To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

RENESAS

To all our customers

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

Renesas Technology Corp.

Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.

User's Manual

H8S Series
E6000 Emulator

User's Manual

Renesas Microcomputer Development Environment System

Renesas Electronics
www.renesas.com

Rev.3.0 2001.01

Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi’s or any third party’s patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party’s rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi’s sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi’s sales office for any questions regarding this document or Hitachi semiconductor products.

IMPORTANT INFORMATION

READ FIRST

- **READ this user's manual before using this E6000 emulator.**
- **KEEP the user's manual handy for future reference.**

Do not attempt to use the E6000 emulator until you fully understand its mechanism.

E6000 emulator:

Throughout this document, the term “E6000 emulator” shall be defined as the E6000 emulator, user system interface cable, PC interface board, and optional SIMM memory module produced only by Hitachi, Ltd. excluding all subsidiary products.

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

Purpose of the E6000 emulator:

This E6000 emulator is a software and hardware development tool for systems employing the Hitachi microcomputer H8S series (hereafter referred to as MCU). This E6000 emulator must only be used for the above purpose.

Improvement Policy:

Hitachi, Ltd. (including its subsidiaries, hereafter collectively referred to as Hitachi) pursues a policy of continuing improvement in design, functions, performance, and safety of the E6000 emulator. 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 E6000 emulator:

This E6000 emulator 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 E6000 emulator 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 E6000 emulator.

LIMITED WARRANTY

Hitachi warrants its E6000 emulators to be manufactured in accordance with published specifications and free from defects in material and/or workmanship. Hitachi, at its option, will repair or replace any E6000 emulators returned intact to the factory, transportation charges prepaid, which Hitachi, upon inspection, determine to be defective in material and/or workmanship. The foregoing shall constitute the sole remedy for any breach of Hitachi's warranty. See the Hitachi warranty booklet for details on the warranty period. This warranty extends only to you, the original Purchaser. It is not transferable to anyone who subsequently purchases the emulator product from you. Hitachi is not liable for any claim made by a third party or made by you for a third party.

DISCLAIMER

HITACHI MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, ORAL OR WRITTEN, EXCEPT AS PROVIDED HEREIN, INCLUDING WITHOUT LIMITATION THEREOF, WARRANTIES AS TO MARKETABILITY, MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE OR USE, OR AGAINST INFRINGEMENT OF ANY PATENT. IN NO EVENT SHALL HITACHI BE LIABLE FOR ANY DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE, OR LOSSES OR EXPENSES RESULTING FROM ANY DEFECTIVE E6000 EMULATOR, THE USE OF ANY E6000 EMULATOR, OR ITS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCEPT AS EXPRESSLY STATED OTHERWISE IN THIS WARRANTY, THIS E6000 EMULATOR IS SOLD “AS IS ”, AND YOU MUST ASSUME ALL RISK FOR THE USE AND RESULTS OBTAINED FROM THE E6000 EMULATOR.

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 E6000 emulator without Hitachi's prior written consent or any problems caused by the user system.

All Rights Reserved:

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.

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 E6000 emulator are therefore not all inclusive. Therefore, you must use the E6000 emulator safely at your own risk.

SAFETY PAGE

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.

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 E6000 emulator and user system before connecting or disconnecting any CABLES or PARTS.**
3. **Always before connecting any CABLES, make sure that pin 1 on both sides are correctly aligned.**
4. **Supply power according to the power specifications and do not apply an incorrect power voltage. Use only the provided power cable.**

About This Manual

This manual explains how to set up and use the E6000 Emulator for the H8S series microcomputers. It is the Debugging Platform User's Manual for all H8S series E6000 emulators. For detailed specifications on each E6000 emulator, refer to the supplementary information supplied with the E6000 emulator.

Section 1, Introduction, gives a rapid introduction to the system's facilities, including an overview of the main emulation features provided by the E6000 emulator and the Hitachi debugging interface (HDI) software that provides access to them.

Section 2, Setting Up, describes how to set up the E6000 emulator and prepare it for use in conjunction with the Hitachi Debugging Interface (HDI).

Section 3, Hardware, explains how to connect the E6000 emulator to an external user system.

Section 4, Tutorial, then introduces each of the E6000 emulator's main features by showing how to load and debug a simple C program. The tutorial program is supplied on disk so that you can follow the steps on your own system to learn first-hand how it operates.

Assumptions

This manual assumes that you already have a working knowledge of the procedures for running and using applications for MS-DOS® and Microsoft® Windows® operating system.

Related Manuals

- Supplementary Information
- Hitachi Debugging Interface User's Manual
- User System Interface Cable User's Manual
- PC Interface Board User's Manual (the user's manuals listed below are referred to in this user's manual)
  ISA Bus Interface Board User's Manual (HS6000EII01HE)
  PCI Bus Interface Board User's Manual (HS6000EIC01HE, HS6000EIC02HE)
  PCMCIA Interface Card User's Manual (HS6000EIP01HE)
  Description Notes on Using LAN Adapter for E6000/E8000 Emulator (HS6000ELN01HE)
- Optional Memory Board User's Manual
  1-M SIMM Memory Module User's Manual (HS6000EMS11HE)
  4-M SIMM Memory Module User's Manual (HS6000EMS12HE)

Conventions

This manual uses the following typographical conventions:

Style	Used for
`computer`	Text that you type in, or that appears on the screen.
`parameter`	A label representing the actual value you should type as part of a command.
bold	Names of menus, menu commands, buttons, dialog boxes, and windows that appear on the screen.

Trademarks

Microsoft®, MS-DOS®, Windows®, Windows® 95, Windows® 98, Windows NT® 4.0, and Windows® 2000 are registered trademarks of Microsoft Corporation in the United States and/or in other countries.

IBM is a registered trademark of International Business Machines Corporation.

The operating environment in this manual is Microsoft® Windows® 98 for English version on the IBM PC.

Contents

Figures

Tables

Section 1 Introduction

The E6000 emulator is an advanced realtime in-circuit emulator which allows programs to be developed and debugged for the H8S series microcomputers.

The E6000 emulator can either be used without a user system, for developing and debugging software, or connected via a user system interface cable to a user system, for debugging user hardware.

The E6000 emulator works with the Hitachi debugging interface (HDI) an application for Microsoft® Windows® operating system. This provides a powerful range of commands for controlling the emulator hardware, with a choice of either fully interactive or automated debugging.

1.1 Debugging Features

1.1.1 Breakpoints

The E6000 emulator provides a comprehensive range of alternative types of breakpoints, to give you the maximum flexibility in debugging applications and user system hardware.

Hardware Break Conditions: Up to 12 break conditions can be defined using the event and range channels in the complex event system (CES). For more information about the hardware break conditions, see section 1.2, Complex Event System (CES).

Program Breakpoints (PC Breakpoints): Up to 256 program breakpoints can be defined. These program breakpoints are set by replacing the user instruction by a BREAK instruction. In target ROM, only one breakpoint (on-chip break) can be set.

1.1.2 Trace

The E6000 emulator incorporates a powerful realtime trace facility which allows you to examine MCU activity in detail. The realtime trace buffer holds up to 32768 bus cycles, and it is continuously updated during execution. The buffer is configured as a rolling buffer, which can be stopped during execution and read back by the host computer without halting emulation.

The data stored in the trace buffer is displayed in both source program and assembly languages for ease of debugging. However, if trace filtering is used, only assembly language can be displayed.

The buffer can be set up to store all bus cycles or just selected cycles. This is called trace acquisition and uses the complex event system (CES) to select the parts of the program you are interested in; see section 1.2, Complex Event System (CES), for more information.

It is also possible to store all bus cycles and then just look at selected cycles. This is called trace filtering.

1.1.3 Execution Time Measurements

The E6000 emulator allows you to measure the total execution time, or to measure the time of execution between specified events in the complex event system. You can set the resolution of the timer to any of the following values:

20 ns, 125 ns, 250 ns, 500 ns, 1 µs, 2 µs, 4 µs, 8 µs, or 16 µs.

At 20 ns the maximum time that can be measured is about six hours, and at 16µs the maximum time is about 200 days.

1.1.4 Performance Analysis

The E6000 emulator provides functions for measuring the performance of a program. The performance of the specified program range can be displayed either as a histogram or in percentage form. A timer resolution of 20 ns, 40 ns, or 160 ns can be selected. In addition, the execution count of the specified program range can be measured (1 to 65535).

1.1.5 Bus Monitoring

The E6000 emulator incorporates a bus monitoring function that monitors and displays the contents of the accessed area in HDI windows without stopping the program execution. Up to eight blocks of 256 bytes can be monitored. In addition, the E6000 emulator can output trigger signals when specified addresses (four points max.) are accessed.

1.2 Complex Event System (CES)

In most practical debugging applications, the program or hardware errors that you are trying to debug occur under a certain restricted set of circumstances. For example, a hardware error may only occur after a specific area of memory has been accessed. Tracking down such problems using simple PC breakpoints can be very time consuming.

The E6000 emulator provides a very sophisticated system for giving a precise description of the conditions you want to examine, called the complex event system. This allows you to define events which depend on the state of a specified combination of the MCU signals.

The complex event system provides a unified way of controlling the trace, break, and timing functions of the E6000 emulator.

1.2.1 Event Channels

The event channels allow you to detect when a specified event has occurred. The event can be defined as a combination of one or more of the following:

- Address or address range
- Address outside range
- Data, with an optional mask
- Read or Write or either
- MCU access type (e.g., DMAC and instruction prefetch)
- MCU access area (e.g., on-chip ROM and on-chip RAM)
- A signal state on one or more of the four external probes
- A certain number of times that the event must be triggered
- Delay cycles after an event

Up to eight events can be combined into a sequence, in which each event is either activated or deactivated by the occurrence of the previous event in the sequence. For example, you can cause a break if an I/O register is written to after a specified area of RAM has been accessed.

1.2.2 Range Channels

The range channels can be set up to be triggered on a combination of one or more of the following:

- Address or address range (inside the range)
- Data, with an optional mask
- Read or Write or either
- MCU access type (e.g., DMAC and instruction prefetch)
- MCU access area (e.g., on-chip ROM and on-chip RAM)
- A signal state on one or more of the four external probes
- Delay cycles after an event

The complex event system can be used to control the following functions of the E6000 emulator:

1.2.3 Breaks

You use breaks to interrupt program execution when a specified event, or sequence of events, is activated. For example, you can set up a break to halt execution when the program reads from one address, and then writes to another address. The break can also optionally be delayed by up to 65535 bus cycles.

1.2.4 Timing

You can set up two events and then measure the execution time of the program between the activation of the first event and second event.

1.3 Hardware Features

1.3.1 Memory

The E6000 emulator provides standard emulation memory as the substitute for on-chip ROM memory and on-chip RAM memory. When a device type or device mode without an on-chip ROM or on-chip RAM is selected, the standard emulation memory is disabled. When debugging with only the E6000 emulator and the user program and data are stored in an external address space, an optional SIMM memory module must be used. The optional SIMM memory modules can be separately purchased.

The emulation memory can be mapped in 64-byte units to any number of separate memory blocks in the MCU address space. Each memory block can be specified using the **Configure Map…** function as user (Target) or emulator (SIMM memory module) and, in each case, the access can be specified as read-write, read-only, or guarded.

The definition of each type of memory is as follows:

Table 1.1 Memory Types

Memory Type	Description
On chip	Uses the MCU on-chip memory.
User	Accesses the user system memory.
Emulator	Accesses the E6000 emulator SIMM memory module.

The contents of a specified block of memory can be displayed using the **Memory…** function. The contents of memory can be modified at any time, even during program execution and the results are immediately reflected in all other appropriate windows.

Note that modifying memory contents during program execution has the following time requirements:

1. MCU on-chip ROM or RAM, or emulator SIMM memory module

   The E6000 emulator modifies the memory contents by temporarily switching the memory bus to the emulator side without stopping the user program execution. For both memory read and memory write accesses, the HDI stores a maximum of 256 bytes of memory contents in the buffer. Therefore, the emulator uses the memory bus for up to 80 µs (25 MHz, on-chip ROM)
2. MCU on-chip I/O or DTCRAM user system memory

   The E6000 emulator stops the user program execution, then modifies the memory contents. As stated above, a maximum of 256 bytes of memory contents are accessed. Therefore, the user program stops for a maximum of 2 ms (25 MHz, emulation memory).

1.3.2 Clocks

The clock can be specified as E6000 emulator internal clock or target clock. The frequencies that can be specified as the emulator internal clock depend on the MCU. For details, refer to the supplementary information supplied together with the emulator.

1.3.3 Probes

External probes 1 and 2 (EXT1 and EXT2) can be connected to the E6000 emulator, to make use of signals on the user system for break or trace. The signal for external probe 1 can be set as the condition for the event detection system depending on the low or high level. Since the signal for external probe 2 outputs high level when the trigger setting (1 to 4) condition is matched in the bus monitor function, the signal can be used for the trigger condition for such as an oscilloscope.

1.3.4 Environment Conditions

Observe the conditions listed in table 1.2 when using the E6000 emulator.

Table 1.2 Environment Conditions

Item	Specifications
Temperature	Operating: +10 to +35°C
	Storage: –10 to +50°C
Humidity	Operating: 35 to 80% RH; no condensation
	Storage: 35 to 80% RH; no condensation
Ambient gases	No corrosive gases
DC input power	Voltage: 5 V ±5%
	Power consumption: 6 A max.
User system voltage (UVcc)	Depends on the target MCU within the range 2.7 V to 5.5 V
AC input power	Voltage: 100 to 240VAC Frequency: 50 / 60 Hz Power consumption: 61 to 70VA

1.3.5 Emulator External Dimensions and Mass

Dimensions: 219 × 170 × 54 mm

Mass: 1,000 g

Section 2 Setting Up

This section explains how to:

- Set up the PC interface board (HS6000EII01H separately purchased).
- Set up the E6000 emulator.
- Install the HDI software and use it to check correct operation of the entire system.

To use another interface board, such as a PC card (PCMCIA), refer to the user's manual for that interface board.

The E6000 emulator communicates with the HDI through the PC interface board, and therefore, the PC interface board must be inserted into the host computer.

The PC interface board is a memory mapped board, and before inserting it you first need to reserve a block of memory addresses for use by the board. This ensures that other programs do not inadvertently use the PC interface hardware.

The allocated memory area must not overlap memory already allocated to other board. If attempted, the PC interface board and the E6000 emulator product will not operate correctly. At shipment, the memory area of PC interface board is allocated to the address range from H'D0000 to H'D3FFF.

When using Microsoft® Windows® 95 or Microsoft® Windows® 98 operating system, refer to section 2.2, Setting Up the PC Interface Board on Windows® 95 or Windows® 98. When using Microsoft® Windows NT® operating system, refer to section 2.3, Setting Up the PC Interface Board on Windows NT® 4.0.

Note: The PC interface board is not supported in Windows® 2000.

2.1 Package Contents

The E6000 emulator is supplied in a package containing the following components.

- E6000 emulator
- 5V and 6A E6000 emulator power supply (AC adapter)
- CD-R(HDI, User’s Manual)
- External probes 1
- External probes 2
- Hitachi Debugging Interface for E6000 Setup Guide

Before proceeding you should check that you have all the items listed above, and contact your supplier if any are missing.

2.2 Setting Up the PC Interface Board on Windows® 95 or Windows® 98

2.2.1 Setting Up the PC Interface Board

- Start Windows® 95 or Windows® 98.
- Click the **My Computer** icon with the right mouse button and select **Properties** from the pop-up menu.

The **System Properties** dialog box will be displayed.

- Double-click the **Computer** icon in the **Device Manager** panel to open the **Computer Properties** dialog box.
- Click the **Memory** in the **View Resources** panel to display the memory resources.

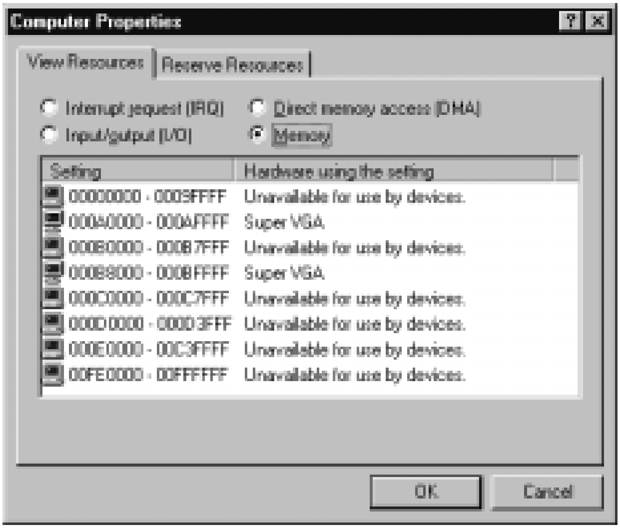


Figure 2.1 Computer Properties Dialog Box (Before Setting)

A memory area that is not listed in the dialog box can be assigned to the PC interface board. Table 2.1 lists the address ranges that can be set by the switch on the rear panel of the PC interface board. Select one of the address ranges that is not listed in the **Computer Properties** dialog box. For example, if you select the range H'D8000 to H'DBFFF, the corresponding switch number will be 6.

Table 2.1 Address Map of PC Interface Board and Memory Switch Setting

Address Range	Switch Setting
From H'C0000 to H'C3FFF	0
From H'C4000 to H'C7FFF	1
From H'C8000 to H'CBFFF	2
From H'CC000 to H'CFFFF	3
From H'D0000 to H'D3FFF (at shipment)	4
From H'D4000 to H'D7FFF	5
From H'D8000 to H'DBFFF	6
From H'DC000 to H'DFFFF	7
From H'E0000 to H'E3FFF	8
From H'E4000 to H'E7FFF	9
From H'E8000 to H'EBFFF	A
From H'EC000 to H'EFFFF	B

Define the memory area so that Windows® 95 or Windows® 98 does not use the area as follows:

- Click **Memory** in the **Reserve Resources** panel and click **Add**.

The **Edit Resource Setting** dialog box will be displayed.

Figure 2.2 Edit Resource Setting Dialog Box

- Enter the memory area addresses in **Start value** and **End value**.
- Shut down the host computer (do not restart it) and turn off the power switch.
- Using a small screwdriver, rotate the switch in the rear panel of the PC interface board so that the arrow points to the number corresponding to the memory area you have selected.
- Remove the cover from the host computer and install the PC interface board in a spare ISA slot.

- Replace the host computer cover.
- Connect the PC interface cable between the PC interface board and the PC IF connector on the E6000 emulator. Press each plug firmly home until it clicks into position.
- Switch on the host computer.
- Open the **Computer Properties** dialog box and check that the memory area you have selected is listed as System Reserved.

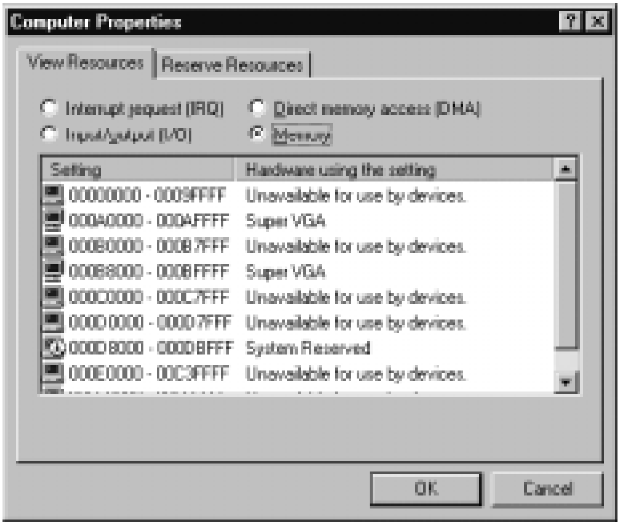


Figure 2.3 Computer Properties Dialog Box (After Setting)

2.2.2 Modifying the CONFIG.SYS File

Prevent the memory area for the PC interface board being accessed by another program as follows:

- Select **Run** from the **Start** menu.
- Type `SYSEDIT` and click **OK.**

When `EMM386.EXE` is used in the `CONFIG.SYS` file, the `CONFIG.SYS` file must be modified. If the `CONFIG.SYS` file is not used, or if `EMM386.EXE` is not used even when the `CONFIG.SYS` file is used, go to Section 2.2.3, Modifying the SYSTEM.INI File.

- Locate the line in the `CONFIG.SYS` file that reads:

```
DEVICE=C:\WINDOWS\EMM386.EXE
```

- Change the line so that it reads as shown below.

  `DEVICE=C:\WINDOWS\EMM386.EXE X=`*`aaaa–bbbb`*

  Here, *aaaa* is the upper four digits of **Start value** and *bbbb* is the upper four digits of **End value**. For example, for the switch set to 6, you would set the line to read:

  `DEVICE=C:\WINDOWS\EMM386.EXE X=D800–DBFF`

- Save the `CONFIG.SYS` file.

2.2.3 Modifying the SYSTEM.INI File

- Add the following line to the `[386enh]` section in the `SYSTEM.INI` file:

  `EMMExclude=`*`aaaa-bbbb`*

  Here, *aaaa* is the upper four digits of **Start value** and *bbbb* is the upper four digits of **End value**. For example, for the switch set to 6, you would set the line to read:

  `EMMExclude = D800-DBFF`

- Save the `SYSTEM.INI` file and exit the `SYSEDIT`.
- Restart the host computer.

  This ensures that Windows® will not use this block of memory. You are ready to connect up the E6000 emulator and run the HDI to check communication to it.

2.3 Setting Up the PC Interface Board on Windows NT® 4.0

The PC interface board uses the ISA bus slot, and therefore the host computer must have a spare ISA bus slot.
This section describes the general procedure for installing the PC interface board in the host computer. For details, refer to the manual of your host computer.

Starting Windows NT®:

- Execute **Start/Programs/Administrative Tools (Common)/WindowsNT Diagnostics**.
- Click the **Memory** button in the **Resource** tab and, in the following form, make a note of the upper memory areas that have already been used.

#	Start	End	#	Start	End	#	Start	End
0			4			8		
1			5			9		
2			6			A		
3			7			B		

- Shut down Windows NT®.

Starting the Host Computer in Setup Mode:

For details on the setup mode, refer to the manual of your host computer.

- Check which upper memory areas have already been used.

#	Start	End	#	Start	End	#	Start	End
0			4			8		
1			5			9		
2			6			A		
3			7			B		

The memory areas being used should be the same as those checked for WindowsNT® above.

- Define the memory area for the PC interface board. Select one of the memory areas that correspond to the following PC interface board switch settings, and no other devices can access the selected memory area.

#	Start	End	#	Start	End	#	Start	End
0	H'C0000	H'C3FFF	4*	H'D0000	H'D3FFF	8	H'E0000	H'E3FFF
1	H'C4000	H'C7FFF	5	H'D4000	H'D7FFF	9	H'E4000	H'E7FFF
2	H'C8000	H'CBFFF	6	H'D8000	H'DBFFF	A	H'E8000	H'EBFFF
3	H'CC000	H'CFFFF	7	H'DC000	H'DFFFF	B	H'EC000	H'EFFFF

Note: 4 is the setting at shipment.

If the **Intel P&P BIOS** disk is supplied with the host computer, define the memory area as follows:

• Start the host computer with the Intel P&P BIOS disk.

• Check the upper memory areas that have already been used, with **View/System Resources**.

• Add **Unlisted Card** with **Configure/Add Card/Others...**.

• Click **No** in the dialog box displayed because there is no .CFG file.

• Move to the **Memory [hex]** list box in the **Configure Unlisted Card** dialog box.

• Click the **Add Memory...** button to display the **Specify Memory** dialog box.

• Enter a memory area range that is not used by any other device and that corresponds to one of the PC interface board switch settings.

- Save the file.
- Exit the current setup program.
- Shut down the host computer (do not restart it) and turn off the power switch.
- Using a small screwdriver, rotate the switch in the rear panel of the PC interface board so that the arrow points to the number corresponding to the memory area you have selected.
- Remove the cover from the host computer and install the PC interface board in a spare ISA slot.
- Replace the host computer cover.
- Connect the PC interface cable between the PC interface board and the PC IF connector on the E6000 emulator. Press each plug firmly home until it clicks into position.
- Switch on the host computer.

2.4 Installing the HDI

To install the HDI, refer to Hitachi Debugging Interface for E6000 Setup Guide.

2.5 Troubleshooting

2.5.1 Faulty Connection

If the following message box appears during initialization, the PC interface board was not able to detect the E6000 emulator.

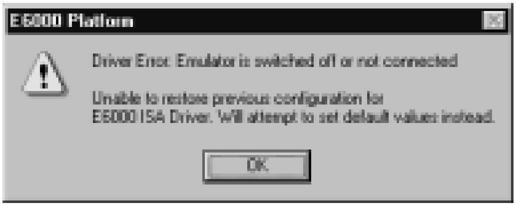


Figure 2.4 Faulty Connection Message

This indicates:

- Power supply not connected to the E6000 emulator, or the emulator not switched on. Check the power LED on the E6000 emulator.
- The PC interface cable is not correctly connected between the PC interface board and the E6000 emulator.

2.5.2 Communication Problems

The following message box indicates that the HDI was not able to set up the E6000 emulator correctly:

Figure 2.5 Communication Problem Message

This indicates:

- The memory area reserved in the `CONFIG.SYS` file does not match the inteerface switch setting on the rear panel of the PC interface board.
- Selected area of memory is in use by another application.

Section 3 Hardware

This section explains how to connect the E6000 emulator to a user system.

3.1 Connecting to the User System

To connect the E6000 emulator to a user system, proceed as follows:

- Connect the user system interface cable head to the user system.
- Plug the cable body into the E6000 emulator.
- Plug the cable body into the cable head.

For details of these steps, refer to the User System Interface Cable User's Manual.

Figure 3.1 gives details of the connectors provided on the E6000 emulator.

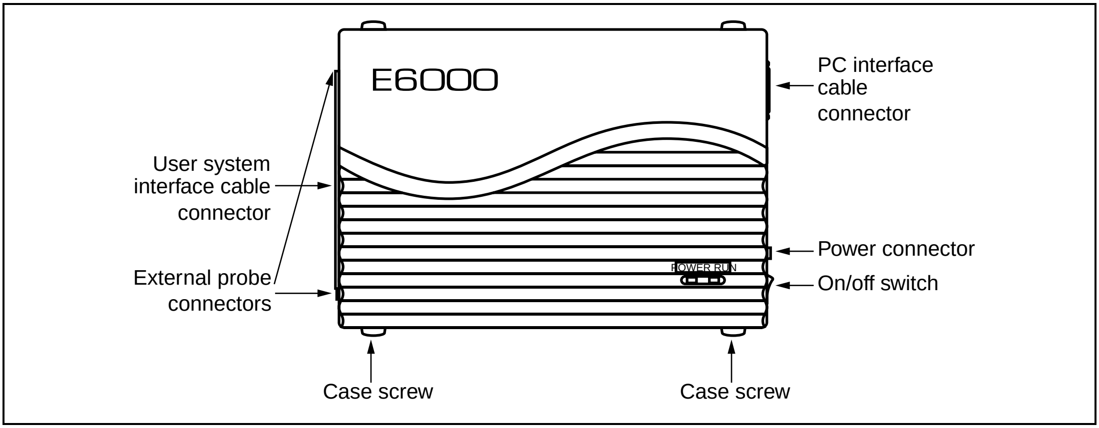


Figure 3.1 E6000 Emulator Connectors

3.1.1 Example of Connecting the User System Interface Cable Head to the User System

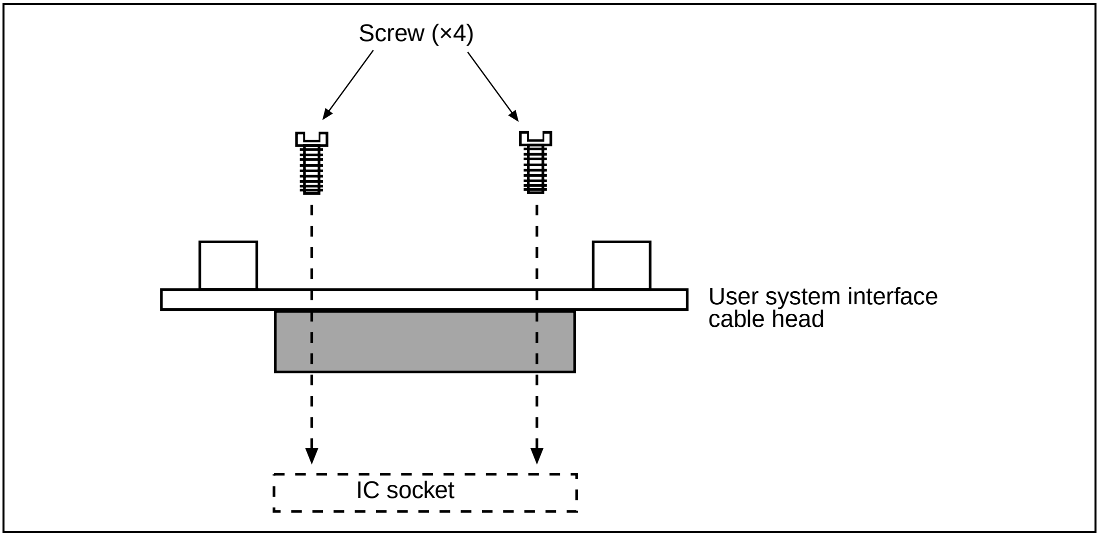


Figure 3.2 Example of Connecting User System Interface Cable Head to User System

- Ensure that all power is off to the E6000 emulator, user hardware, and associated equipment.
- Insert the cable head into the socket on the user system hardware.

Depending upon the package, it may be possible to orientate this cable head in any position on the socket, so care should be taken to correctly identify pin 1 on the E6000 emulator and socket when installing.

- Screw the cable head to the socket with the screws provided. Progressively tighten the screws in the sequence shown in figure 3.3 until all are ‘finger tight’.

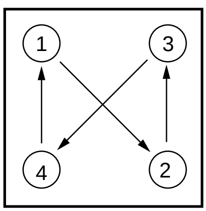


Figure 3.3 Sequence of Screw Tightening

Note: Be careful not to over-tighten the screws as this may result in contact failure on the user system hardware or damage the cable head. Where provided, use the ‘solder lugs’ on the QFP socket to provide extra strength to the E6000 emulator/user system connection.

3.1.2 Plugging the User System Interface Cable Body into the E6000 Emulator

Plug the cable body into the E6000 emulator, taking care to insert it straight, and push it firmly into place.

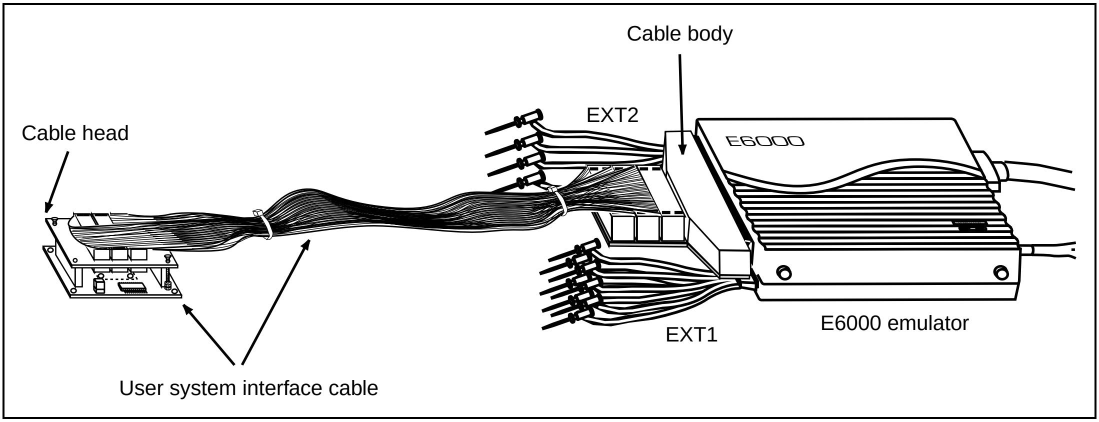


Figure 3.4 Plugging User System Interface Cable Body to E6000 Emulator

3.1.3 Plugging the User System Interface Cable Body into the Cable Head

Plug the cable body into the cable head on the user system hardware.

3.2 Power Supply

3.2.1 AC Adapter

The AC adapter supplied with the E6000 emulator must be used at all times.

3.2.2 Polarity

Figure 3.5 shows the polarity of the power-supply plug.

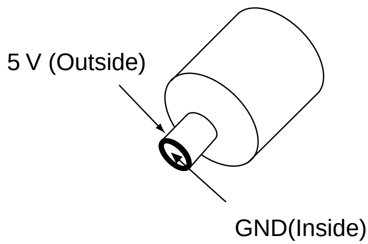


Figure 3.5 Polarity of Power Supply Plug

3.2.3 Power Supply Monitor Circuit

The E6000 emulator incorporates a power supply monitor circuit which only lights the red LED when a voltage higher than 4.75 V is supplied. If this LED does not light, you should check the E6000 emulator voltage level. An input voltage less than 4.75 V could indicate that enough current cannot be supplied to the E6000 emulator.

Note: Use the provided AC adapter for the E6000 emulator.

3.3 SIMM Memory Module

E6000 emulator optional SIMM memory modules are available which provide emulation memory for user code without needing a user system. The optional SIMM memory modules are available in different memory size, but all are partitioned into four equal banks. These banks can be relocated on page boundaries anywhere in the user area.

3.3.1 Optional SIMM Memory Module Configuration

The configuration of the optional SIMM memory module is controlled by the mapping RAM. Opening the **Memory** sheet of the **System Status** window allows you to check which optional SIMM memory module, if any, is installed and also allows the four banks to be relocated to the required addresses from the **Memory Mapping** dialog box.

3.4 Hardware Interface

All signals are directly connected to the MCU in the E6000 emulator with no buffering with the exception of those listed in the Supplementary Information:

3.4.1 Signal Protection on the E6000 Emulator

All signals are over/under voltage protected by use of diode arrays. The only exceptions being the AV_{CC} and Vref.

All ports have pull-up resistors except for analog port.

All V_{CC} pins on the cable head assembly are connected together (with the exception of the AV_{CC} pin), and are then monitored by the E6000 emulator to detect powered user system hardware presence.

3.4.2 User System Interface Circuits

The interface circuit between the MCU in the E6000 emulator and the user system has a signal delay of about 8 ns due to the user system interface cable and it includes pull-up resistors. Therefore, high-impedance signals will be pulled up to the high level. When connecting the E6000 emulator to a user system, adjust the user system hardware to compensate for propagation delays.

The following diagrams show the equivalent circuit examples of the interface signals. The interface circuits depend on the MCU type. For details, refer to the supplementary information supplied together with the E6000 emulator.

General Ports:

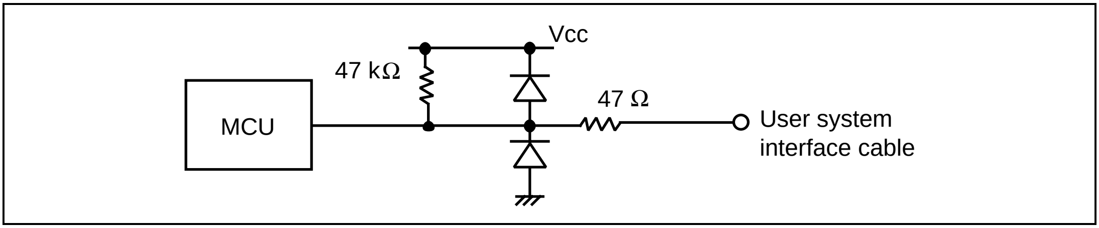


Figure 3.6   User System Interface Circuit for General Ports

Mode Pins (MD2, MD1, and MD0), WAIT, and NMI: The WAIT and NMI signals are input to the MCU through the emulator control circuit. The rising/falling time of these signals must be 8 ns/V or less. The mode pins are only monitored. The CPU mode depends on the HDI settings.

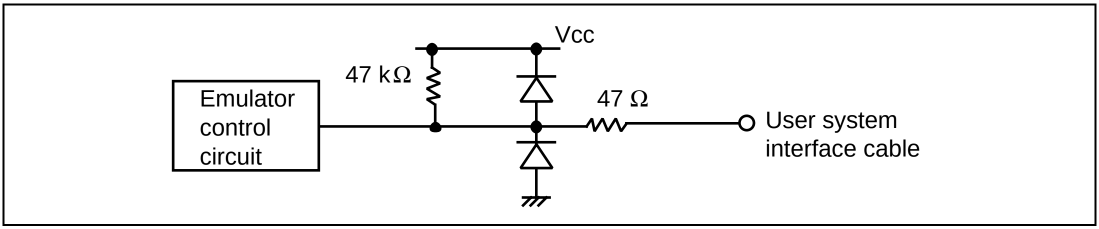


Figure 3.7   User System Interface Circuit for MD2, MD1, MD0, WAIT, and NMI

RESET:

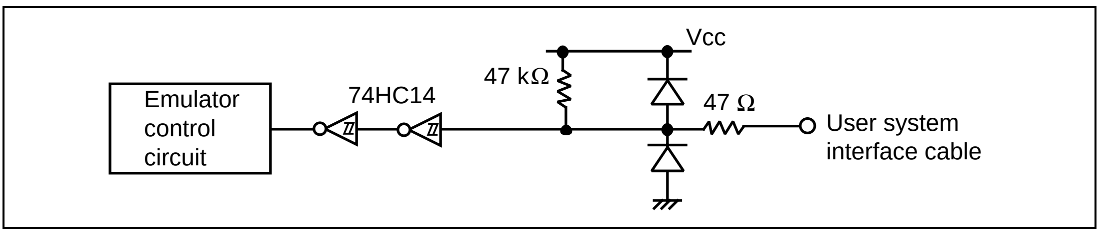


Figure 3.8   User System Interface Circuit for RESET

Analog Port Control Signals:

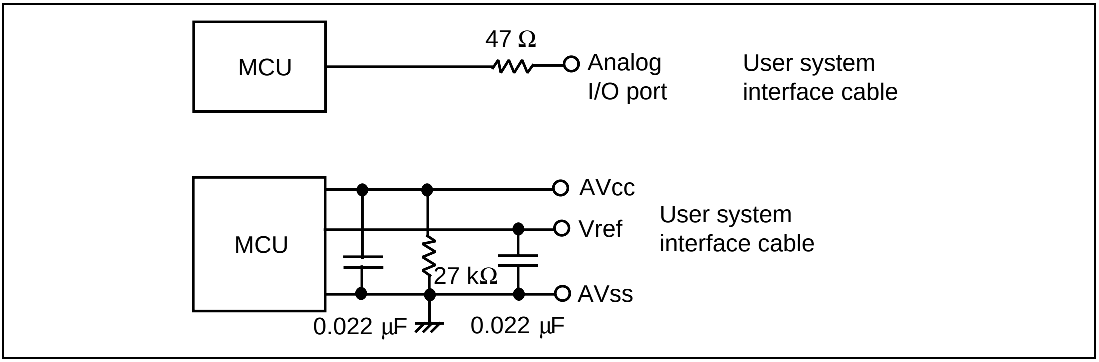


Figure 3.9 User System Interface Circuit for Analog Port Control Signals

IRQ0–IRQ7: The IRQ0 to IRQ7 signals are input to the MCU and also to the trace acquiring circuit. Therefore, the rising and falling time of these signals must be within 8 ns/V or shorter.

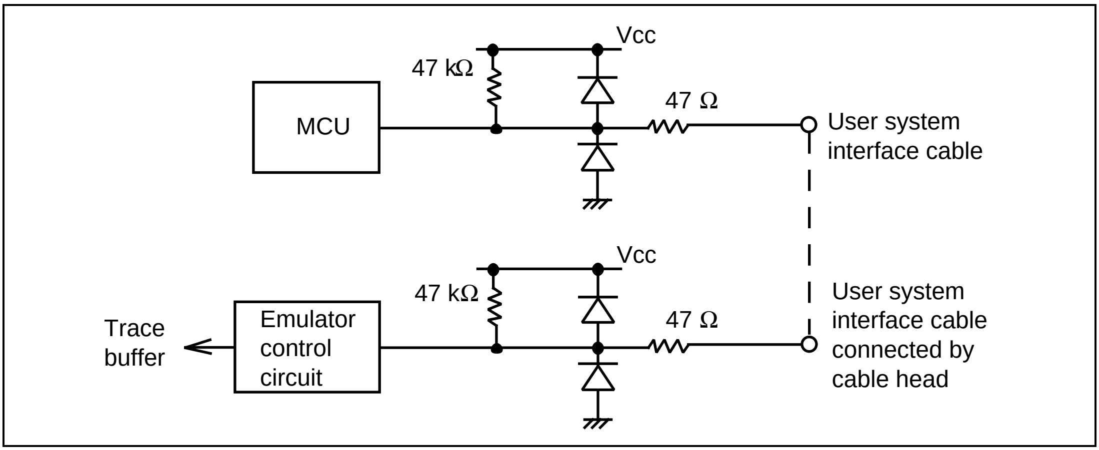


Figure 3.10 IRQ0–IRQ7 User System Interface Circuit

3.4.3 Clock Oscillator

The oscillator circuit has been implemented on the user system interface cable head. For details on the oscillator circuit, refer to the user's manual for each user system interface cable.

3.4.4 External Probe 1 (EXT1)/Trigger Output

An 8-pin connector, marked EXT1 (on the right under the user system interface cable connector), on the E6000 emulator case accommodates four external probe inputs and two trigger outputs. The pin assignment of this connector is shown in figure 3.11.

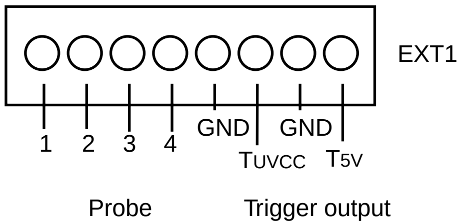


Figure 3.11 External Probe Connector 1

The interface circuit for the external probe 1 is shown in figure 3.12.

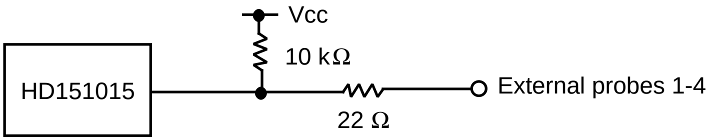


Figure 3.12 Interface Circuit for External Probe 1

The trigger output is controlled by event channel 8 and is an active low signal. The trigger output is available as either T5V (within the range from 2.5 V to 5 V; does not depend on the user V_{CC} level) or TUV_{CC} (the user V_{CC} level).

3.4.5 External Probe 2 (EXT2)/Trigger Output

A 6-pin connector, marked EXT2 (on the right under the user system interface cable connector), on the E6000 emulator case accommodates four trigger outputs. The pin assignment of this connector is shown in figure 3.13.

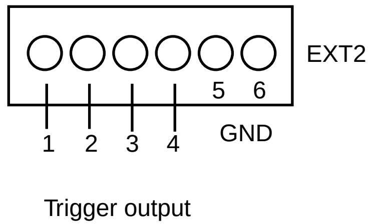


Figure 3.13 External Probe Connector 2

The trigger output is an active high signal which is output during the read or write cycles when a trace condition (1 to 4) of the bus monitor function is satisfied. The trigger output is available as user V_{CC} level.

3.4.6 Voltage Follower Circuit

CAUTION

1. Do not connect the user system interface cable to the E6000 emulator without user system connection.
2. Turn on the user system before starting up the E6000 emulator.

A voltage follower circuit is implemented on the E6000 emulator which allows the user system voltage level from the user system to be monitored. This monitored voltage level is automatically supplied to the logic on the E6000 emulator and is derived from the E6000 emulator power supply unit. This means that no power is taken from the user system board.

If no user system interface cable is connected to the E6000 emulator, the E6000 emulator will operate at a specified voltage and all clock frequencies will be available to the user. If the user system interface cable is attached, the E6000 emulator will match the voltage supplied to the user target in all cases; i.e. even when the user V_{CC} is below the operating voltage for the MCU. You must be careful not to select an invalid clock frequency. When the E6000 emulator is connected to the user system and the user system is turned off, the voltage follower circuit output voltage level is 0 V. In this case, the E6000 emulator will not operate correctly.

You can set a user V_{CC} threshold in the range Vcc max. – 0 V by using the E6000 emulator configuration dialog box. If the user V_{CC} drops below this threshold, the **User System Voltage** in the **Platform** sheet of the **System Status** window will display `Down`, otherwise `OK` is displayed.

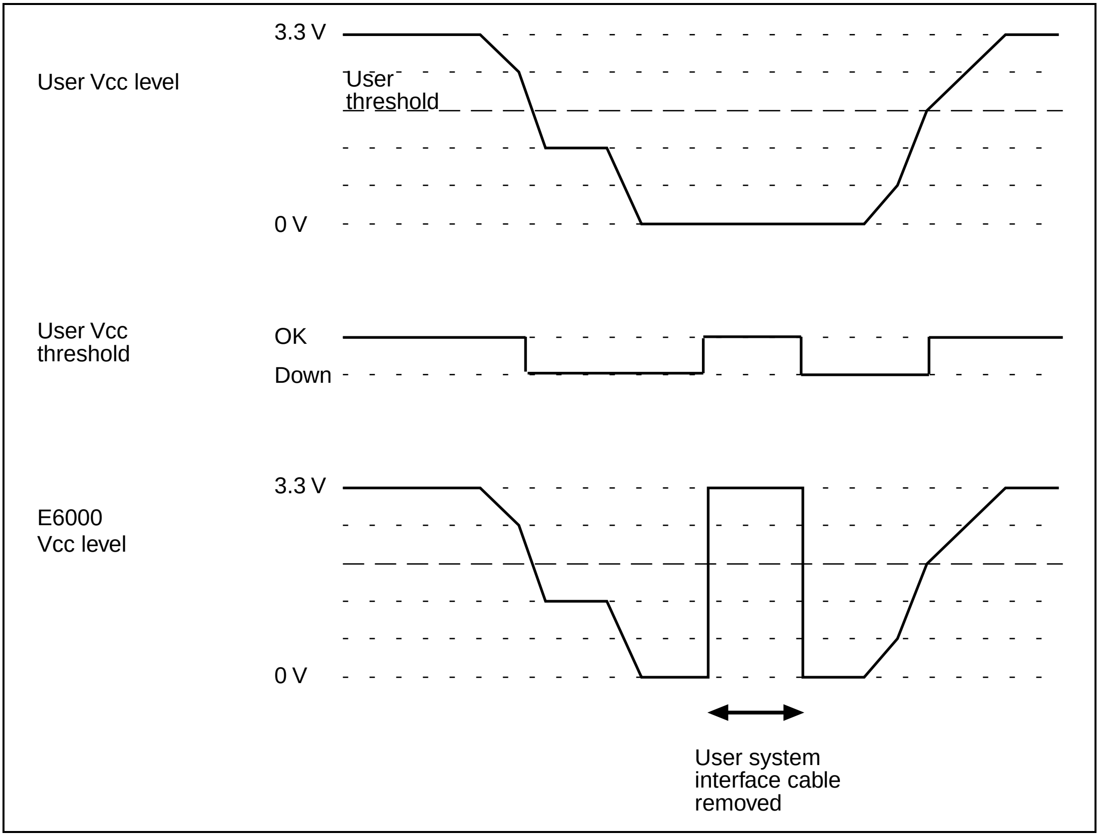


Figure 3.14 Voltage Level Monitoring (Example for Vcc = 3.3 V)

3.5 Differences between MCU and E6000 Emulator

When the E6000 emulator is initialized or the system is reset, there are some differences in the initial values in some of the general registers between the MCU and E6000 emulator as shown in table 3.1.

Table 3.1 Initial Value Differences between MCU and E6000 Emulator

Status	Register	E6000 Emulator	MCU
Power-on	PC	Reset vector value	Reset vector value
	ER0 to ER6	Undefined	Undefined
	ER7 (SP)	H'10	Undefined
	CCR	The I mask is set to 1 and the other bits are undefined	The I mask is set to 1 and the other bits are undefined
Reset command	PC	Reset vector value	Reset vector value
	ER0 to ER6	Undefined	Undefined
	ER7 (SP)	H'10	Undefined
	CCR	The I mask is set to 1 and the other bits are undefined	The I mask is set to 1 and the other bits are undefined

Please refer to the supplied supplementary information for details of the protection circuit used on the I/O ports of the E6000 emulator.

3.5.1 A/D Converter and D/A Converter

Due to the use of a user system interface cable, there is a slight degradation in the A/D and D/A conversion than that quoted in the Hardware Manual for the MCU being emulated.

Section 4 Tutorial

The following describes a sample debugging session, designed to introduce the main features of the E6000 emulator used in conjunction with the Hitachi debugging interface (HDI) software.

The tutorial is designed to run in the E6000 emulator's resident memory so that it can be used without connecting the E6000 emulator to a user system.

The tutorial assumes that the H8S/2655 E6000 is used. When using another type of E6000 emulator, change the file and directory names to your target ones.

4.1 Introduction

The tutorial is based on a simple C program.

Before reading this chapter:

- Set up the E6000 emulator from the HDI software. See section 2, Setting Up. You do not need to connect the E6000 emulator to a user system to use this tutorial.
- Make sure you are familiar with the architecture and instruction set of the MCU. For more information, refer to the Hardware Manual for the target MCU.

The tutorial program arranges ten random data in ascending/descending order. The source program (`Sort.C`), and the object file in the ELF/DWARF2 format (`Tutorial.abs`) are provided in the HDI installation disk.

4.2 Starting HDI

To start the HDI:

- Select **Hitachi Debugging Interface** from **HDI for E6000 H8S_2655** of the **Start** menu.

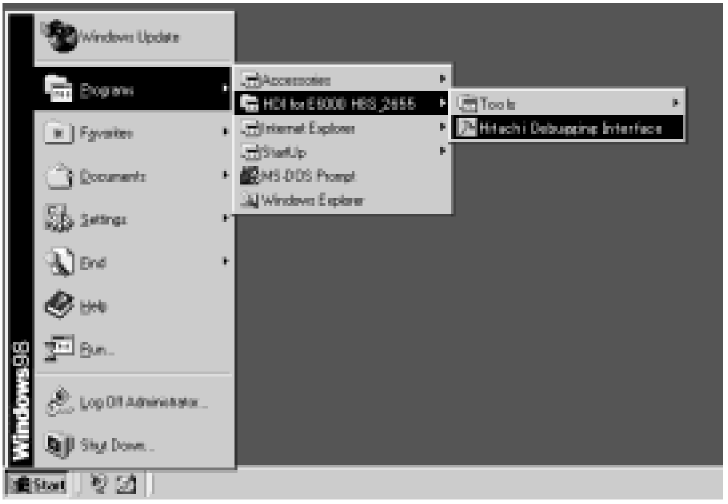


Figure 4.1 HDI Start Menu

4.2.1 Selecting the Target Platform

The HDI has extended functions for supporting multiple target platforms, and if your system is set up for more than one platform you will first be prompted to choose the target platform.

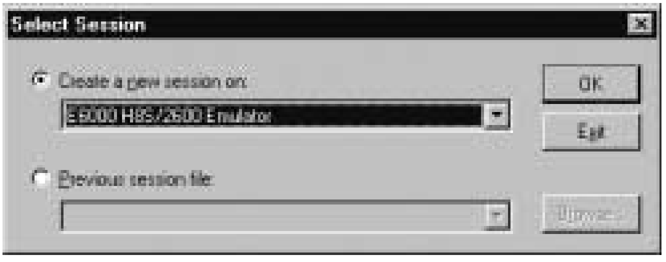


Figure 4.2 Select Platform Dialog Box

- For this tutorial select `E6000 H8S/2600 Emulator` and click **OK** to continue.

Note that you can change the target platform at any time by choosing **Select Platform…** from the **Setup** menu. If you have only one platform installed, this menu option will not be available.

When the emulator has been successfully set up, the **Hitachi Debugging Interface** window will be displayed, with the message `Link up` in the status bar.

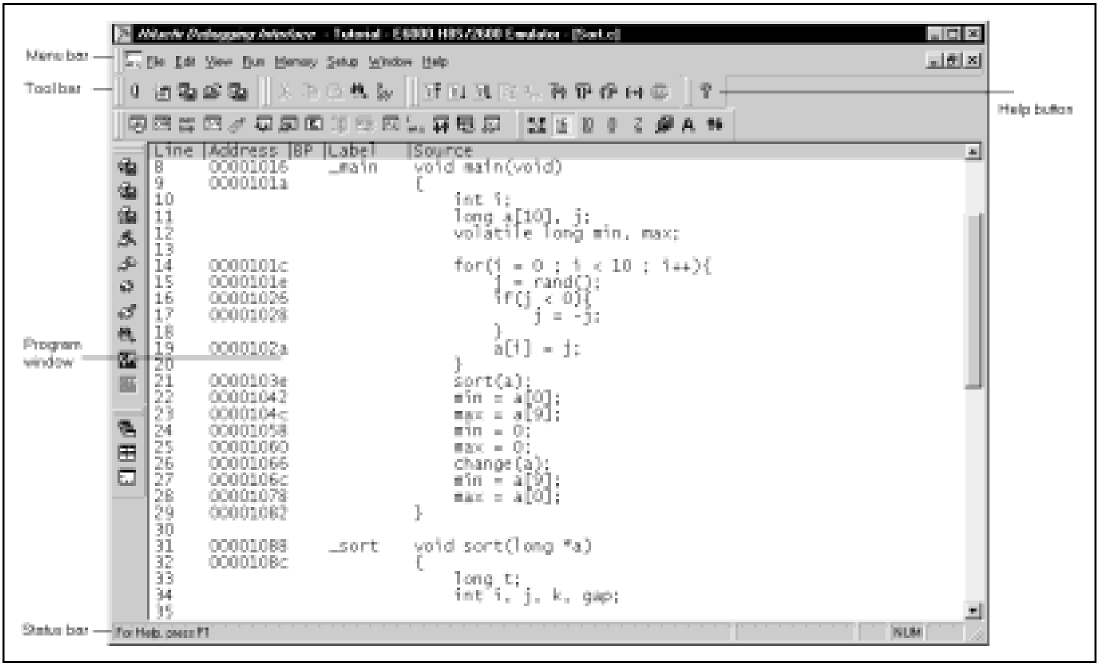


Figure 4.3 Hitachi Debugging Interface Window

For the key features of HDI, see Hitachi Debugging Interface User's Manual. For the functions specialized for the E6000 emulator, refer to the on-line help.

Menu Bar: Gives you access to the HDI commands for setting up the E6000 emulator and using the HDI debugging functions.

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

Program Window: Displays the source of the program being debugged.

Status Bar: Displays the status of the E6000 emulator. For example, progress information about downloads, snapshots of address bus in run mode.

Help Button: Activates context sensitive help about any feature of the HDI user interface.

4.3 Setting up the E6000 Emulator

Before downloading a program to the E6000 emulator, you first need to set up the target MCU conditions. The following items need to be configured:

- The device type
- The operating mode
- The clock source
- The user signals
- The memory map

The following sections describe how to set up the E6000 emulator as appropriate for the tutorial program.

4.3.1 Configuring the Platform

To set up the target configuration:

- Choose **Configure Platform…** from the **Setup** menu to set up the conditions for the selected platform.
- The following dialog box will be displayed:

Figure 4.4 Configuration Dialog Box

RENESAS

- Set up the options as shown in table 4.1.

Table 4.1 Configuration Options

Option	Value (Depending on Evaluation Chip)
Device	H8/2655
Mode	7 (advanced mode, single chip)
Clock	10 MHz
Timer resolution	125 ns
All other options	Default

- Click **OK** to change the target MCU settings.

4.3.2 Mapping the Memory

After you have selected the device and mode in the **Configuration Dialog Box**, the HDI automatically maps the E6000 emulator memory for the device and mode you have selected.

- To display the current memory mapping, choose **Configure Map...** from the **Memory** menu, or click the **Memory Map** button in the toolbar.

The **Memory Mapping** dialog box shown in figure 4.5 is displayed.

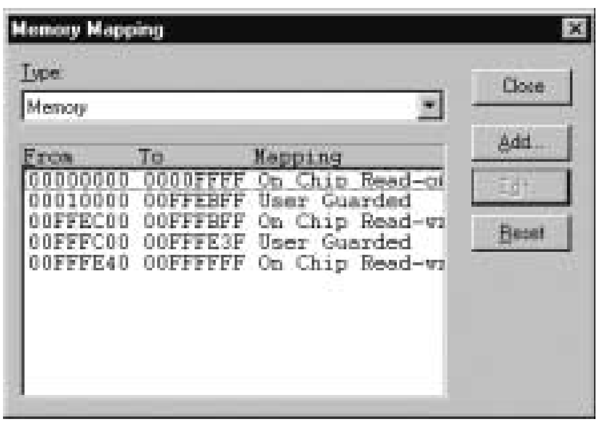


Figure 4.5 Memory Mapping Dialog Box

RENESAS

Table 4.2 lists the three memory types available in the E6000 emulator.

Table 4.2 Memory Types

Memory Type	Description
On-chip	Accesses the MCU on-chip memory.
User	Accesses the memory on the user system hardware.
Emulator	Accesses the optional SIMM memory module.

Table 4.3 lists the three access types.

Table 4.3 Access Types

Access Type	Description
Read-write	RAM
Read-only	ROM
Guarded	No access allowed

For this tutorial, we can use the default mapping, but you can edit the mapping as follows:

- To change the map setting, click the **Edit** button after selecting the target mapping line, or simply double-click that line.

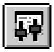
Here, double-click the `On Chip Read-only` in the **Memory Mapping** dialog box.

The **Edit Memory Mapping** dialog box is displayed.

Figure 4.6 Edit Memory Mapping Dialog Box

- Click **OK** to close the dialog box.

- To display the device map information, select **Status** from the **View** menu or click the **Status** button in the toolbar to open the **System Status** window, and select the **Memory** sheet. The device map information is then displayed as follows:



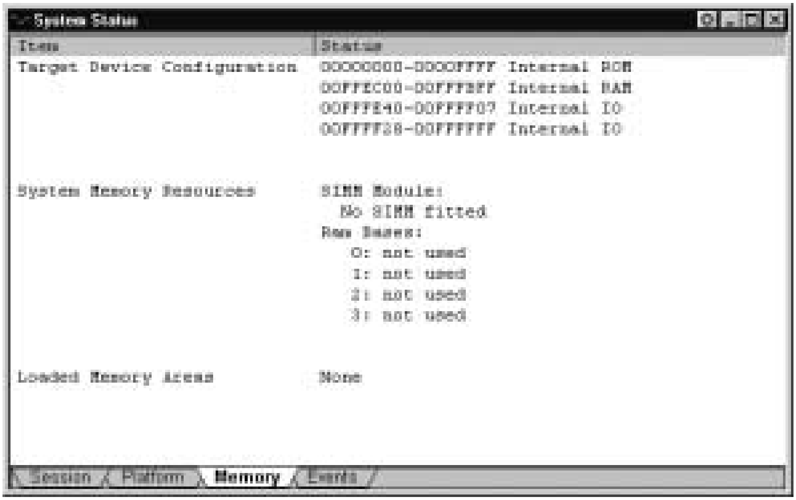


Figure 4.7 System Status Window (Memory Sheet)

Note: The memory map differs depending on the target MCU and the MCU mode.

4.4 Downloading the Tutorial Program

After the E6000 emulator is set up, you can download the object program you want to debug.

4.4.1 Loading the Object File

First load the ELF/DWARF2-format object file, as follows:

- Choose **Load Program...** from the **File** menu, or click the **Load Program** button in the toolbar. **The Load Program** dialog box is then displayed.

- Click the **Browse...** button, select the **Tutorial.abs** file in the **Tutorial** directory from the **Open** dialog box, and click the **Open** button. The **Load Program** dialog box is displayed. Click the **Open** button to start to download the file.

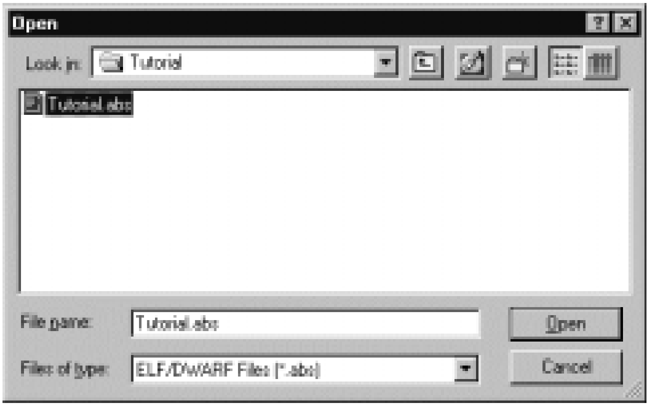


Figure 4.8 Open Dialog Box (Object File Selection)

When the file has been loaded, the dialog box shown in figure 4.9 displays information about the memory areas that have been filled with the program code.

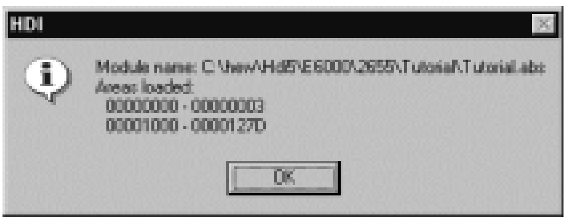


Figure 4.9 HDI Dialog Box

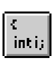
- Click **OK** to continue.

The program has been loaded into the on-chip ROM.

4.4.2 Displaying the Program Listing

The HDI allows you to debug a program at a source level.

- Choose **Source…** from the **View** menu, or click the **Program Source** button in the toolbar.



RENESAS

You will be prompted for the C source file corresponding to the object file you have loaded.

Figure 4.10 Open Dialog Box (Source File Selection)

- Select `Sort.c` and click **Open** to display the program window.

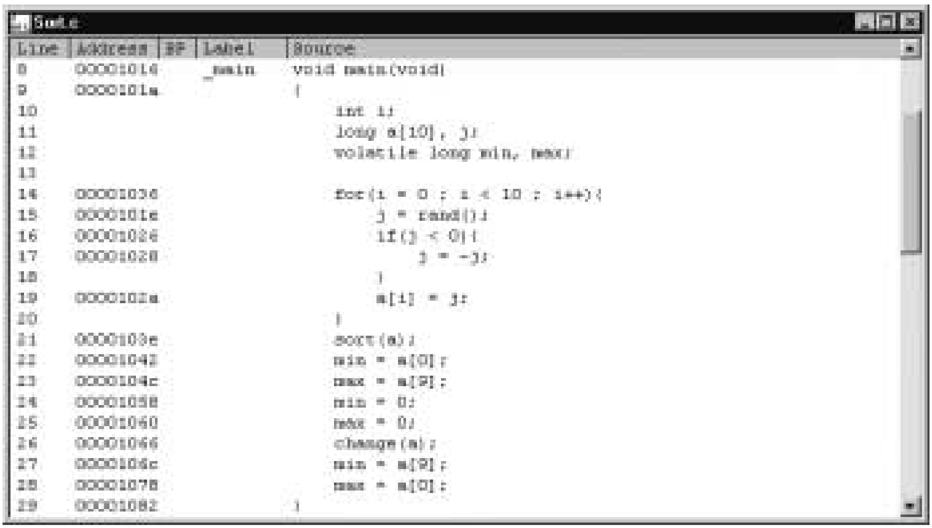


Figure 4.11 Source Program Window

- If necessary, choose **Font...** option from the **Customize** submenu on the **Setup** menu to choose a font and size suitable for your host computer.

Initially the program window opens showing the beginning of the `main` program, but you can scroll through the program with the scroll bars to see other sections.

4.5 Using Breakpoints

The simplest debugging aid is the PC break, which lets you halt execution when a particular point in the program is reached. You can then examine the state of the MCU and memory at that point in the program.

4.5.1 Setting a PC Break

The program window provides a very simple way of setting a PC break. For example, set a PC break at address H'103e as follows:

- Double-click in the **BP** column on the line containing address H'103e.

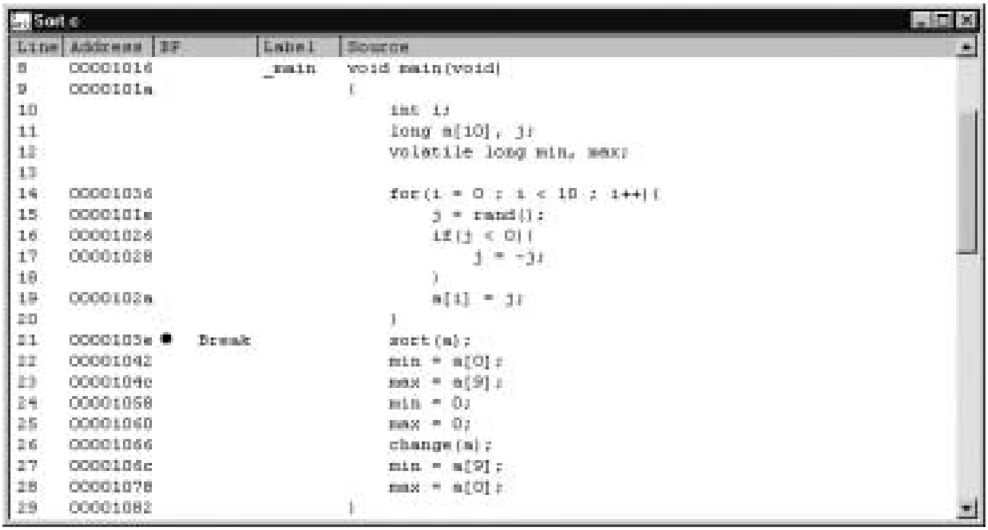


Figure 4.12 Setting a Breakpoint (PC Break)

The word • `Break` will be displayed there to show that a PC break is set at that address. Although not performed in this tutorial, double-clicking repeatedly in the **Break** column can change the display in the cyclic order shown below to set the event for measuring the execution time between events (`+Timer`: start time measurement; `-Timer`: stop time measurement), point-to-point trace (`+Trace`: start trace; `-Trace`: temporarily stop trace), or trace stop (`TrStop`: stop trace). When `-Trace` is followed by `+Trace`, trace is resumed. However, when `-Trace` is followed by `TrStop`, trace will not resume even after `+Trace` appears.

(Blank) → `Break` → `+Timer` → `-Timer` → `+Trace` → `-Trace` → `TrStop` → (Blank) → ...

Note: Events -Timer and -Trace can be selected only when the corresponding +Timer and +Trace have been set.

4.5.2 Executing the Program

To run the program from the address pointed to by the reset vector:

- Choose **Reset Go** from the **Run** menu, or click the **Reset Go** button in the toolbar.

The program will be executed up to the PC break you inserted, and the statement will be highlighted in the program window to show that the program has halted.

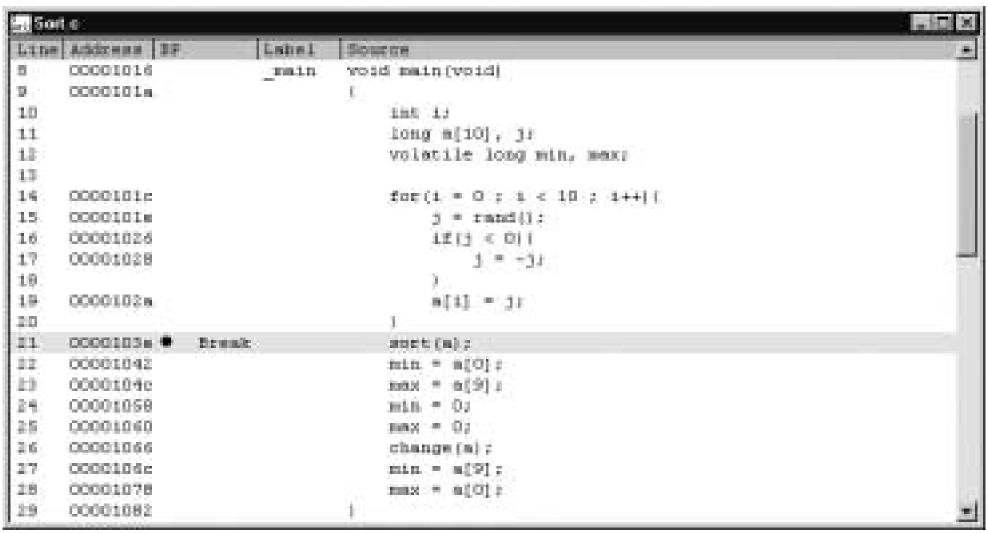


Figure 4.13 Program Break

The message `Break=PC Break` is displayed in the status bar to show the cause of the break.

You can also see the cause of the last break in the **System Status** window.

- Choose **Status** from the **View** menu or click the **Status** button in the toolbar to open the **System Status** window, and select the **Platform** sheet.

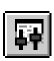


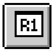
Figure 4.14 System Status Window (Platform Sheet)

The `Cause of last break` line shows that the break was a PC break. The `Run Time Count` line shows that the user program executing time (from user program start to break) is 218.375µs. The timer resolution of the event time (set by `+Timer` and `–Timer`) and the run time timer's resolution is decided by the **Timer Resolution** option in the target **Configuration** dialog box. When using a small resolution (e.g. 20 ns) for a long time measurement, the inaccuracy may be large. Select the timer resolution suitable for the length of measurement time.

4.5.3 Examining Registers

While the program is halted you can refer to the contents of the MCU registers. These are displayed in the **Registers** window.

- Choose **Registers** from the **View** menu, or click the **CPU Registers** button in the toolbar:



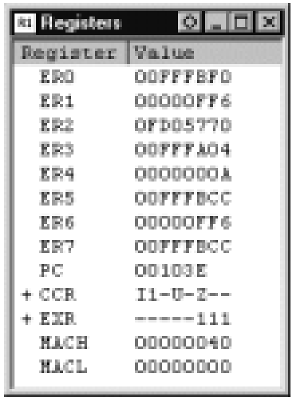


Figure 4.15 Registers Window

As expected, the value of the program counter, PC, is the same as the highlighted statement, `H'103e`.

(Note: The values of the other registers may differ from those shown in the above figure.)

You can also change the registers from the **Registers** window. For example, to change the value of the PC:

- Double-click the **Value** column corresponding to `PC` in the **Registers** window.

The **Register** dialog box allows you to edit the value.

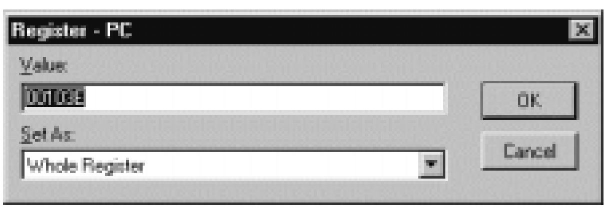


Figure 4.16 Register Dialog Box

- Edit the value to `H'1016`, the start address of the main program, and click **OK**.

The highlighted bar will move to the top of the main program to show the new PC value.

- Choose **Go** from the **Run** menu, or click the **Go** button in the toolbar, to execute up to the breakpoint again.

4.5.4 Reviewing the Breakpoints

You can see a list of all the breakpoints set in the program in the **Breakpoints** window.

- Choose **Breakpoints** from the **View** menu, or click the **Breakpoint** button in the toolbar:

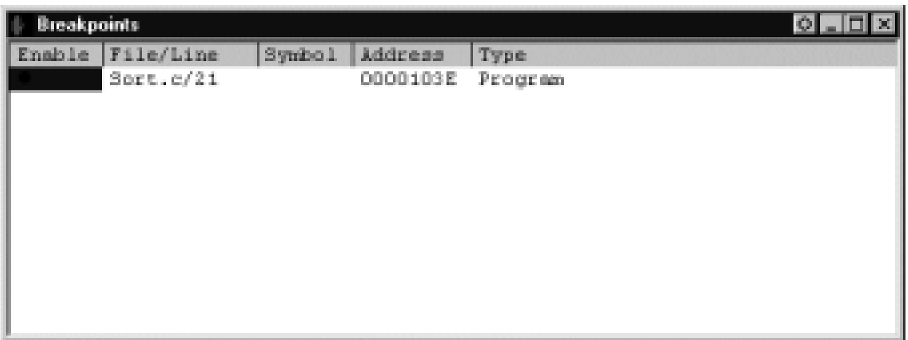


Figure 4.17 Breakpoints Window

The **Breakpoints** window also allows you to enable or disable breakpoints, define new breakpoints, and delete breakpoints.

4.6 Examining Memory and Variables

You can monitor the behavior of a program by examining the contents of an area of memory, or by displaying the values of variables used in the program.

4.6.1 Viewing Memory

You can view the contents of a block of memory in the **Memory** window.

For example, to view the memory corresponding to the array `main` in Byte:

- Choose **Memory…** from the **View** menu, or click the **Memory** button in the toolbar.

- Enter `main` in the **Address** field, and set **Format** to `Byte`.

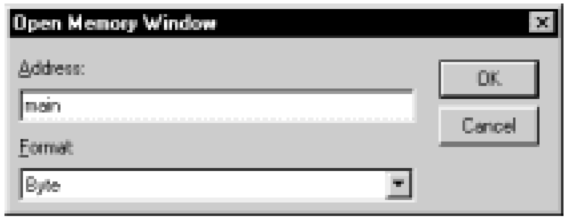


Figure 4.18 Open Memory Window Dialog Box

- Clicking **OK** opens the **Memory** window showing the specified area of memory and enables to check the contents of the memory block.

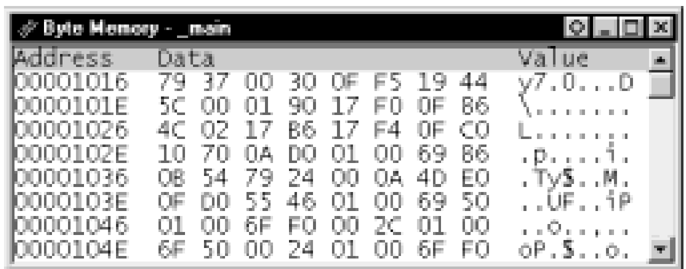


Figure 4.19 Memory Window (Byte)

4.6.2 Watching Variables

As you execute a step of a program, it is useful to be able to look at the values of variables used in your program, to verify that they change in the way that you expected.

For example, look at the `long`-type array variable `a`, declared at the beginning of the program, using the following procedure:

- Click the left of array variable `a` and move the cursor to the position in the program window.
- Click in the Program window with the right mouse button to display a pop-up menu, and choose **Add Watch**.

The **Watch** window will display the variable.

RENESAS

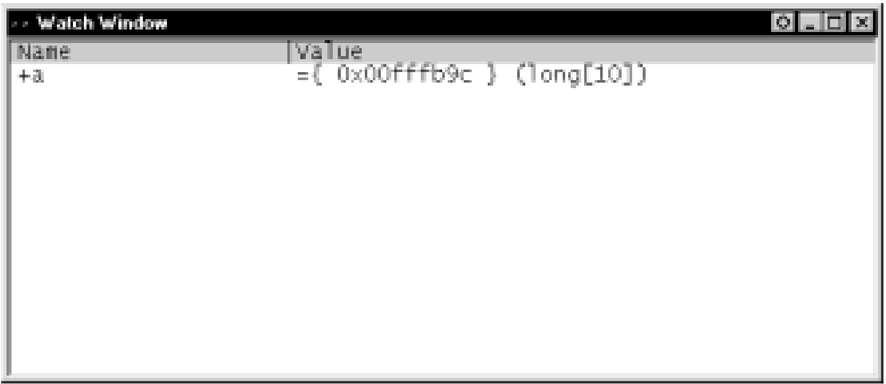


Figure 4.20 Watch Window (After Adding Variables)

You can double-click the + symbol to the left of symbol `a` in the **Watch** window to expand it and display the individual elements in the array.

If necessary, select **Decimal** from the **Radix** submenu in the **Setup** menu, or click the **Radix=Decimal** button in the toolbar to display in decimal form.

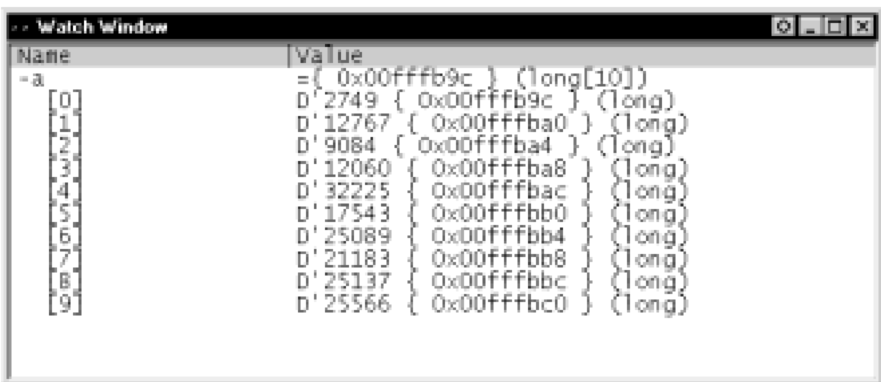


Figure 4.21 Watch Window (Symbol Expansion)

A variable name can be specified to add a variable to the **Watch** window.

- Click in the **Watch** window with the right mouse button to display a popup menu, and choose **Add Watch...**.
- Enter variable name `max` and click the **OK** button.

Figure 4.22 Add Watch Dialog Box

The `volatile long`-type variable `max` is added to the **Watch** window.

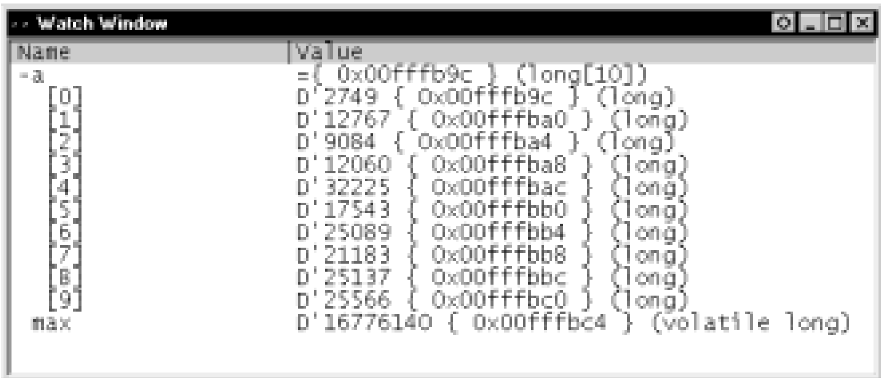


Figure 4.23 Watch Window (Adding Variables)

4.7 Stepping Through a Program

The E6000 emulator provides a range of options to perform step execution by executing an instruction or statement at a time. The alternative step commands listed in table 4.4 are provided.

Table 4.4 Step Commands

Command	Description
Step In	Executes every statement, including statements within functions.
Step Over	Executes a function call in a single step.
Step Out	Exits a function and stops at the next statement of the calling program.
Step…	Allows you to step repeatedly the specified number of times.

4.7.1 Single Stepping

- Confirm that a PC break is set at H'103e.
- Select **Reset Go** from the **Run** menu or click the **Reset Go** button in the toolbar.

The program is executed and stopped at `H'103e` by set `PC break`. The statement of `sort(a)` will be highlighted.

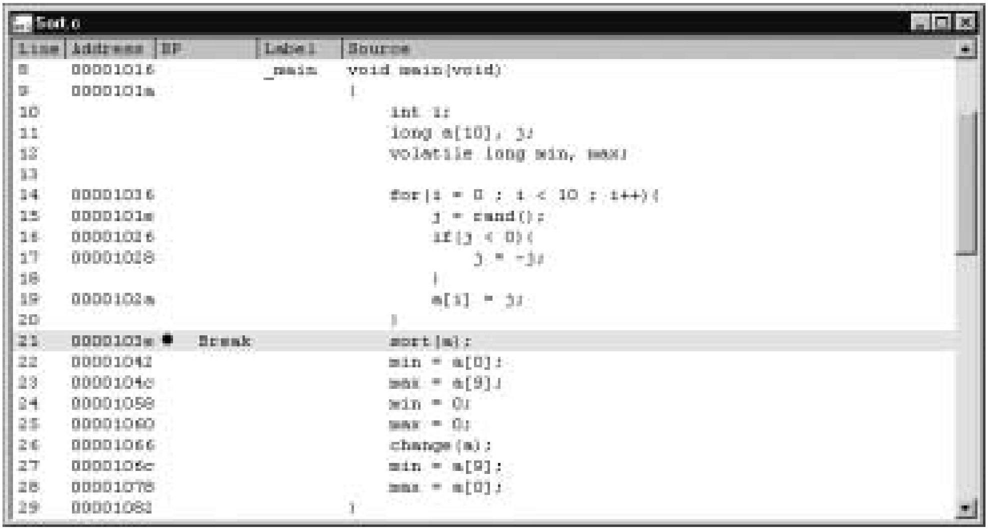
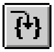


Figure 4.24 Program Window after Executing the Reset Go Command

- Choose **Step In** from the **Run** menu, or click on the **Step In** button in the toolbar, to step through the `sort` statement.



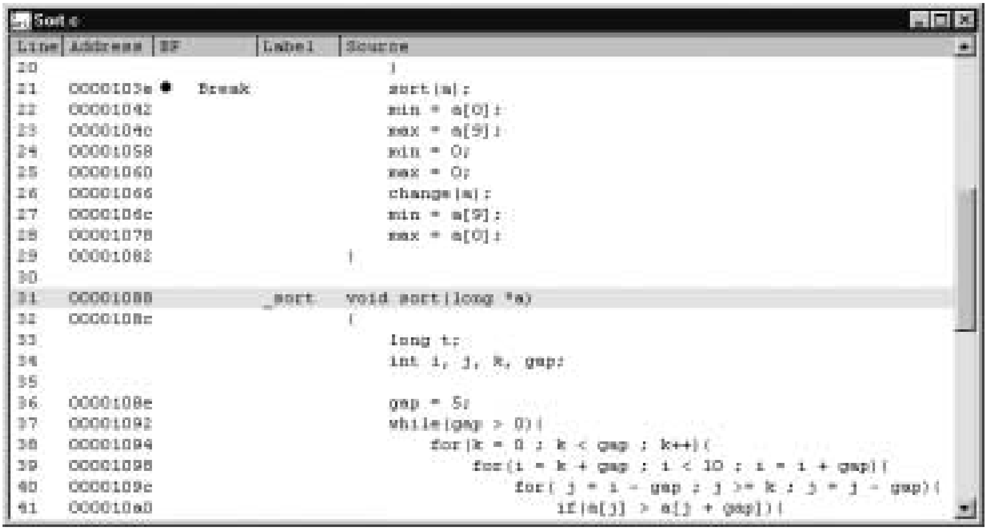


Figure 4.25 Program Window after Executing the Step In Command

Exit the function, and back to the next statement in the `main` program, by choosing **Step Out** from the **Run** menu, or clicking the **Step Out** button in the toolbar.

Address `H'1042` will be highlighted showing that the emulator has exit from the function.

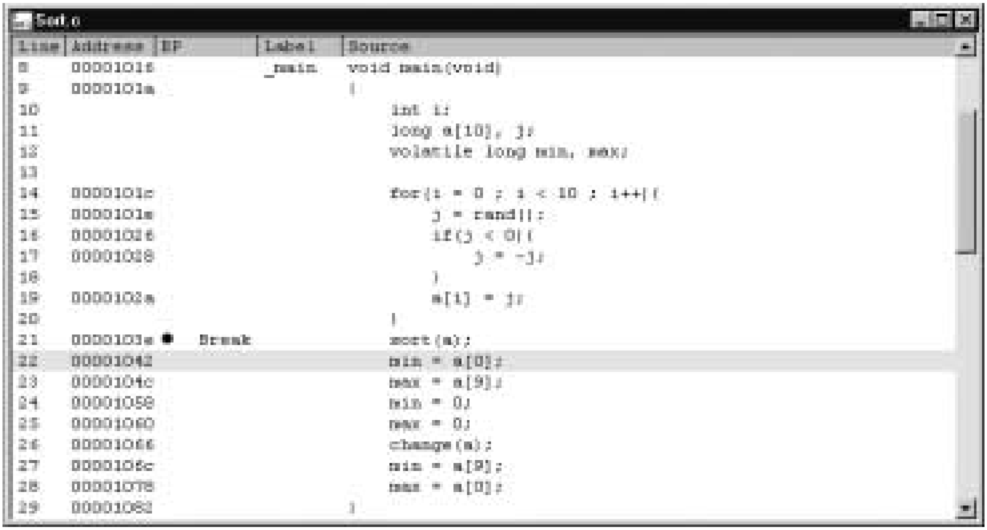


Figure 4.26 Program Window after Executing the Step Out Command

- Use the **Step In** command and execute the program up to the `change` function call.

Note: When a step instruction is executed and a program counter enters the C/C++ library function or execution routine, the Disassembly window is automatically opened. In this state, the step instruction is executed in an assembler level. When the step instruction is executed in the C/C++ source level, exit the C/C++ library function or execution routine with the **Step Out** command and close the Disassembly window.

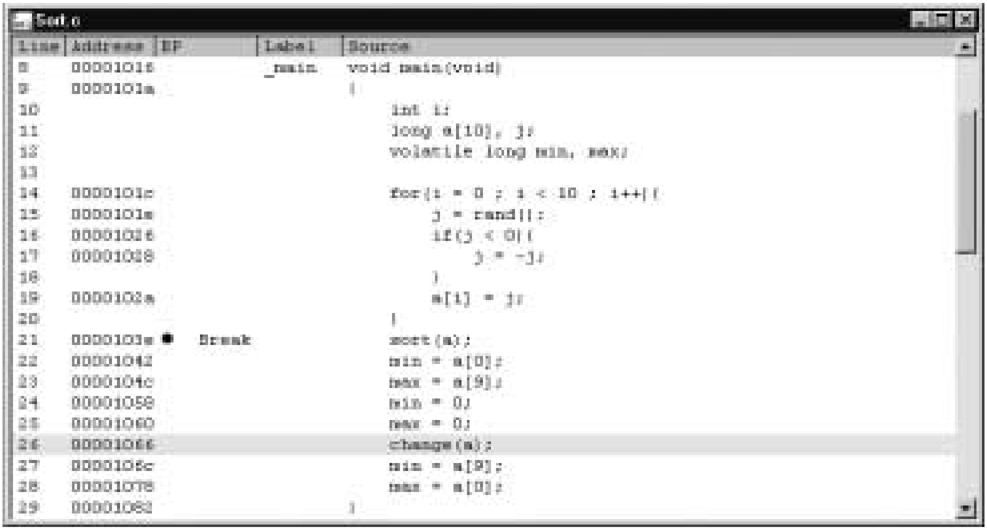


Figure 4.27 Program Window after Executing the Step In Command (2)

4.7.2 Stepping Over a Function

The **Step Over** command executes a function, without single-stepping through the body of the function, and stops at the next statement in the `main` program.

- Choose Step Over from the Run menu, or click the Step Over button in the toolbar.

The program executes the `change` function and stops at the beginning of the next address, `H'106c`.

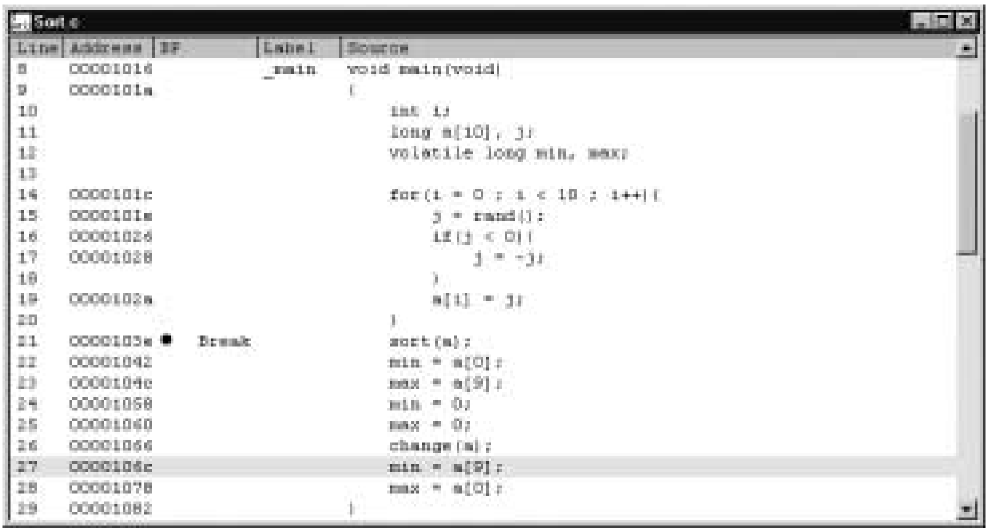


Figure 4.28 Program Window after Executing the Step Over Command

4.7.3 Displaying Local Variables

You can display local variables of a function using the **Locals** window. For example, we will examine the local variables in the function `main`. This function declares five local variables: `a`, `j`, `i`, `min`, and `max`.

- Choose **Locals** from the **View** menu, or click the **Locals** button in the toolbar.

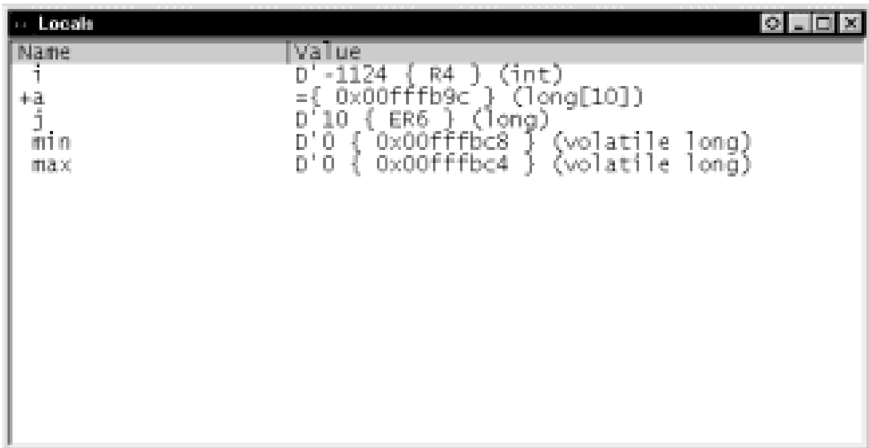
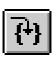


Figure 4.29 Locals Window

The **Locals** window will show nothing when there are no local variables.

- Choose **Step In** from the **Run** menu or click the **Step** button in the toolbar to perform step execution one time.



The contents of variable `min` are changed and their values are displayed.

Figure 4.30 Local Window (After Contents of Variable min are Changed)

- Double-click the + symbol to the left of variable `a` in the **Locals** window to expand variable `a` and to display the individual element of each array.
- Refer to elements of variable `a` before `sort` function execution and confirm that random data has been sorted in descending order.

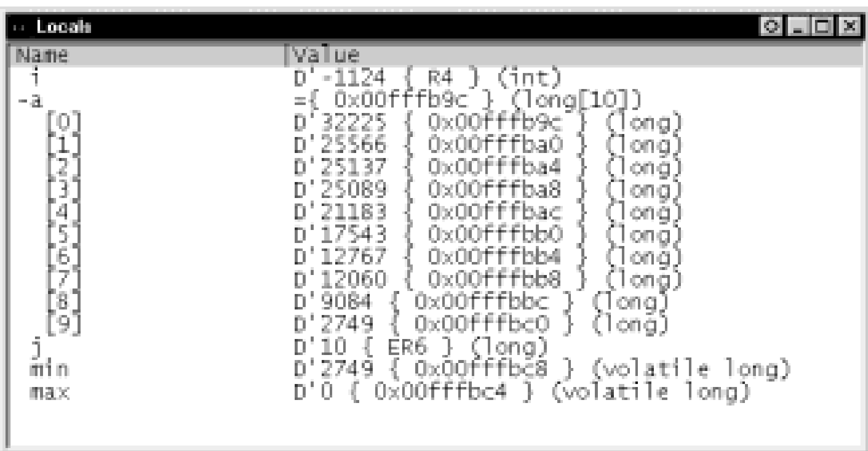


Figure 4.31 Local Window (After Array Variable `a` is Sorted)

4.8 Using the Complex Event System

So far in this tutorial we have monitored the behavior of the program by observing the contents of an area of memory in the **Memory** window, or the values of variables in the **Watch** and **Locals** windows.

Sometimes the action of a program is too complex to allow us to do this. Using the emulator's complex event system, you can, for example, detect the time when the program has accessed `H'1098` five times.

4.8.1 Defining an Event Using the Complex Event System

Now define an event, using the complex event system, to monitor a part of the program as follows:

- Select Hexadecimal in the Radix submenu from the Setup menu, or click the Radix = Hex button in the toolbar to display hexadecimal.

When hexadecimal is input, prefix H' for the radix can be omitted.

- Choose **Breakpoints** from the **View** menu to display the **Breakpoints** window, or click the **Breakpoints** button in the toolbar.

- Click in the **Breakpoints** window with the right mouse button, and choose **Add...** to set a new breakpoint.

The following dialog box allows you to set the breakpoint's properties.

- Set the **Type** to `Event` and enter the address `H'1098` into the **Address Lo** box as a condition.

Figure 4.32 Adding Breakpoints (Address Specification)

- Click the **Action** tab and display the **Action** panel.
- To generate a break after accessing five times, enter 5 in the **Required number of event occurrences** edit box.

Figure 4.33 Adding Breakpoints (Count Specification)

- Click **OK** to define the breakpoint.

A break occurs when address `H'1098` has accessed (read or written) five times.

The **Breakpoints** window shows the new event you have defined.

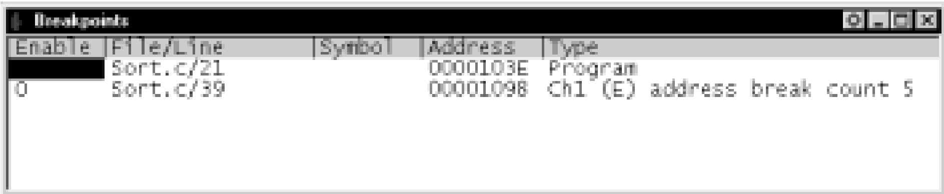


Figure 4.34 Breakpoints Window

- Choose **Reset Go** from the **Run** menu, or click the **Reset Go** button in the toolbar.

Execution will stop at the PC breakpoint set at address `H'103e`.

- Run the program from the current position, by choosing **Go** from the **Run** menu, or click the **Go** button in the toolbar.

The execution stops when address `H'1098` has accessed five times.

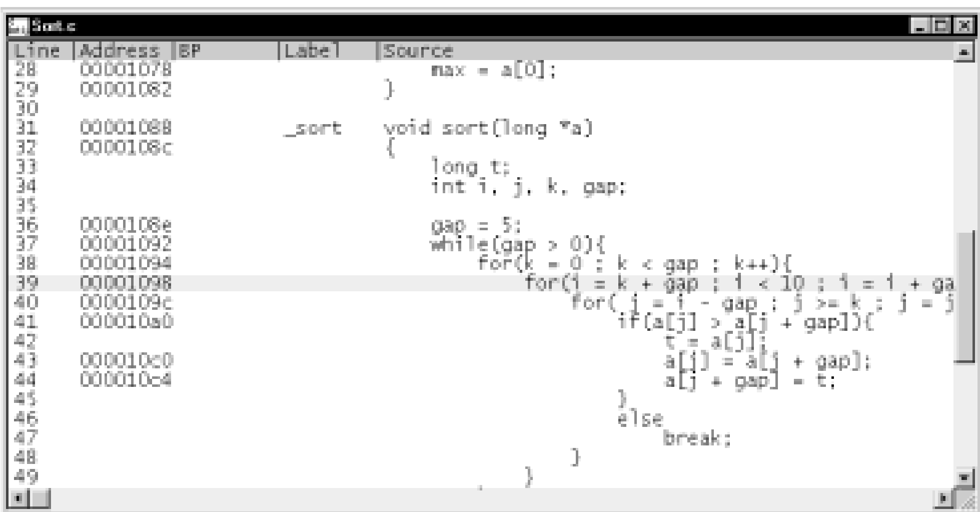


Figure 4.35 Stopping the Program by an Event Breakpoint

The status bar will display `Break = Complex Event System` to indicate that the break was caused by satisfaction of the event condition.

Note: In the complex event system, after the event condition is satisfied, a delay will occur until execution stops.

4.9 Using the Trace Buffer

The trace buffer allows you to look back over previous MCU cycles to see exactly what the MCU was doing prior to a specified event.

4.9.1 Displaying the Trace Buffer

You can specify the address accessed by the program to use the trace buffer to look back to see what accesses took place.

- Open the **Trace** window by choosing **Trace** from the **View** menu, or click the **Trace** button in the toolbar.

If necessary scroll the window down so that you can see the last few cycles. The **Trace** window is displayed, as shown in figure 4.36.

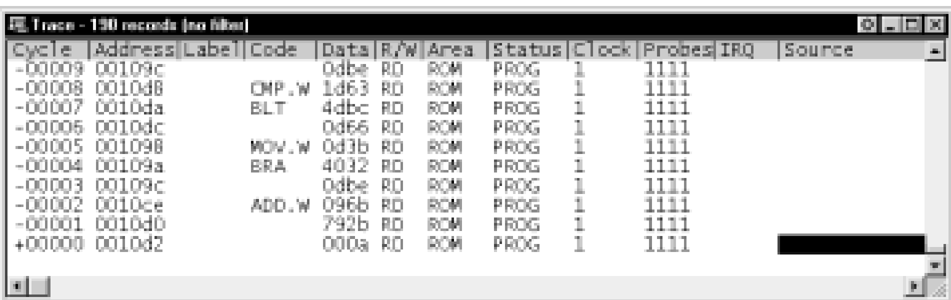


Figure 4.36 Trace Window

- If necessary, adjust the width of each column by dragging the column dividers on either side of the labels just below the title bar.

In cycle `-00005`, you can see that address `H'1098` has been accessed.

4.9.2 Setting a Trace Filter

Currently the **Trace** window shows all the MCU cycles.

- Display the **Trace Filter** dialog box by clicking the **Trace** window with the right mouse button and selecting **Filter...** from the popup menu.

This allows you to define a filter to restrict which cycles will be displayed in the trace buffer.

- If necessary, click **General** to show the **General** panel.
- Select `Pattern` in the **Type** section.
- In the **Address** section click `Address` and type `H'1098` in the **Address Lo** field.

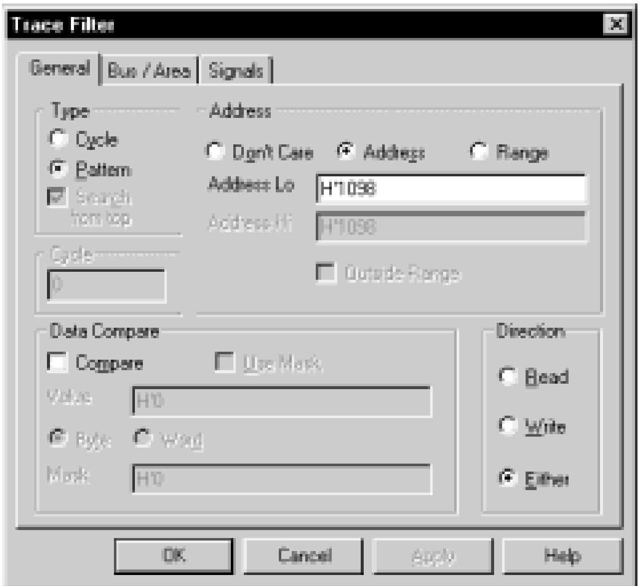


Figure 4.37 General Panel in Trace Filter Dialog Box

- Click **Bus / Area** to display the **Bus / Area** panel.
- Set **Bus State** to `CPU Prefetch`.

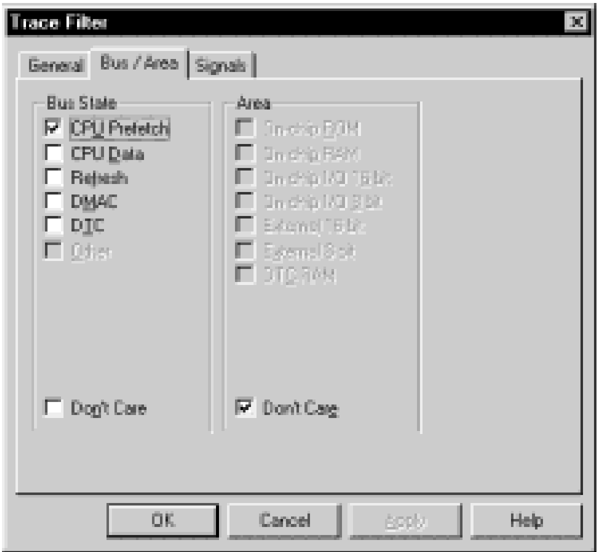


Figure 4.38 Bus / Area Panel in Trace Filter Dialog Box

- Click **OK** to save the trace filter.

In the **Trace** window, only the cycles in which the MCU accessed address `H'1098` are displayed. The program has stopped by accessing `H'1098` five times.

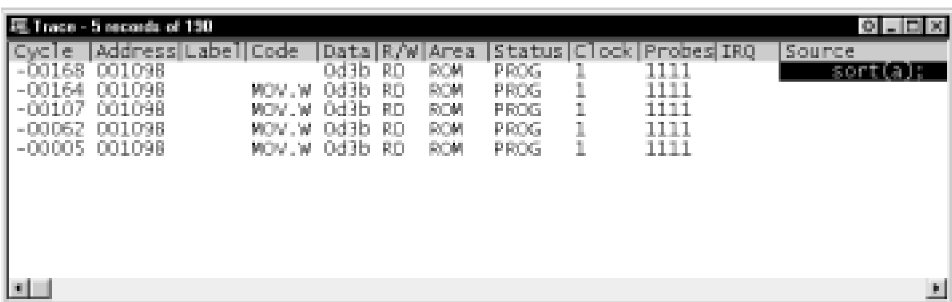


Figure 4.39 Showing Trace Buffer Contents

4.10 Measuring the Performance

By using the performance analysis function in the HDI, you can measure the performance of a program. The results are displayed as a histogram or as percentages.

4.10.1 Selecting the Measurement Conditions

Select the conditions for measurement as follows:

- Select **Performance Analysis** from the **View** menu or click the **PA** button in the toolbar and open the **Performance Analysis** dialog box.

- Click the **Conditions** button and open the **Performance Analysis Conditions** window.
- After clicking **No. 1** in the **Performance Analysis Conditions**, click the **Edit** button and open the **Performance Analysis Properties** dialog box.

The following dialog box will be displayed to allow selection of the measuring conditions.

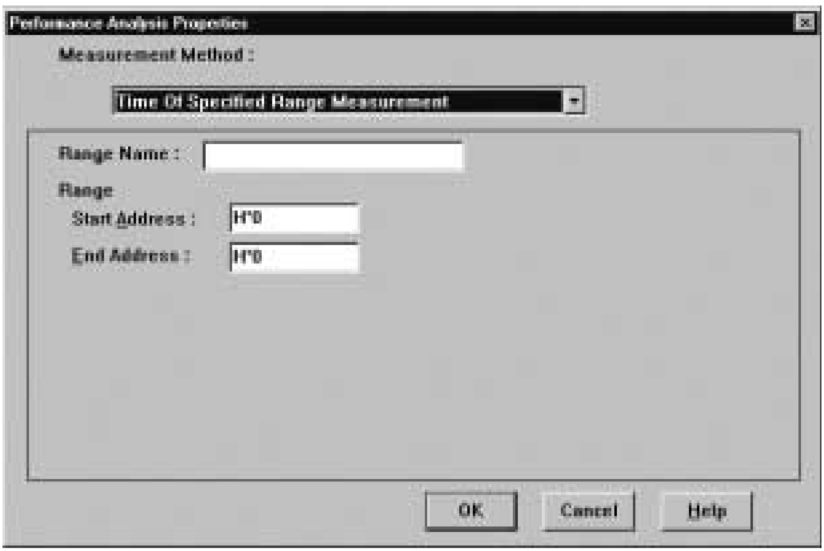


Figure 4.40 Selecting the Conditions for Measurement

- Select **Time Of Specified Range Measurement** from **Measurement Method** to measure the performance over the specified range.
- Input `Analysis` as the **Range Name**.
- Input address `H'1088` as the **Start Address** and address `H'10ec` as the **End Address**.
- Click **OK** to select the conditions.

This completes the setting for No.1.

In the **Performance Analysis Conditions** window, the conditions selected in the **Performance Analysis Properties** dialog box are displayed.

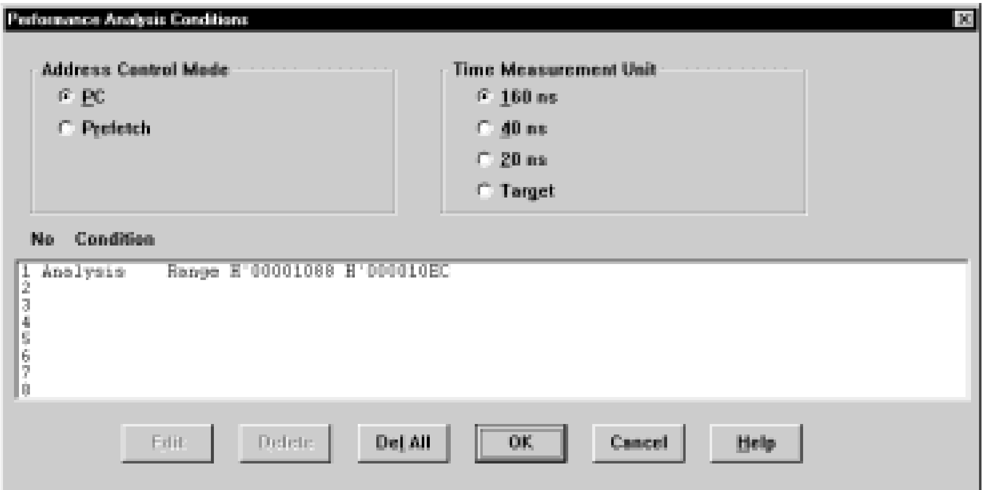


Figure 4.41 Displaying the Measurement Conditions

- Click **OK** to set the measurement conditions.

Now, the performance of the execution in the address range `H'1088` to `H'10ec` can be measured.

- Click **Close** and close the **Performance Analysis** dialog box.
- Double-click the **BP** column of the line that includes address `H'1082` and set a PC break.
- Select **Reset Go** from the **Run** menu or click the **Reset Go** button in the toolbar, and execute the program from the beginning.

The program will stop at address `H'1082`.

4.10.2 Displaying the Analysis Results

The performance analysis results are displayed as a histogram or as percentages.

- Select **Performance Analysis** from the **View** menu or click the **PA** button in the toolbar and open the **Performance Analysis** dialog box.

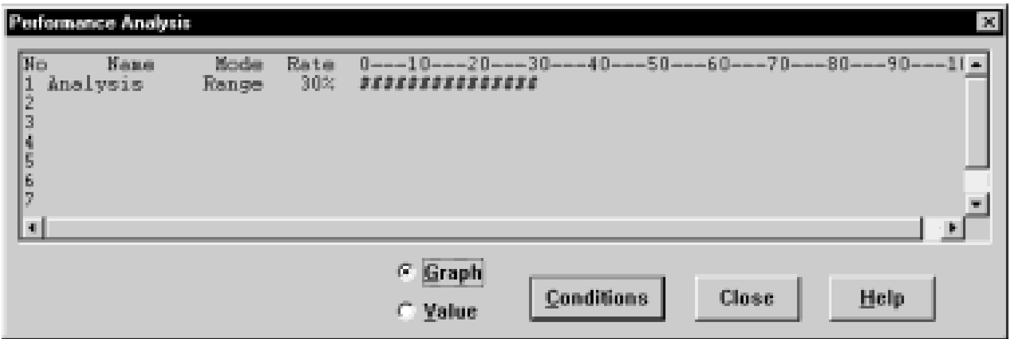


Figure 4.42 Displaying the Analysis Results (1)

The performance analysis results are displayed as a histogram and as percentages.

- Click **Value**.

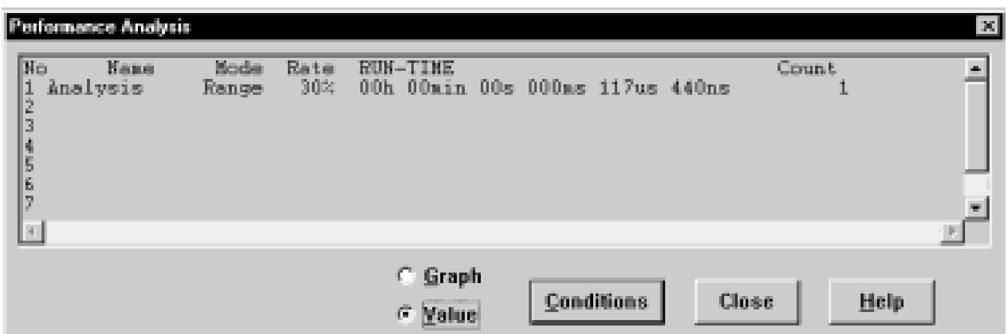


Figure 4.43 Displaying the Analysis Results (2)

The analysis results are displayed as percentages and as the actual time measured.

4.11 Bus Monitor

The bus monitor functions enable the user to display memory contents in realtime during the user program execution. Display of the memory contents is updated at minimum intervals of 1 s.

- Select the **Bus Monitor Window...** from the **View** menu.

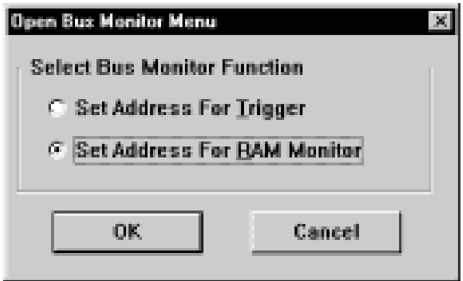


Figure 4.44 Open Bus Monitor Menu Dialog Box

- Select **Set address for RAM Monitor** under **Set Bus Monitor Function**, then click **OK**.
- Select the **Monitor1** check box, enter `H'fffa00` in the text box, select **Access**, then click **OK**.

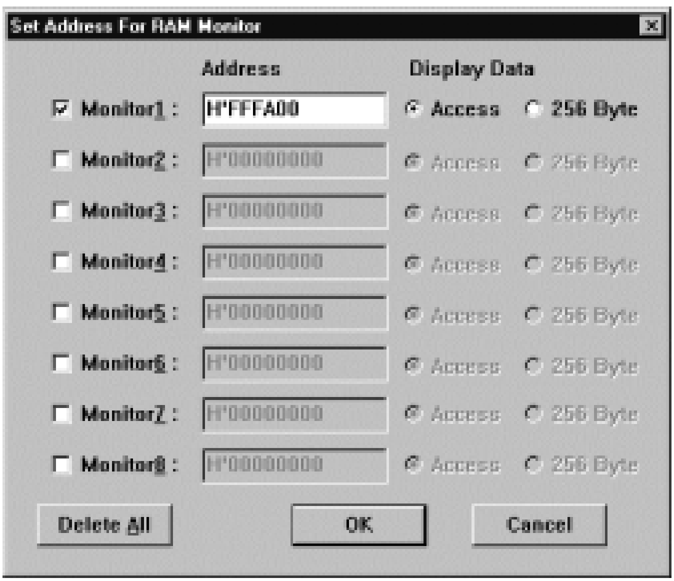


Figure 4.45 Set Address For RAM Monitor Dialog Box

- Select **Reset Go** from the **Run** menu or click the **Reset Go** button of the toolbar.

The following window displays how the memory contents are modified in realtime (the display is updated at minimum intervals of 1 s).

In this tutorial, since program execution stops at PC breakpoints, only the addresses that satisfy the specified condition are displayed.

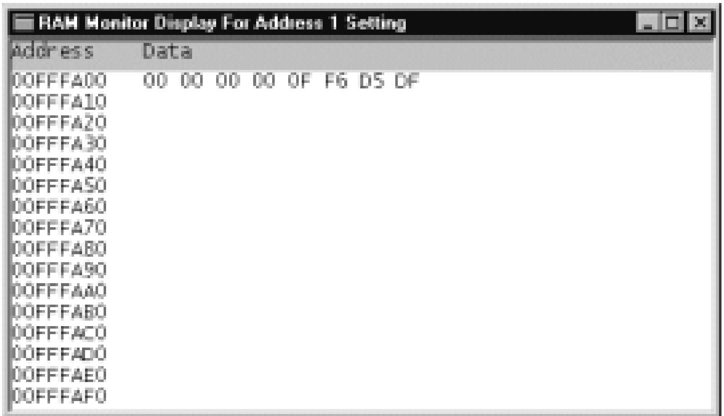


Figure 4.46 RAM Monitor Display Window

4.12 Stack Trace Function

The function-call history can be checked by using the stack trace function when the user program is halted.

- Double-click the **BP** column of the line that includes address `H'108e` and set a PC break.
- Select **Reset Go** from the **Run** menu or click the **Reset Go** button in the toolbar, and execute the program from the beginning.

Execution stops at address `H'108e` by the PC break that has been set.

- Select **Stack Trace** from the **View** menu to open the **Stack Trace** window.

RENESAS

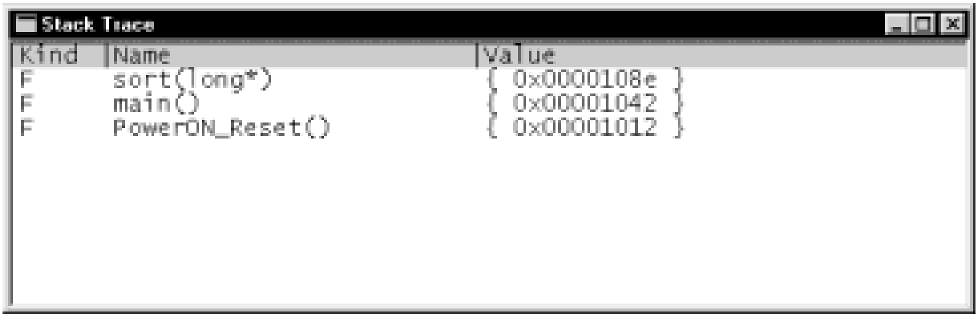


Figure 4.47   Stack Trace Window

Figure 4.47 shows that the position of the program counter is currently at the selected line of the `sort()` function, and that the `sort()` function is called from the `main()` function.

Note: This function can be used only when the load module that has the Dwarf2-type debugging information is loaded.

For details on the functions described above, refer to the on-line help, which can be displayed by clicking the **Help** button or pressing the F1 key when the target window is open.

4.13 Saving the Session

Before exiting, it is good practice to save your session, so that you can resume with the same E6000 emulator and HDI configuration at your next debugging session.

- Choose **Save Session** from the **File** menu.
- Choose **Exit** from the **File** menu to exit HDI.

4.14 What Next?

This tutorial has introduced you to some of the key features of the E6000 emulator, and their use in conjunction with the HDI. By combining the emulation tools provided in the E6000 emulator you can perform extremely sophisticated debugging, allowing you to track down hardware and software problems efficiently by precisely isolating and identifying the conditions under which they occur. For details on HDI operation, refer to the Hitachi Debugging Interface User's Manual, supplied separately.

Appendix A Command Line Functions

This section lists the E6000 emulator command line functions.

Command Type:
General: HDI general commands
Specific: Commands specific to the E6000 emulator

For HDI general command line functions, refer to the Hitachi Debugging Interface User's Manual or the on-line help. For E6000-specific commands, refer to the on-line help. To display the on-line help, enter the following in the **Command Line** window:

```
help <command>
```

<command>: Command name or its abbreviation

Table A.1 Command List

Command Name	Abbrevia-tion	Command Type	Description
!	—	General	Comments
ACCESS	AC	General	Sets operation for invalid access
ANALYSIS	AN	Specific	Enables or disables the performance analysis range
ANALYSIS_RANGE	AR	Specific	Sets or displays the performance analysis range
ANALYSIS_RANGE_DELETE	AD	Specific	Cancels the performance analysis range
ASSEMBLE	AS	General	Assembles a program
ASSERT	—	General	Checks conditions
BREAKPOINT / EVENT	BP, EN	Specific	Sets a breakpoint or an event
BREAKPOINT_CLEAR, EVENT_CLEAR	BC, EC	Specific	Clears a breakpoint or an event
BREAKPOINT_DISPLAY, EVENT_DISPLAY	BD, ED	Specific	Displays breakpoints or events
BREAKPOINT_ENABLE, EVENT_ENABLE	BE, EE	Specific	Enables or disables a breakpoint or an event
BREAKPOINT_SEQUENCE, EVENT_SEQUENCE	BS, ES	Specific	Defines or clears a breakpoint or event sequence

Table A.1 Command List (cont)

Command Name	Abbreviation	Command Type	Description
CLOCK	CK	Specific	Sets the CPU clock rate in the E6000 emulator
DEVICE_TYPE	DE	Specific	Selects the target device in the E6000 emulator
DISASSEMBLE	DA	General	Disassembles and displays a program
ERASE	ER	General	Clears the contents of the Command Line window
EVALUATE	EV	General	Evaluates an expression
FILE_LOAD	FL	General	Loads an object program file
FILE_SAVE	FS	General	Saves memory contents in a file
FILE_VERIFY	FV	General	Verifies memory contents against file contents
GO	GO	General	Executes a user program
GO_RESET	GR	General	Executes a user program from the reset vector
GO_TILL	GT	General	Executes a user program until a temporary breakpoint
HALT	HA	General	Stops user program execution
HELP	HE	General	Displays the help message for the command line or the command
INITIALISE	IN	General	Initializes the platform
LOG	LO	General	Manipulates the logging file

Table A.1 Command List (cont)

Command Name	Abbrevia-tion	Command Type	Description
MAP_DISPLAY	MA	General	Displays the memory map information
MAP_SET	MS	Specific	Sets memory mapping
MEMORY_DISPLAY	MD	General	Displays memory contents
MEMORY_EDIT	ME	General	Modifies memory contents
MEMORY_FILL	MF	General	Fills the memory with the specified data
MEMORY_MOVE	MV	General	Moves a memory block
MEMORY_TEST	MT	General	Tests a memory block
MODE	MO	Specific	Sets or displays the MCU mode
MODULES	MU	Specific	Sets or displays the on-chip peripheral functions of the MCU
QUIT	QU	General	Terminates the HDI
RADIX	RA	General	Sets a radix for input value
REFRESH	RF	Specific	Updates the memory-related windows
REGISTER_DISPLAY	RD	General	Displays the MCU register values
REGISTER_SET	RS	General	Sets the MCU register values
RESET	RE	General	Resets the MCU
SLEEP	—	General	Delays command execution
STEP	ST	General	Performs single-step execution in instruction unit or source line unit
STEP_OUT	SP	General	Step out of the current function
STEP_OVER	SO	General	Performs step-over execution
STEP_RATE	SR	General	Set rate for multiple steps

Table A.1 Command List (cont)

Command Name	Abbreviation	Command Type	Description
SUBMIT	SU	General	Executes an emulator command file
SYMBOL_ADD	SA	General	Adds a symbol
SYMBOL_CLEAR	SC	General	Deletes a symbol
SYMBOL_LOAD	SL	General	Loads a symbol information file
SYMBOL_SAVE	SS	General	Saves a symbol information file
SYMBOL_VIEW	SV	General	Displays a symbol
TEST_EMULATOR	TE	Specific	Tests the E6000 emulator hardware
TIMER	TI	Specific	Sets or displays the timer minimum measurement unit for execution time measurement
TRACE	TR	General	Displays trace data
TRACE_ACQUISITION	TA	Specific	Sets or displays trace acquisition information
TRACE_COMPARE	TC	Specific	Compares trace data
TRACE_SAVE	TV	Specific	Saves trace data
TRACE_SEARCH	TS	Specific	Searches for trace data
USER_SIGNALS	US	Specific	Enables or disables user signals

Note: No commands are available for the bus monitor functions.

H8S Series E6000 Emulator User's Manual

Publication Date: 1st Edition, March 1999
3rd Edition, January 2001
Published by: Electronic Devices Sales & Marketing Group
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.