invalid target system: <recently used debug platform name>
```

In this case, initiate this HDI with “Start Menu-Run” as follows and without using the session files.

```
<Directory path name in which HDI is installed>\hdi /n (RET)
```

/n initiates the HDI without loading the recently used session files.

— Uninstallation of another HDI

When another HDI is uninstalled after installing this HDI, the Japanese memory-dump function cannot be used. To use this function, re-install this HDI.

9. [Select Function] dialog box

This HDI does not support software breakpoint setting in the [Select Function] dialog box (described in section 10, Selecting Functions, in the Hitachi Debugging Interface User's Manual).

10. [Performance Analysis] window

This HDI does not support the [Performance Analysis] window (described in section 13.7, Performance Analysis, in the Hitachi Debugging Interface User's Manual).

11. Loading Motorola S-type files

This HDI does not support Motorola S-type files with only the CR code (H'0D) at the end of each record. Load Motorola S-type files with the CR and LF codes (H'0D0A) at the end of each record.

12. [Memory] window

If the following memory contents are displayed, they will be incorrect.

Word access from address 2n + 1

Longword access from address 4n + 1, 4n + 2, or 4n + 3

13. [I/O Registers] window

— Display and modification

- Do not change values in the [I/O Registers] window because the emulator uses the User Break Controller.
- Do not change values of the HDI in the [I/O Registers] window. If changed, emulator will not operate correctly.
- For each Watchdog Timer register, there are two registers to be separately used for write and read.

Table 6.6 Watchdog Timer Register

Register Name	Usage	Register
WTCSR(W)	Write	Watchdog timer control/status register
WTCNT(W)	Write	Watchdog timer counter
WTCSR(R)	Read	Watchdog timer control/status register
WTCNT(R)	Read	Watchdog timer counter

- The watchdog timer operates only when the user program is executed. Do not change the value of the frequency change register in the [I/O Registers] window or [Memory] window.

14. Software Break

— Session file

When the software breakpoint address set in the session file is H'0, the breakpoint will not be set.

— Breakpoint cancellation

When the contents of the software breakpoint address is modified during user program execution, the following message is displayed when the user program stops.

```
BREAKPOINT IS DELETED A=xxxxxxxx
```

If the above message is displayed, cancel all software breakpoint settings with the [Del All] or [Disable] button in the [BreakPoints] window.

— Run program

If an invalid software breakpoint address is specified as a stop address in the [Run Program] dialog box, the invalid software breakpoint will become valid after the user program has stopped at the invalid breakpoint. Up to nine breakpoints can be specified for the stop address in the [Run Program] dialog box. The user program will not break at the 10th or further breakpoints.

15. Note on Using Windows NT®

Be sure to start in the administrator mode at installation. When the installation has been completed, power off the host computer and insert the card emulator. Then, execute SETUPPC3.exe in the installed directory.

16. Note on RUN-TIME Display

The execution time of the user program displayed in the [Status] window is not a correct value since the timer in the host computer has been used.

17. Note on Memory Access

When the SH7612 is used, do not access the range of addresses H’A2000000 to H’A3FFFFFF. An address exception occurs immediately after user program execution.