

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

Hitachi Microcomputer Development Environment System

AE-4X Series

E6000 Emulator

User's Manual

HITACHI

ADE-702-314

Rev. 1.0

03/28/02

Hitachi, Ltd.

HS0AE4XEPI61HE

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, and PC interface board, 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 AE-4 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:

This user's manual and E6000 emulator 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.

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.
- **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 indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.



WARNING indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.



CAUTION indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.



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. Before inserting the user system interface cable, make sure that front and back sides are correctly aligned to the user system.**
- 4. Supply power according to the power specifications and do not apply an incorrect power voltage. Use only the provided power cable.**

Preface

Thank you for purchasing the AE-4X series E6000 emulator.

The AE-4X series E6000 emulator (hereafter referred to as the E6000) was designed as a software and hardware development tool for systems based on Hitachi's original AE-4 series microcomputers, including the ISO14443 contactless type.

The E6000 provides a CD-R that contains the Hitachi Debugging Interface (HDI) system program, a test program, and the user's manual.

There are two manuals for the E6000: AE-4X Series E6000 Emulator User's Manual (this manual) and the Hitachi Debugging Interface User's Manual. The E6000 Emulator User's Manual describes E6000 functions common to all AE-4X series microcomputers. Please read this manual before using the E6000.

About This Manual

This manual describes how to set up and use the E6000 emulator for AE-4X series microcomputers and gives specifications for the 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 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 provided on the CD-R so that you can follow the steps on your own system and learn firsthand how it operates.

Section 5, Diagnostic Test Procedure, describes the diagnostic test procedure using the E6000 test program.

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

This manual also assumes that the operating environment is the English version of Microsoft[®] Windows[®] 98 running on the IBM PC.

Assumptions

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

This manual also assumes that the operating environment is the English version of Microsoft[®] Windows[®] 98 running on the IBM PC.

Related Manuals

- Hitachi Debugging Interface User's Manual
- PC Interface Board User's Manual (indicates one of the following manuals 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)

LAN Adapter User's Manual (HS6000ELN01HE)

Conventions

This manual uses the following typographical conventions:

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

Trademarks

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

IBM is a registered trademark of International Business Machines Corporation.

Contents

About This Manual	ii
Section 1 Introduction	1
1.1 Debugging Features	1
1.1.1 Breakpoints	1
1.1.2 Trace	1
1.1.3 Execution Time Measurements	2
1.1.4 Performance Analysis	2
1.1.5 Bus Monitoring	2
1.1.6 Coverage	2
1.2 Complex Event System (CES)	2
1.2.1 Event Channels	3
1.2.2 Range Channels	3
1.2.3 Breaks	4
1.2.4 Timing	4
1.3 Hardware Features	4
1.3.1 Memory	4
1.3.2 Clocks	5
1.3.3 Probes	5
1.3.4 Environment Conditions	6
1.3.5 Emulator External Dimensions and Mass	6
Section 2 Setting Up	7
2.1 Package Contents	7
2.2 Setting Up the PC Interface Board on Windows® 98	8
2.2.1 Setting Up the PC Interface Board	8
2.2.2 Modifying the CONFIG.SYS File	11
2.2.3 Modifying the SYSTEM.INI File	12
2.3 Setting Up the PC Interface Board on Windows NT® 4.0	13
2.4 Installing the HDI	14
2.5 Troubleshooting	15
2.5.1 Faulty Connection	15
2.5.2 Communication Problems	16
Section 3 Hardware	17
3.1 Connecting to the User System	17

3.1.1	Connecting the User System Interface Cable Body to the E6000 Emulator	18
3.1.2	User System Interface Cable Head	19
3.1.3	Connecting the Use System Interface Cable Head.....	20
3.1.4	Antenna at the User System Interface Cable Head	21
3.2	Power Supply	22
3.2.1	AC Adapter	22
3.2.2	Polarity of Power Supply Plug	22
3.2.3	Power Supply Monitor Circuit.....	23
3.3	Hardware Interface.....	23
3.3.1	Signal Protection on the E6000 Emulator.....	23
3.3.2	User System Interface Circuits.....	23
3.3.3	External Probes/Trigger Output 1	25
3.3.4	External Probe 2 (EXT2)/Trigger Output	26
3.3.5	Power Supply Circuit.....	27
3.4	Jumper Pin Settings.....	28
3.5	Differences between MCU and E6000 Emulator.....	29
3.5.1	Registers.....	29
3.5.2	EEPROM	29
3.5.3	WDT	30
3.5.4	SYSCR Register.....	30
3.5.5	Security	31
3.5.6	Contactless Operation	31
Section 4 Tutorial.....		32
4.1	Introduction.....	32
4.2	Starting HDI.....	33
4.2.1	Selecting the Target Platform	34
4.3	Setting up the E6000 Emulator	36
4.3.1	Configuring the Platform	36
4.3.2	Mapping the Memory	38
4.4	Downloading the Tutorial Program	41
4.4.1	Loading the Object File	41
4.4.2	Displaying the Program Listing	42
4.5	Using Breakpoints.....	44
4.5.1	Setting a PC Break	44
4.5.2	Executing the Program.....	46
4.5.3	Examining Registers	48
4.5.4	Reviewing the Breakpoints.....	49
4.6	Examining Memory and Variables	50
4.6.1	Viewing Memory	50

4.6.2	Watching Variables.....	51
4.7	Stepping Through a Program	54
4.7.1	Single Stepping	54
4.7.2	Stepping Over a Function	59
4.7.3	Displaying Local Variables.....	60
4.8	Using the Complex Event System.....	62
4.8.1	Defining an Event Using the Complex Event System	62
4.9	Using the Trace Buffer.....	66
4.9.1	Displaying the Trace Buffer.....	66
4.9.2	Setting a Trace Filter.....	67
4.10	Measuring the Performance	70
4.10.1	Selecting the Measurement Conditions.....	70
4.10.2	Displaying the Analysis Results.....	73
4.11	Bus Monitor	74
4.12	Stack Trace Function	76
4.13	Coverage Function	77
4.14	Saving the Session	78
4.15	What Next?	80
Section 5 Diagnostic Test Procedure.....		81
5.1	System Set-Up for Test Program Execution	81
5.2	Diagnostic Test Procedure Using Test Program	82
Appendix A Command Line Functions.....		89

Figures

Figure 2.1	Computer Properties Dialog Box (Before Setting)	8
Figure 2.2	Edit Resource Setting Dialog Box	10
Figure 2.3	Computer Properties Dialog Box (After Setting)	11
Figure 2.4	Faulty Connection Message	15
Figure 2.5	Communication Problem Message	16
Figure 3.1	E6000 Emulator Connectors	17
Figure 3.2	External View of the E6000 Emulator	18
Figure 3.3	External View of the Contact less User System Interface Cable	19
Figure 3.4	External View of the User System Interface Cable Head	21
Figure 3.5	Resonance Characteristics of the User System Interface Cable Head	22
Figure 3.6	Polarity of Power Supply Plug	22
Figure 3.7	User System Interface Signal Circuit	24
Figure 3.8	User System Interface Circuit for CLK	24
Figure 3.9	User System Interface Circuit for RESET	24
Figure 3.10	User System Interface Circuit for I/O-1/IRQ and I/O-2/IRQ	25
Figure 3.11	User System Interface Circuit for La and Lb	25
Figure 3.12	Probe1 and Trigger Connector	25
Figure 3.13	External Probe1 Interface Circuit	26
Figure 3.14	Connector for External Probe 2	26
Figure 3.15	Relationship of Vcc between the User System and E6000 Emulator	27
Figure 4.1	HDI Start Menu	33
Figure 4.2	Select Platform Dialog Box	34
Figure 4.3	Hitachi Debugging Interface Window	35
Figure 4.4	Configuration Dialog Box	37
Figure 4.5	Memory Mapping Dialog Box	39
Figure 4.6	Edit Memory Mapping Dialog Box	40
Figure 4.7	System Status Window (Memory Sheet)	41
Figure 4.8	Open Dialog Box (Object File Selection)	42
Figure 4.9	HDI Dialog Box	42
Figure 4.10	Open Dialog Box (Source File Selection)	43
Figure 4.11	Source Program Window	44
Figure 4.12	Setting a Breakpoint (PC Break)	45
Figure 4.13	Program Break	46
Figure 4.14	System Status Window (Platform Sheet)	47
Figure 4.15	Registers Window	48
Figure 4.16	Register Dialog Box	49
Figure 4.17	Breakpoints Window	49

Figure 4.18	Open Memory Window Dialog Box	50
Figure 4.19	Memory Window (Byte)	51
Figure 4.20	Watch Window (After Adding Variables).....	52
Figure 4.21	Watch Window (Symbol Expansion)	53
Figure 4.22	Add Watch Dialog Box	53
Figure 4.23	Watch Window (Adding Variables)	54
Figure 4.24	Program Window after Executing the Reset Go Command	55
Figure 4.25	Program Window after Executing the Step In Command (1).....	56
Figure 4.26	Program Window after Executing the Step Out Command.....	57
Figure 4.27	Program Window after Executing the Step In Command (2).....	58
Figure 4.28	Program Window after Executing the Step Over Command.....	59
Figure 4.29	Locals Window.....	60
Figure 4.30	Locals Window (After Contents of Variable min are Changed)	61
Figure 4.31	Locals Window (Elements of Array Variable a after Function sort has been Executed)	62
Figure 4.32	Breakpoint/Event Properties Dialog Box	63
Figure 4.33	Breakpoints Window	64
Figure 4.34	Stopping the Program by an Event Breakpoint	65
Figure 4.35	Trace Window	66
Figure 4.36	General Panel in Trace Filter Dialog Box	67
Figure 4.37	Bus / Area Panel in Trace Filter Dialog Box	68
Figure 4.38	Trace Window (When Trace Filter is Specified).....	69
Figure 4.39	Selecting the Conditions for Measurement.....	71
Figure 4.40	Displaying the Measurement Conditions.....	72
Figure 4.41	Displaying the Analysis Results (1)	73
Figure 4.42	Displaying the Analysis Results (2)	74
Figure 4.43	Open Bus Monitor Menu Dialog Box	74
Figure 4.44	Set Address For RAM Monitor Dialog Box.....	75
Figure 4.45	RAM Monitor Display	76
Figure 4.46	Stack Trace Window	77
Figure 4.47	Coverage Main Window.....	78

Tables

Table 1.1	Memory Types.....	5
Table 1.2	Environment Conditions.....	6
Table 2.1	Address Map of PC Interface Board and Memory Switch Setting.....	9
Table 3.1	Initial Value Differences between MCU and E6000 Emulator.....	29
Table 3.2	EEPROM Differences between MCU and E6000 Emulator.....	29
Table 3.3	WDT Differences between MCU and E6000 Emulator.....	30
Table 3.4	SYSCR Register Differences between MCU and E6000 Emulator.....	30
Table 3.5	Security Differences between MCU and E6000 Emulator.....	31
Table 3.6	Differences in Contactless Operation.....	31
Table 4.1	Configuration Options.....	38
Table 4.2	Memory Types.....	39
Table 4.3	Access Types.....	39
Table 4.4	Step Commands.....	54
Table A.1	Command List.....	89

Section 1 Introduction

The E6000 emulator is an advanced realtime in-circuit emulator which allows programs to be developed and debugged for the AE-4 series microcomputers (herein after called the MCU).

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

The E6000 emulator works with the Hitachi debugging interface (HDI), a Microsoft® Windows®-based interface program. 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 with a BREAK instruction.

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 μ s, 2 μ s, 4 μ s, 8 μ s, or 16 μ s.

At 20 ns, the maximum time that can be measured is approximately six hours, and at 16 μ s 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 to external probe 2 (EXT2) when specified addresses (four points max.) are accessed.

1.1.6 Coverage

The E6000 emulator provides a coverage function. The coverage function executes programs and obtains and stores histories of data access. This function allows you to know what part of your program has been or has not been tested.

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., 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
- Data, with an optional mask
- Read or Write or either
- MCU access type (e.g., 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 event detection system can be used to control the break and trace 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 the on-chip ROM or RAM memory capacity is insufficient during user program development, this emulation memory can be used instead. However, at the final stage of the development, do not use this emulation memory and verify that the user program correctly operates within the on-chip ROM and RAM.

On-chip ROM and RAM can be expanded for memory mapping. However, there are limitations as follows:

ROM: 448 kB from ROM start address. Can only be modified to emulator read-only.

RAM: 64 kB advanced from RAM end address (H'FFCFFF). Can only be modified to emulator read/write.

This expanded area can be specified for the memory on the E6000 emulator using the **Memory Mapping...** command.

Table 1.1 Memory Types

Memory Type	Description
On-chip	Accesses the MCU on-chip memory.
User	Accesses the memory on the user system.
Emulator	Accesses the emulation memory.

Note) "User" can not be set on this emulator.

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.

1.3.2 Clocks

The emulation clock can be specified as any of the following frequencies: 1.7856 MHz, 2.4576 MHz, 3.5712 MHz, 4.9152 MHz, 6.78 MHz, 7.1424 MHz, 9.8304 MHz, or target clock. The E6000 emulator works within the voltage and frequency ranges described in each MCU hardware manual.

1.3.3 Probes

External probe 1 (EXT1) and external probe 2 (EXT2) can be connected to the E6000 emulator to make use of any signal for braking or tracing. The probe 1 signal can be used as an event detection system condition, depending on whether the level is high or low. Probe 2 outputs high-level signals for use in triggering oscilloscopes, etc., when the condition for setting (1 to 4) the bus monitor function is satisfied.

1.3.4 Environment Conditions

Observe the conditions listed in table 1.2 when using HS0AE4EPI61H.

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 voltage	Voltage: 5 V±5% Current: Max 6 A
User system voltage (UVcc)	Voltage: Follow power specification of each MCU within the voltage range of 2.7 V to 5.5 V
AC input power supply	Voltage: 100 to 240 VAC Frequency: 50/60 Hz Power consumption: 61 to 70 VA

1.3.5 Emulator External Dimensions and Mass

Dimensions: 219 x 170 x 54 mm

Mass: 900 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® 98, refer to section 2.2, Setting Up the PC Interface Board on Windows® 98. When using Microsoft® Windows NT®, 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® ME and 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)
- User interface cable
- External probe 1
- External probe 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® 98

2.2.1 Setting Up the PC Interface Board

- Start up 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.

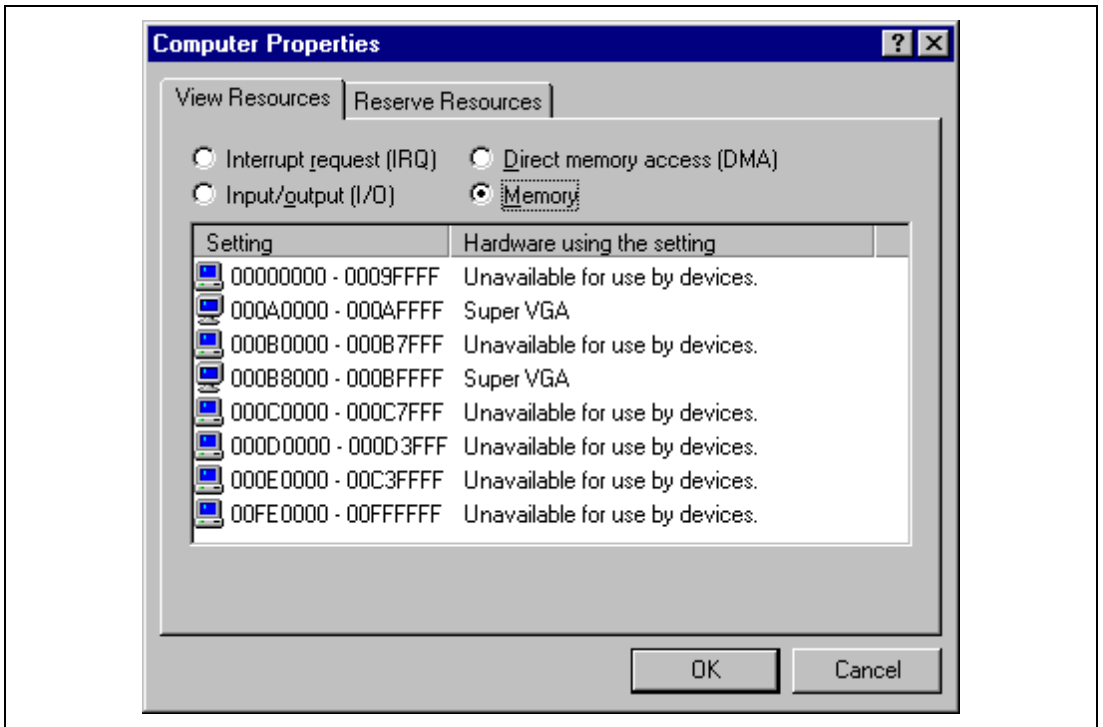


Figure 2.1 Computer Properties Dialog Box (Before Setting)

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® 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.

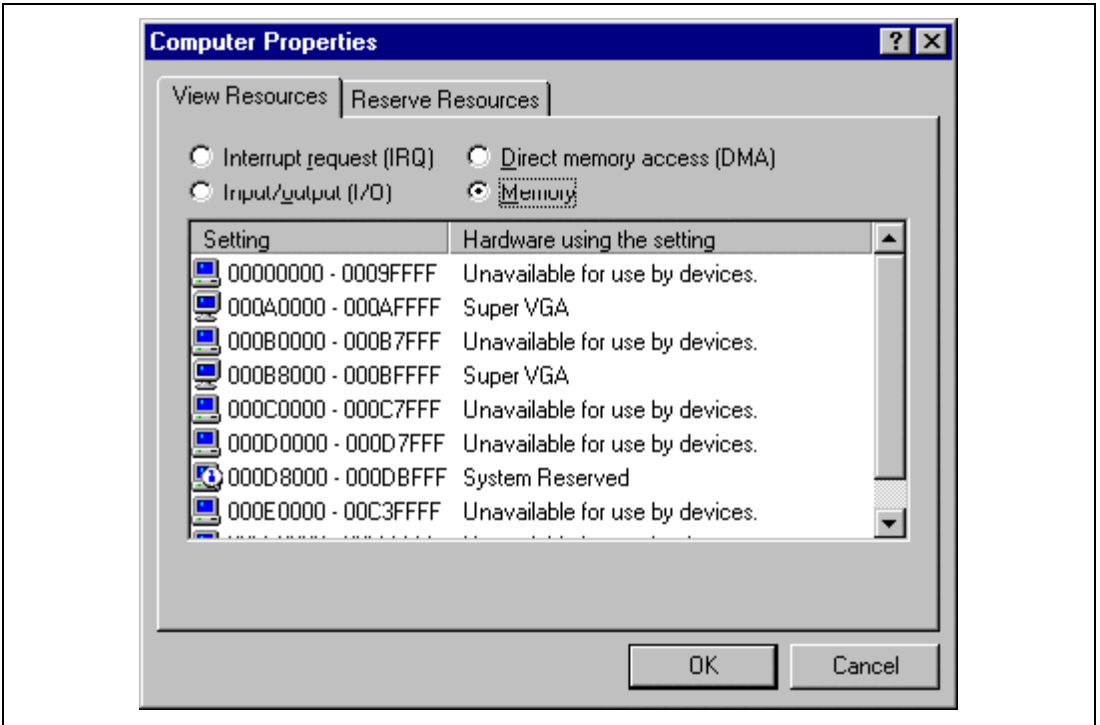


Figure 2.3 Computer Properties Dialog Box (After Setting)

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 Windows® NT 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'CEFFF	7	H'DC000	H'DFFFF	B	H'EC000	H'EFFFF

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.

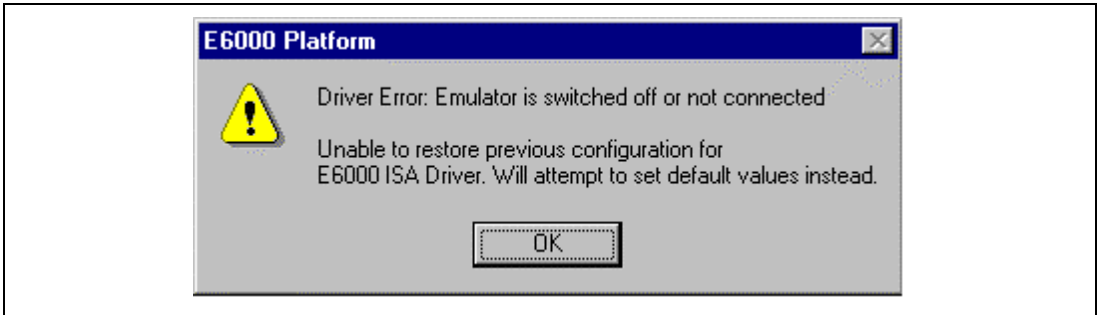


Figure 2.4 Faulty Connection Message

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:

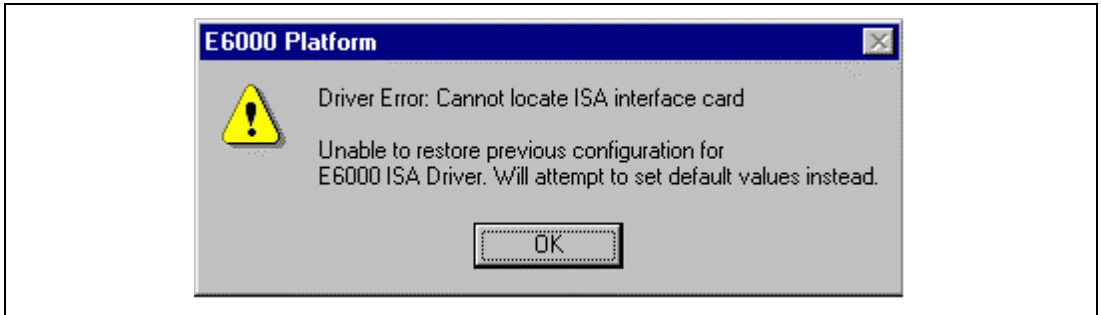


Figure 2.5 Communication Problem Message

This indicates:

- The memory area reserved in the `CONFIG.SYS` file does not match the interface 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 the user system.

3.1 Connecting to the User System

This E6000 emulator package includes a user system interface cable dedicated for use with the IC card reader. The head of the user system interface cable has the same shape as the IC card, and therefore, the E6000 emulator can be easily connected to the IC card reader on the user system by inserting the cable head into the IC card reader. To connect the E6000 emulator to the user system, proceed as follows:

- After checking that the E6000 emulator power is off, plug the user system interface cable body into the E6000 emulator.
- Insert the user system interface cable head into the IC card reader on the user system.
- Turn on and initiate the E6000 emulator.

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

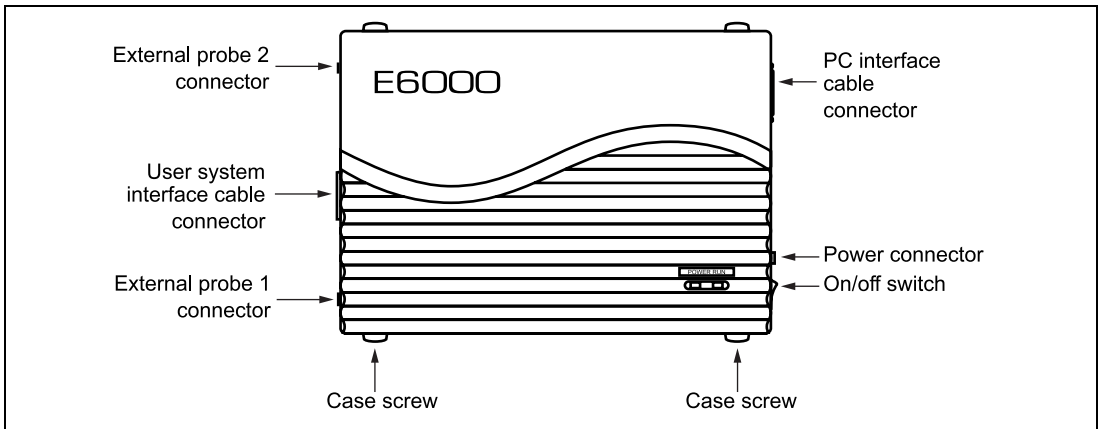


Figure 3.1 E6000 Emulator Connectors

3.1.1 Connecting the User System Interface Cable Body to the E6000 Emulator

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

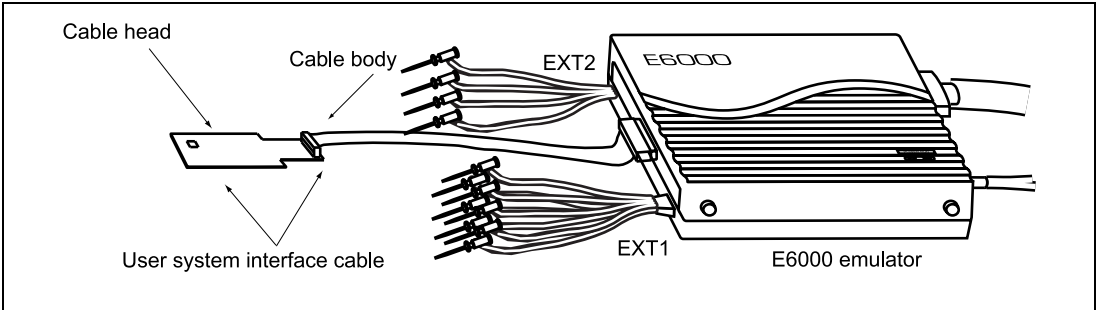


Figure 3.2 External View of the E6000 Emulator

3.1.2 User System Interface Cable Head

The following figure shows an overview of the contactless user system interface cable.

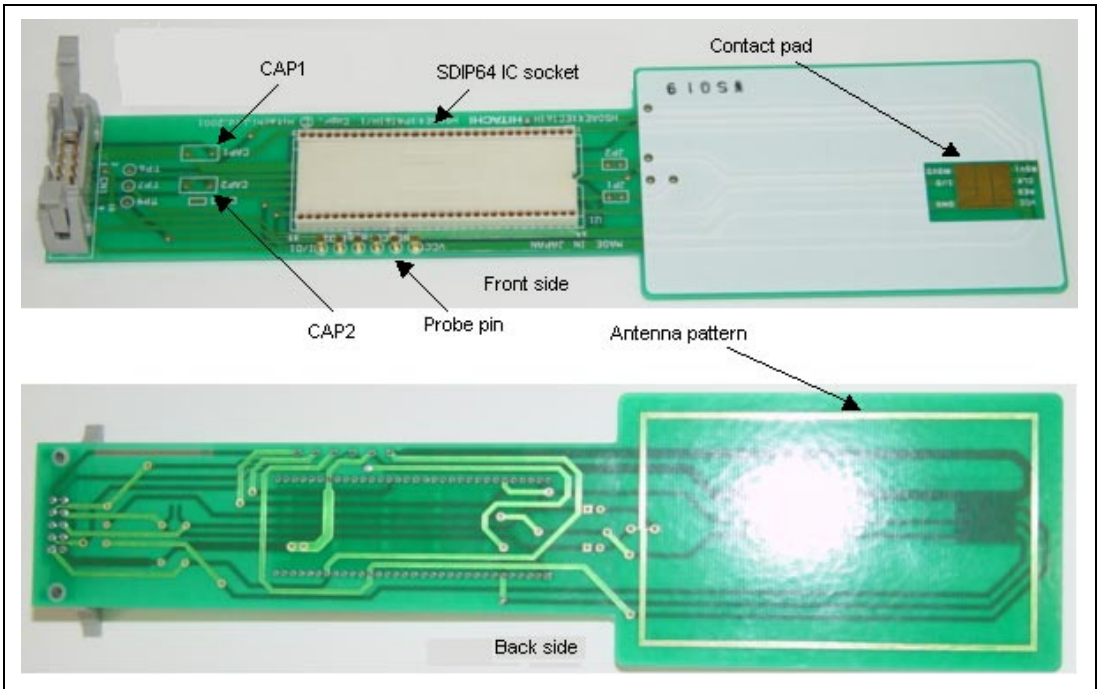


Figure 3.3 External View of the Contactless User System Interface Cable

The following explains each section.

- Through hole CAP1 for bypass capacitor between contact Vcc and GND

Solder a 0.1-uF capacitor to prevent overshooting at the rising edge of Vcc due to your contact IC-card reader.

- Probe pin for measuring contact signals

Use the probe pin to measure Vcc, GND, CLK, RES, and I/O-1 signals with an oscilloscope at contact operation.

- Through hole CAP2 for antenna tuning capacitor

The through hole of an adjustment capacitor is used when contactless operation prevents the emulator from communicating with a contactless IC-card reader.

- Antenna pattern

This antenna pattern is for the contact less operation based on ISO14443.

- SDIP-64 IC socket

This IC socket is used to check working samples. The operation of working samples of SDIP-64-type chips can be checked. This socket is available for the MCU of contact type and Dual Way.

- Contact pad

This contact pad is for contact operation based on ISO7816.

3.1.3 Connecting the Use System Interface Cable Head

To use the emulator in ISO7816 contact operation, plug in the user system interface cable head into the user system after activating the emulator. Note the following when inserting.

- Check that the cable head is facing the right way towards the IC card reader slot.
- When the IC card reader automatically pulls or pushes an IC card into or out of the IC card slot, check the distance in which IC card moves, and place the E6000 emulator at an appropriate distance from the IC card reader.
- Dust or particles on the contact pads on the surface of the cable head will degrade the electrical connection between the cable head and the IC card reader, and the user system interface cable will not operate correctly. To prevent this, clean the contact pads with a dry cloth.

To use the emulator in ISO14443 contactless operation, hold the IC-card-shaped section of the user system interface cable to the contactless IC-card reader.

- The distance at which communication can be provided in contactless operation between the IC-card reader and the user system interface cable varies depending on the output power of the IC-card reader, characteristics of the antenna, etc.
- If communication with your IC-card reader cannot be provided, solder a capacitor to CAP2 for adjustment.
- Contact Hitachi's sales department for types of contactless IC-card readers on the emulator.

Note: The distance for contactless operation between the card and the IC-card reader varies depending on the characteristics of the device, card antenna, and IC-card reader. The shape of the antenna of this user system interface cable is an example for software evaluation. For your real card, design an optimum antenna for your system.

3.1.4 Antenna at the User System Interface Cable Head

Figures 3.4 and 3.5 show the shape and resonance characteristics of an antenna for ISO14443 contact operation on the user system interface cable head.

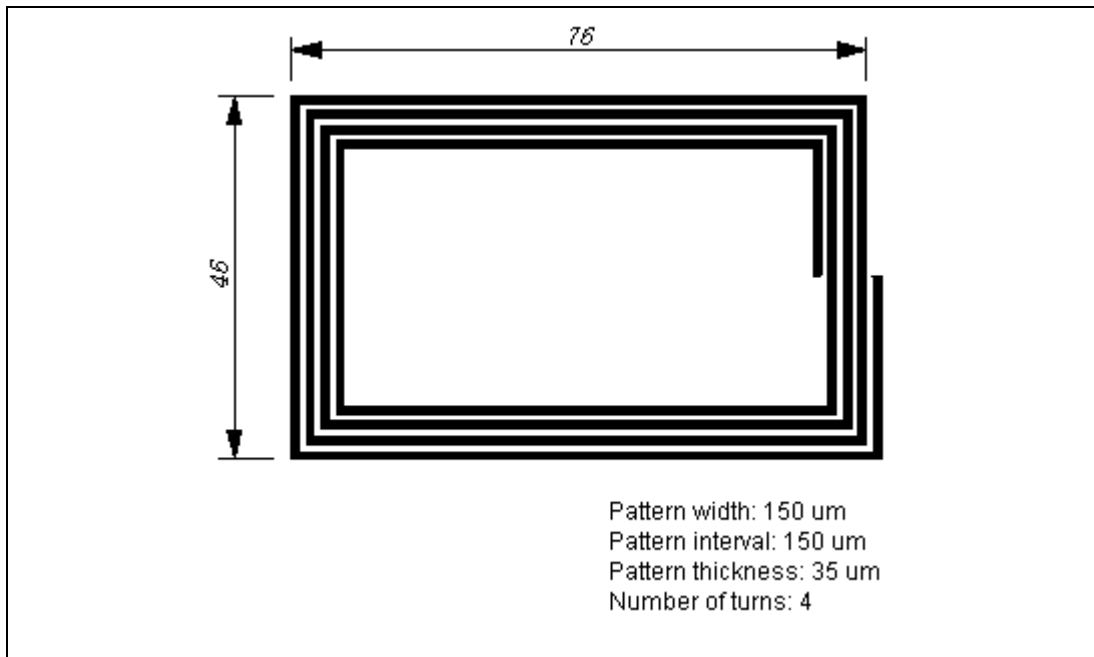


Figure 3.4 External View of the User System Interface Cable Head

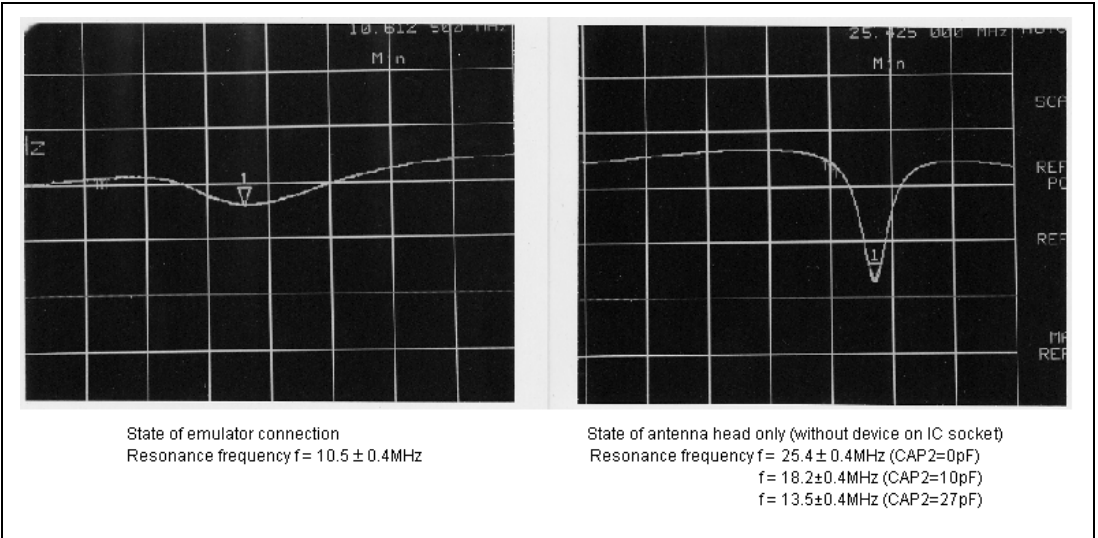


Figure 3.5 Resonance Characteristics of the User System Interface Cable Head (Horizontal axis: frequency 1 MHz/Div, Vertical axis: attenuation factor: relative value (log scale) dB/Div)

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 of Power Supply Plug

Figure 3.6 shows the polarity of the power-supply plug.

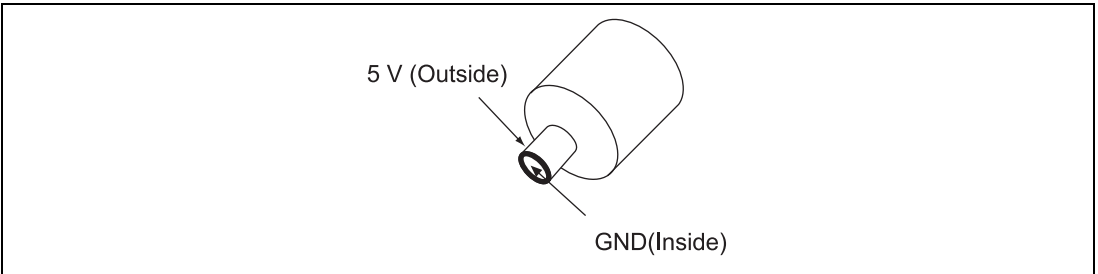


Figure 3.6 Polarity of Power Supply Plug

3.2.3 Power Supply Monitor Circuit

The E6000 emulator incorporates a power supply monitor circuit. From AC adapter that 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 insufficient current is being supplied to the E6000 emulator.

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

3.3 Hardware Interface

For the following user system interface signals, control circuits are connected between the user system and the MCU in the E6000 emulator:

- RESET
- CLK
- I/O-1/IRQ and I/O-2/IRQ
- La and Lb

3.3.1 Signal Protection on the E6000 Emulator

All user system interface signals are over/reverse voltage protected by use of diode arrays and have pull-up resistors.

The VCC pin is monitored by the E6000 emulator to detect a powered user system hardware presence.

3.3.2 User System Interface Circuits

The interface circuit between the MCU in the E6000 emulator and the user system has a signal delay of about 3 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 circuits of the interface signals:

Vcc:

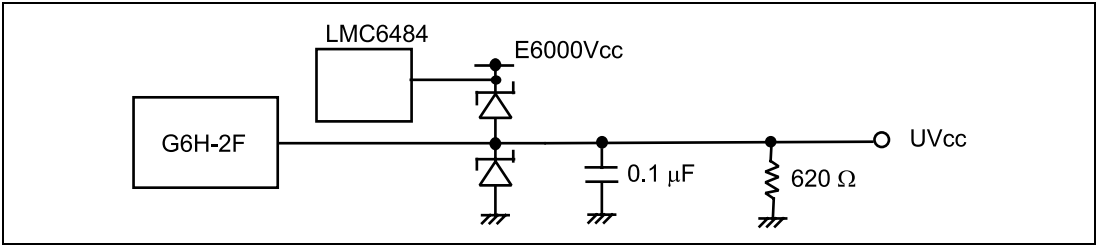


Figure 3.7 User System Interface Signal Circuit

CLK:

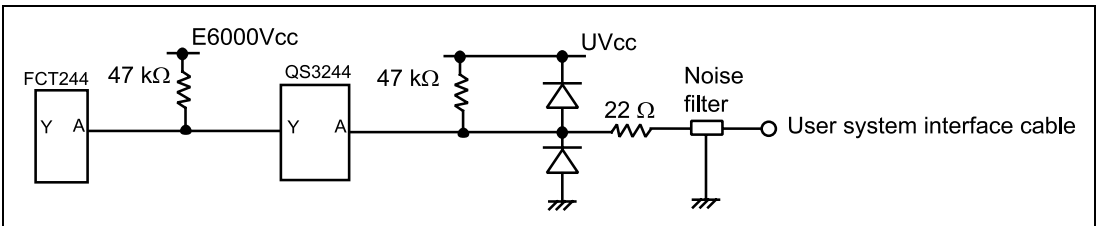


Figure 3.8 User System Interface Circuit for CLK

RESET:

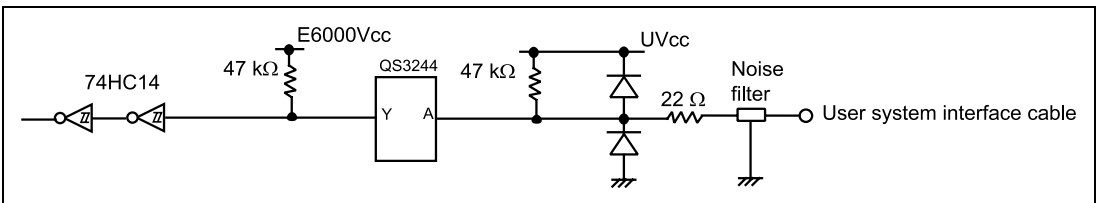


Figure 3.9 User System Interface Circuit for RESET

I/O-1/IRQ and I/O-2/IRQ:

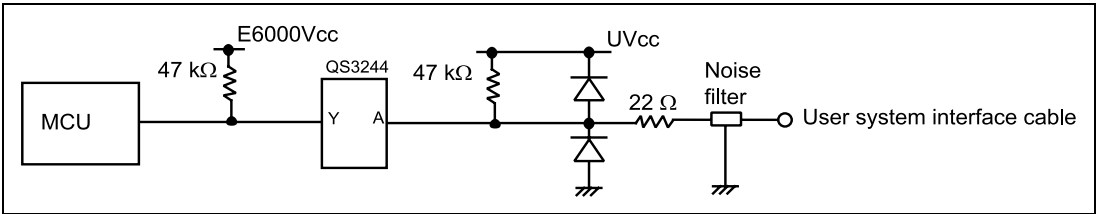


Figure 3.10 User System Interface Circuit for I/O-1/IRQ and I/O-2/IRQ

La and Lb:

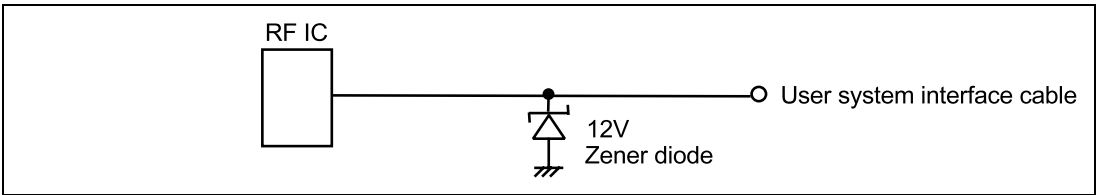


Figure 3.11 User System Interface Circuit for La and Lb

3.3.3 External Probes/Trigger Output 1

An 8-pin connector, marked EXT1 (at the lower right of 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 3.12.

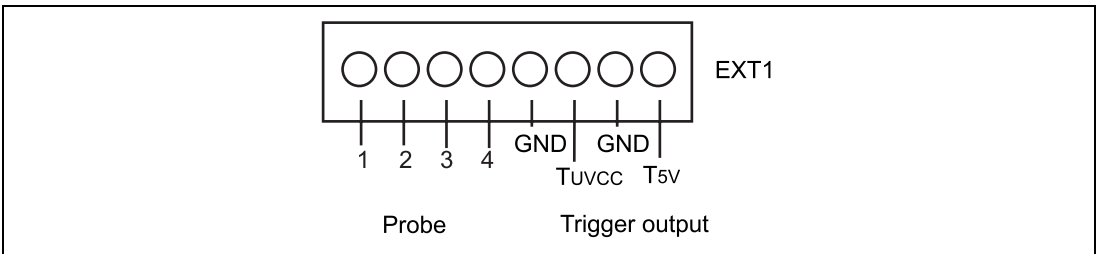


Figure 3.12 Probe1 and Trigger Connector

The external probe interface circuit is shown in figure 3.13.

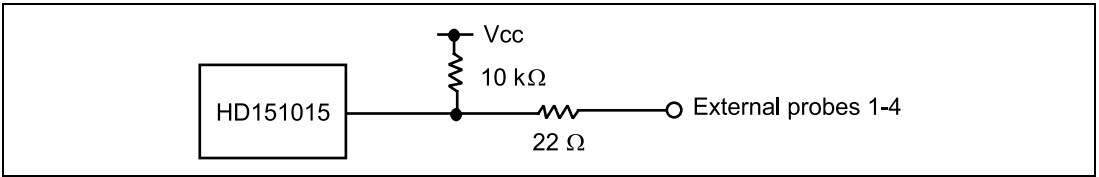


Figure 3.13 External Probe1 Interface Circuit

The trigger output is a low-level signal and is controlled by event channel 8. The trigger output is available as either T5V (within the range from 2.5 V to 5 V; does not depend on the user Vcc level) or TUvcc (the user system supply voltage level).

3.3.4 External Probe 2 (EXT2)/Trigger Output

A 6-pin connector, marked EXT2 (at the lower right of 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 3.14.

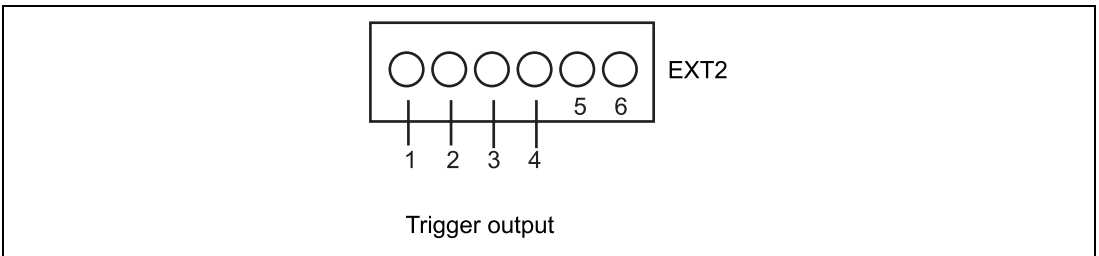


Figure 3.14 Connector for External Probe 2

The trigger output is a high-level signal and is output during read or write cycles when a trigger which condition (1 to 4) of the bus monitor function is satisfied. The trigger output is available as Vcc (user system power voltage level).

3.3.5 Power Supply Circuit

CAUTION

When the user system is connected to the E6000 emulator, start up the E6000 emulator and then turn on/off the user system.

Failure to do so will result in a FIRE HAZARD and will damage the emulator product and the host computer.

The MCU in the E6000 emulator always operates at 5 V E6000 emulator power supply. This means that no power is taken from the user system. The circuit interfacing the MCU and the user system includes a voltage level shifter to support operation at a low voltage. Therefore, note that even if the user system operates at a voltage lower than 3 V, the MCU in the E6000 emulator can operate at the maximum operating frequency 5MHz (Input CLK frequency 10MHz).

You can set a user low V_{cc} threshold in the range 5 V – 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 **System Status** window will display **Down** and Reset is input, otherwise **OK** is displayed. When user system interface cable is not connected, this function does not work.

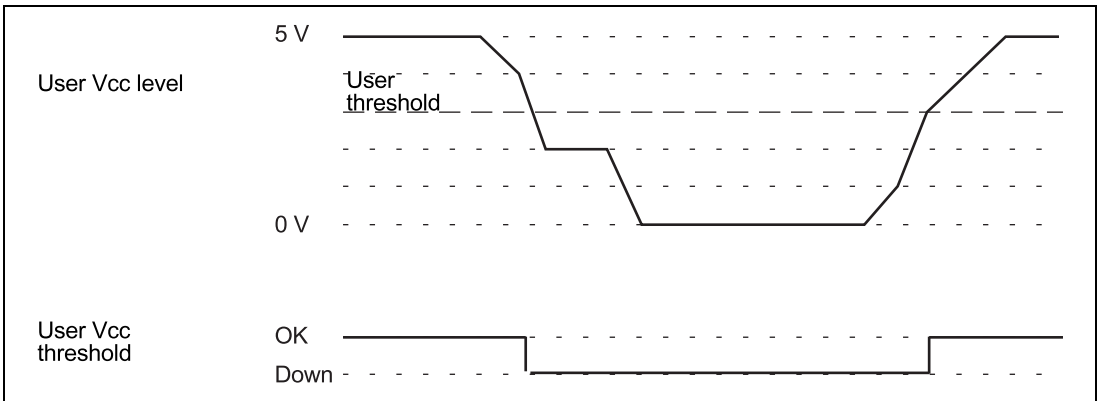


Figure 3.15 Relationship of Vcc between the User System and E6000 Emulator

3.4 Jumper Pin Settings

Do not change the settings of jumper pin JP1 if removing the cover of the emulator. Keep pins 1 and 2 closed.

3.5 Differences between MCU and E6000 Emulator

3.5.1 Registers

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	MCU	E6000 Emulator
Power-on	PC	Reset vector value	Reset vector value
	ER0 to ER6	Undefined	H'00000000
	ER7 (SP)	Undefined	H'00000010
	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)	Undefined	H'00000010
	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 section 3.3.2, User System Interface Circuits, for details of the protection circuit used on the I/O ports of the E6000 emulator.

3.5.2 EEPROM

Table 3.2 EEPROM Differences between MCU and E6000 Emulator

Item	MCU	E6000 Emulator
Programming time	4 ms (max) in normal mode/2 ms (max) in fast mode	Always MCU max time.
Overwrite time	2 ms (max) in normal mode/1 ms (max) in fast mode	Always MCU max time.
Erase time	2 ms (max) in normal mode/1 ms (max) in fast mode	Always MCU max time.
Setting the registers	MCU limits the setting as follows. Source address(ER5):RAM Destination address(ER6):EEPROM Byte count(R4L):up to page size If the setting is wrong, the result of EEPMOV is not guaranteed.	If the setting wrong, the result of EEPMOV is not guaranteed and the result is different from the one of MCU.

3.5.3 WDT

Table 3.3 WDT Differences between MCU and E6000 Emulator

Item	MCU	E6000 Emulator
EWE interrupt	The interrupt processing starts after instruction 1) is executed, in the following example: 1) MOV.B R0L,@ECR 2) EEPMOV.B	If a break is specified or single-step execution is performed for the example instructions on the left, the EWE interrupt timing in the E6000 emulator differs from that in the MCU.
UDF interrupt	No UDF interrupt processing is performed between the RTE instruction at the end of the EWE interrupt processing and the instruction following the RTE instruction.	If a break is specified or single-step execution is performed for the RTE instruction described on the left, a UDF interrupt will occur.

3.5.4 SYSCR Register

Table 3.4 SYSCR Register Differences between MCU and E6000 Emulator

Item	MCU	E6000 Emulator
CRES bit	Set to 1 at cold reset.	The CRES bit is always set to 1 when the user system interface cable is not connected. When the user system interface cable is connected, high or low of the CRES bit is determined by the User Vcc Threshold power supply level of the user power supply voltage monitoring function of the emulator.

3.5.5 Security

Table 3.5 Security Differences between MCU and E6000 Emulator

Item	MCU	E6000 Emulator
LVD	The MCU is reset if low voltage is detected.	The emulator has a circuit for detecting drops in Vcc, but true LVD is not supported (the emulator only provides pseudo-LVD)
HVD	The MCU is reset if a high voltage is detected.	Does not support high-voltage detecting function.
LFD	The MCU is reset if a low frequency CLK is detected.	Supports low-frequency CLK detecting function.
HFD	The MCU is reset if a high frequency CLK is detected.	Does not support high-frequency CLK detecting function.

3.5.6 Contactless Operation

Table 3.6 Differences in Contactless Operation

Item	MCU	E6000 Emulator
Characteristics of antenna input pin	Refer the MCU hardware manual.	Different from those of MCU
Switching contact/contactless (CCLMD bit)	The results of MCU self-judgment are reflected.	Set with the Communication mode item in the Configuration Properties window.
Residual multiplication coprocessor operation register (CPRMD bit)	The results of MCU self-judgment are reflected.	Set with the Active Coprocessor item in the Configuration Properties window.
Coprocessor operation in contactless mode	Coprocessor operation is not allowed with CPRMD = 1.	Coprocessor operation is allowed even with CPRMD = 1.

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 AE-4 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 and Programming 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 Sysrof format (`Tutorial.abs`) are provided in the HDI installation disk.

4.2 Starting HDI

To start the HDI:

- Select **HDI for E6000 AE-4** from the **Start** menu.

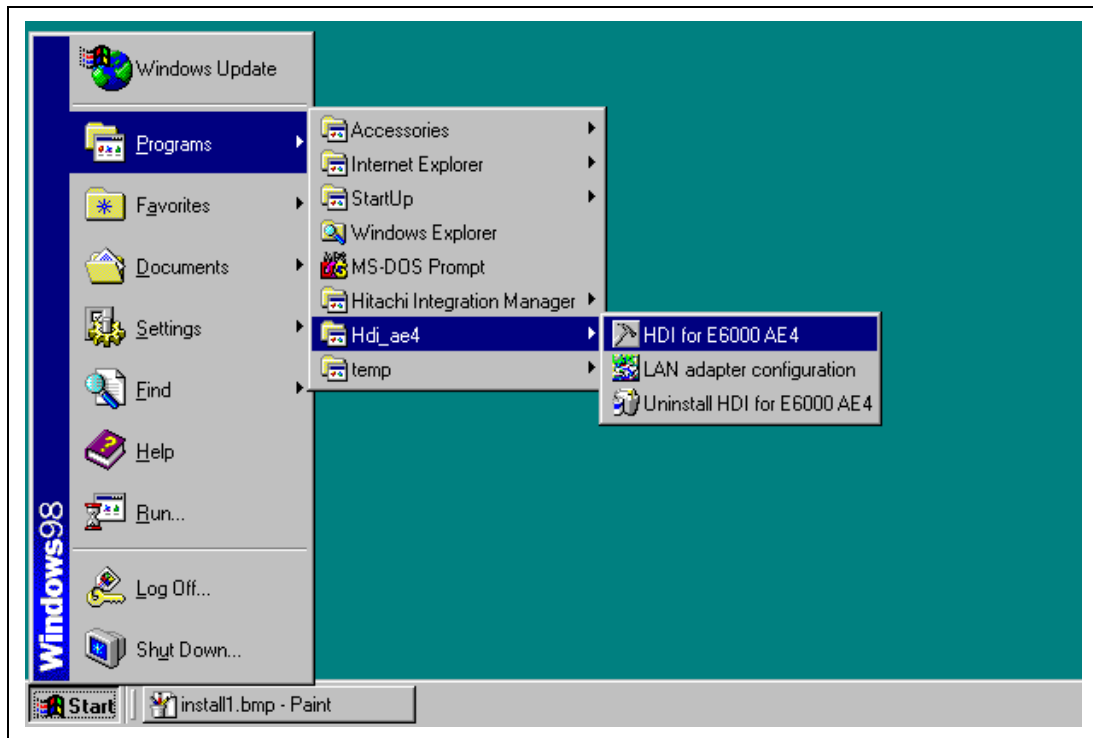


Figure 4.1 HDI Start Menu

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.

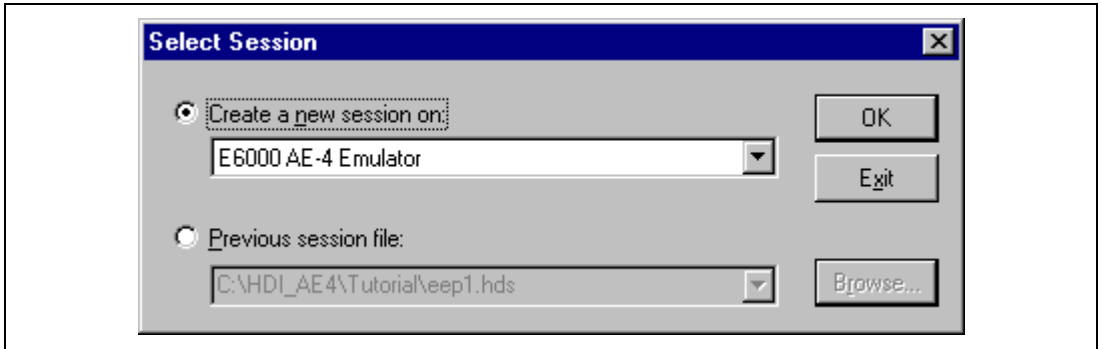


Figure 4.2 Select Platform Dialog Box

- For this tutorial select E6000 AE-4 Emulator and click **OK** to continue.

Note that you can change the target platform at any time by choosing **New Session...** from the **File** menu.

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.

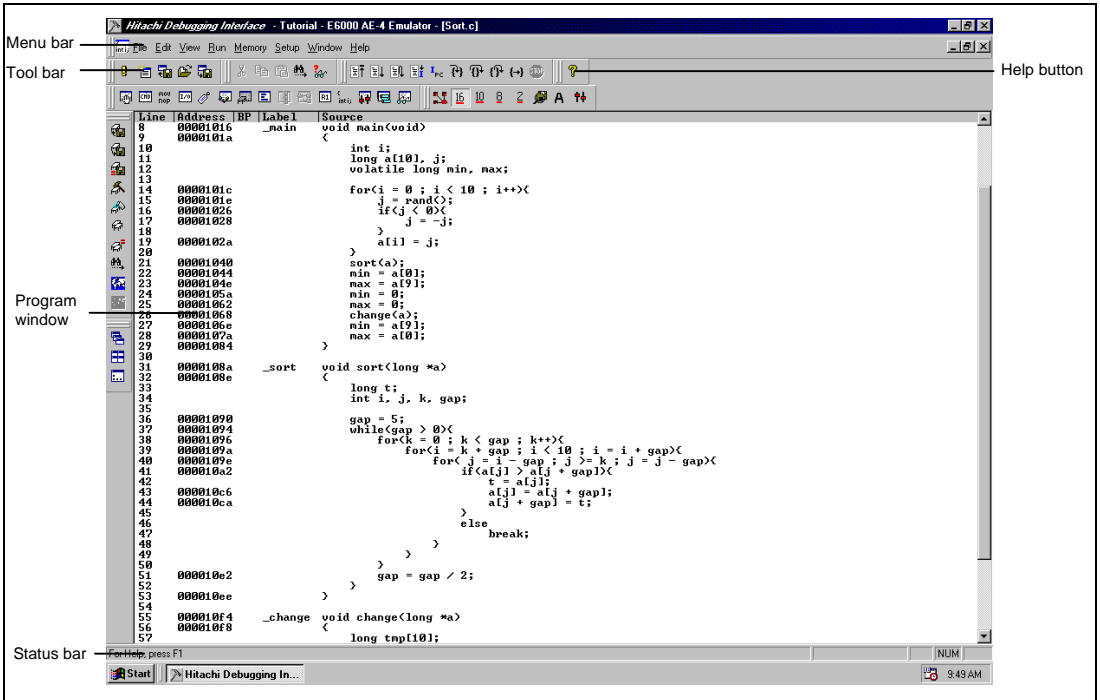


Figure 4.3 Hitachi Debugging Interface Window

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 device option
- The clock source
- Communications mode
- 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:

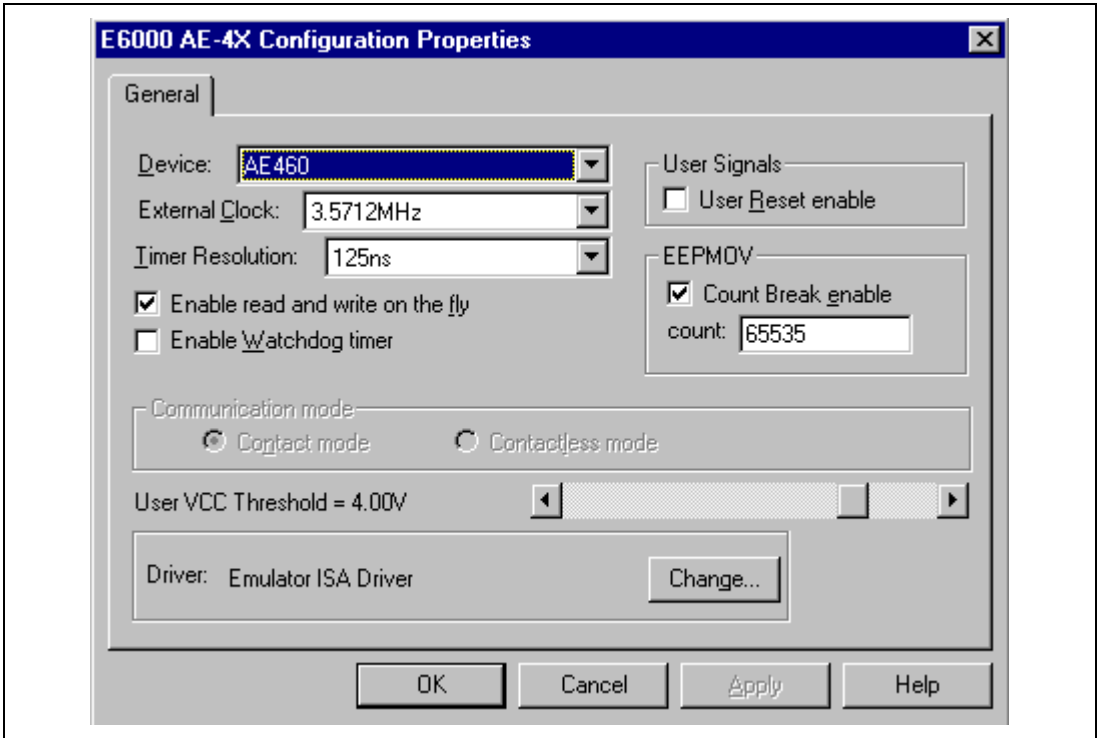


Figure 4.4 Configuration Dialog Box

- Set up the options as shown in table 4.1.

Table 4.1 Configuration Options

Option	Value (Depending on Evaluation Chip)
Device	AE460
Clock	3.5712 MHz
Timer resolution	125 ns
User system voltage monitoring level (User LowVCC threshold)	4.00 V
Driver	E6000 ISA Driver
Watchdog timer and user reset	Disabled
All other options	Enabled
Communication mode	Contact mode
Break count	65535

- 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.

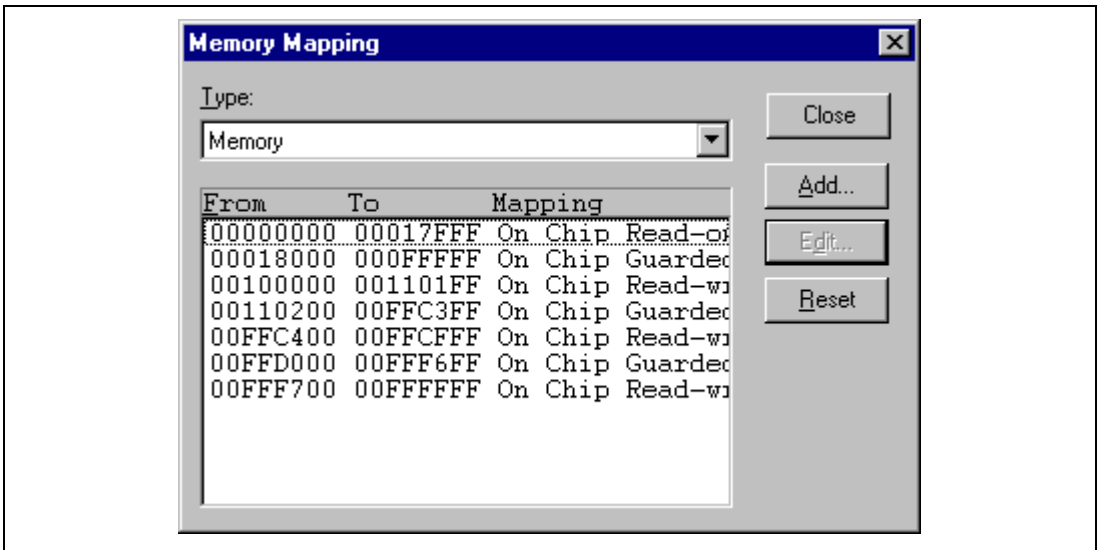


Figure 4.5 Memory Mapping Dialog Box

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
Emulator	Accesses the emulation memory

Note) "User" can not be set on this emulator.

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.

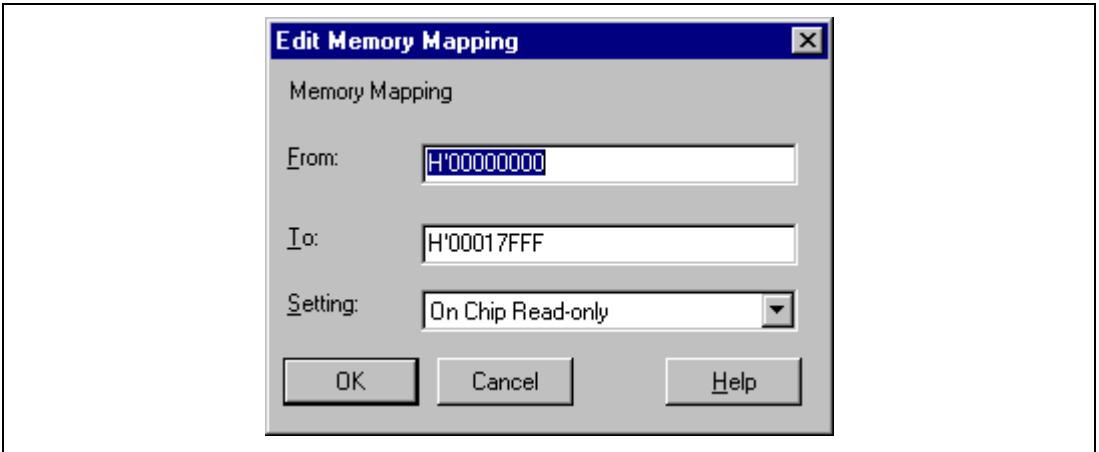


Figure 4.6 Edit Memory Mapping Dialog Box

- 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:



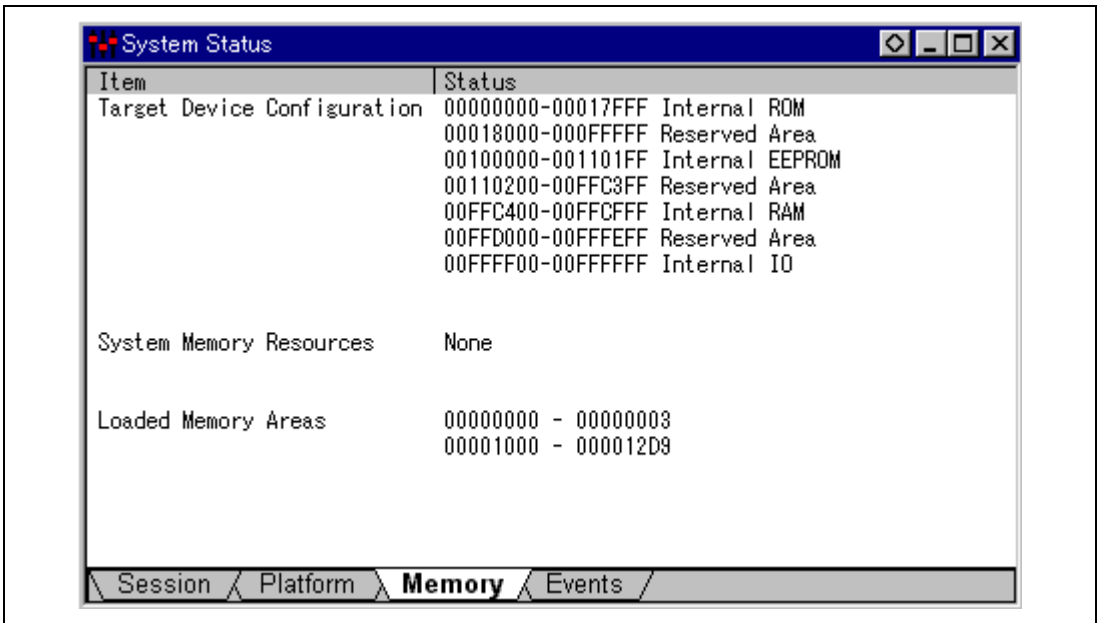


Figure 4.7 System Status Window (Memory Sheet)

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

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 Sysprof-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.

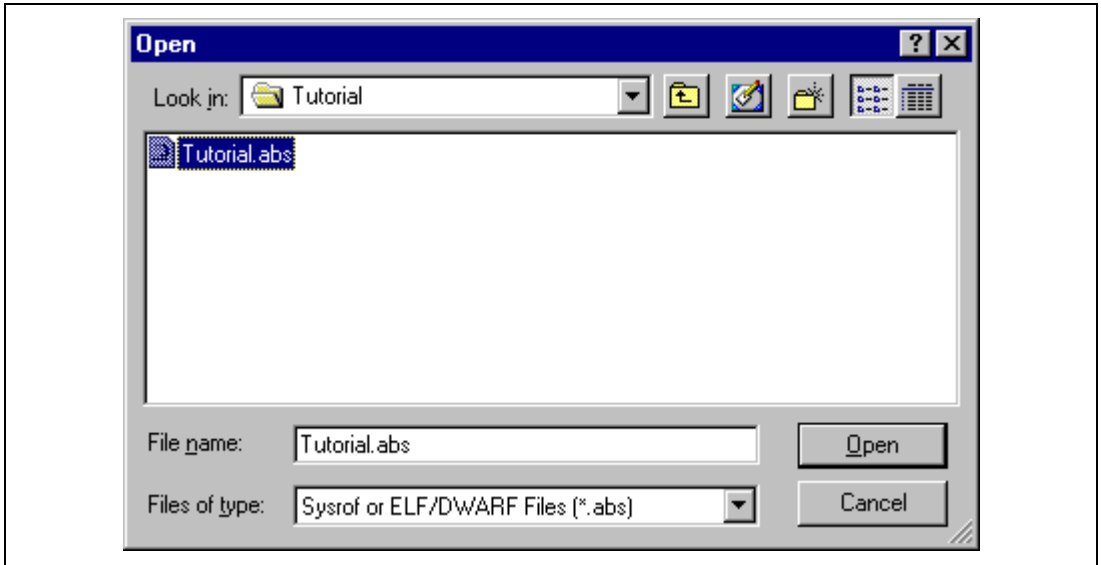


Figure 4.8 Open Dialog Box (Object File Selection)

When a 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.

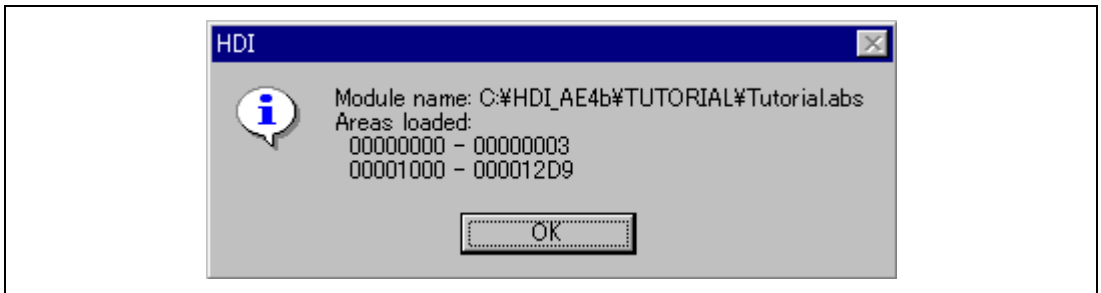


Figure 4.9 HDI Dialog Box

- 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.



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

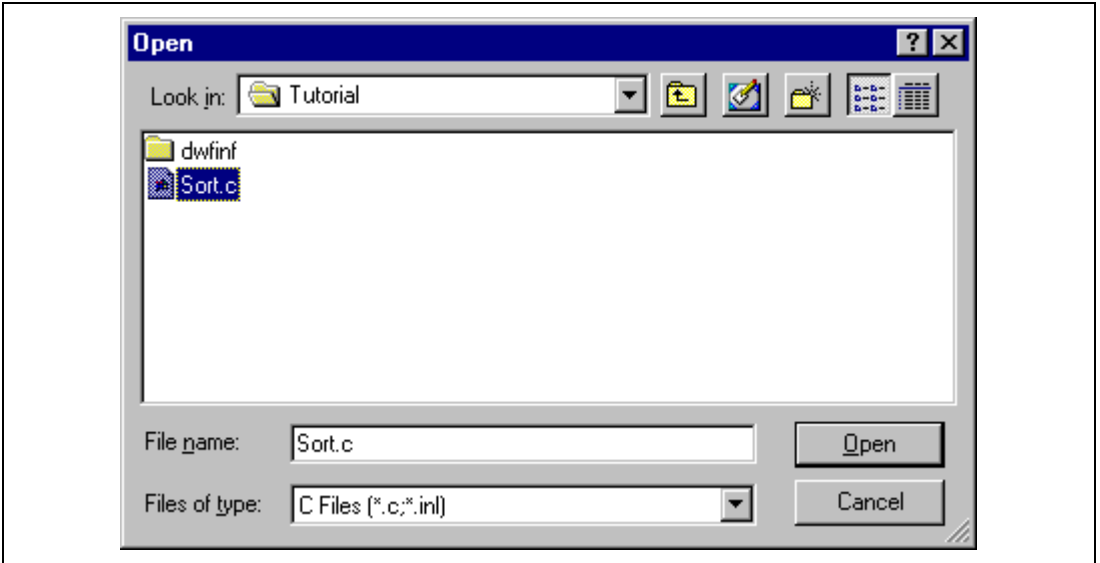


Figure 4.10 Open Dialog Box (Source File Selection)

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

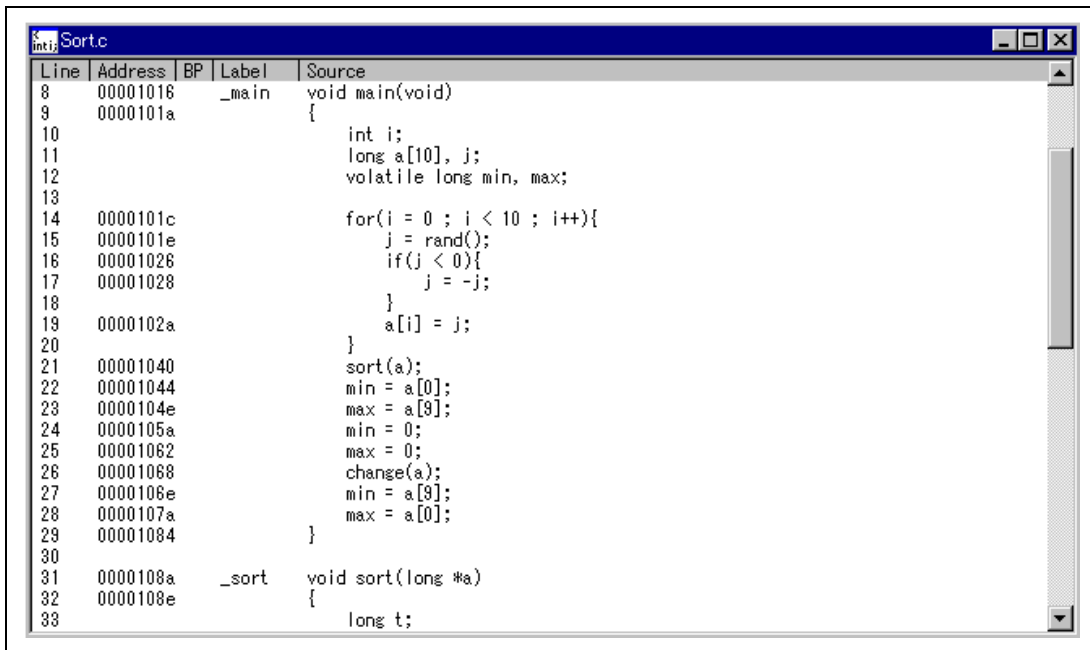


Figure 4.11 Source Program Window

- 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'1040 as follows:

- Double-click in the **BP** column on the line containing address H'1040.

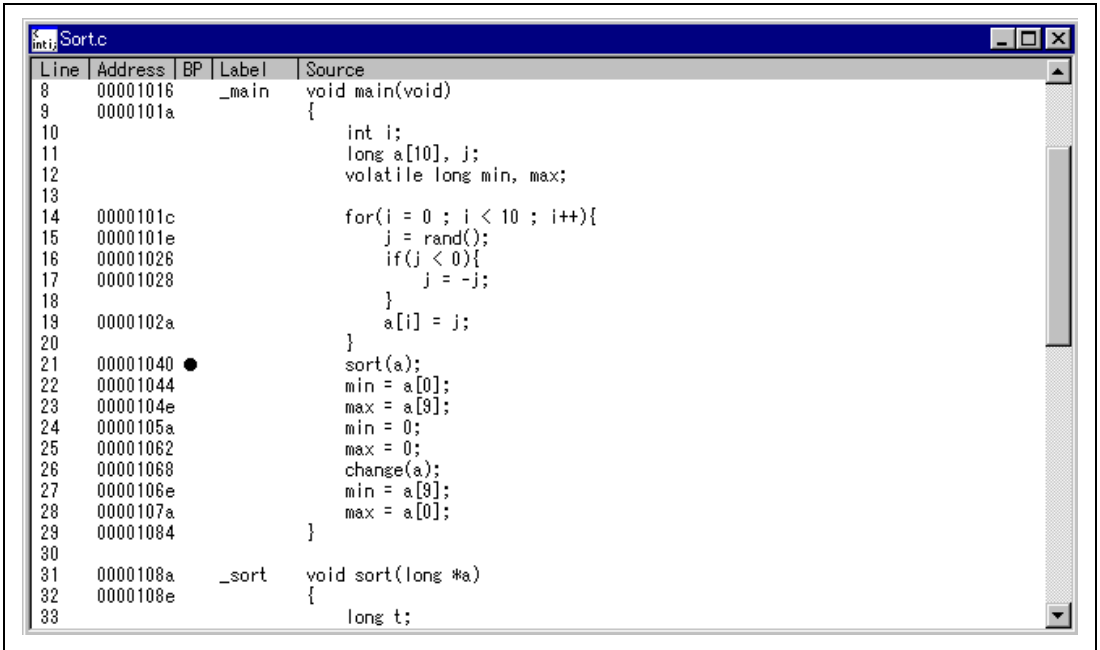


Figure 4.12 Setting a Breakpoint (PC Break)

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) →
 ...

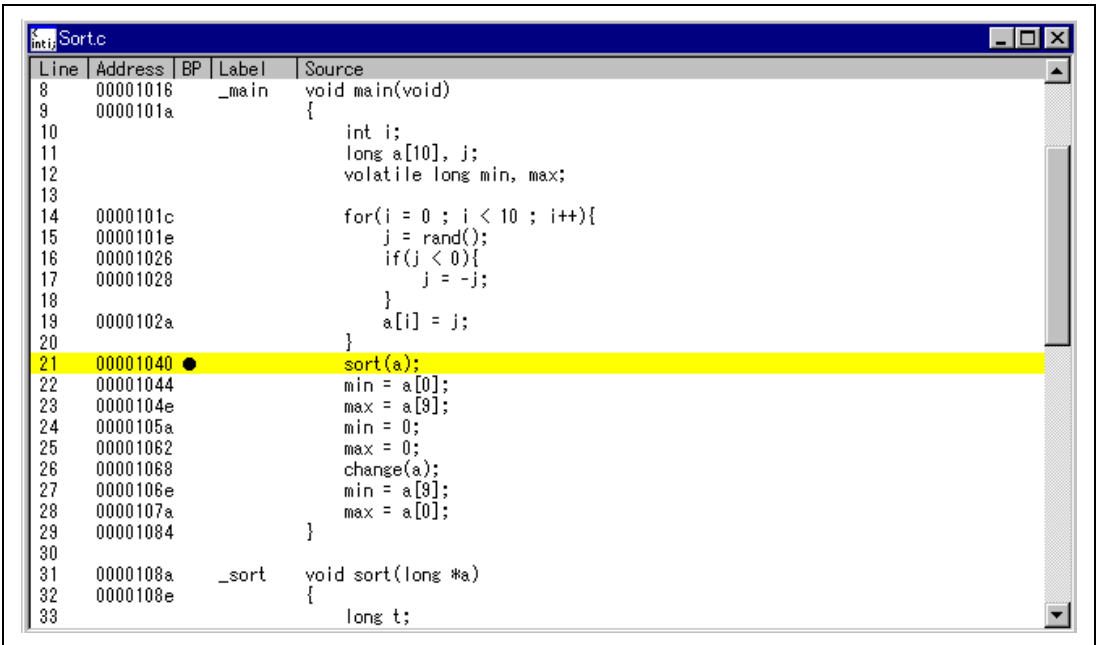
4.5.2 Executing the Program

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.

A screenshot of a debugger window titled 'Sort.c'. The window displays a table of source code with columns for Line, Address, BP, Label, and Source. Line 21 is highlighted in yellow, and a black dot is visible in the BP column for that line. The code includes a main function and a sort function.

Line	Address	BP	Label	Source
8	00001016		_main	void main(void)
9	0000101a			{
10				int i;
11				long a[10], j;
12				volatile long min, max;
13				
14	0000101c			for(i = 0 ; i < 10 ; i++){
15	0000101e			j = rand();
16	00001026			if(j < 0){
17	00001028			j = -j;
18				}
19	0000102a			a[i] = j;
20				}
21	00001040	●		sort(a);
22	00001044			min = a[0];
23	0000104e			max = a[9];
24	0000105a			min = 0;
25	00001062			max = 0;
26	00001068			change(a);
27	0000106e			min = a[9];
28	0000107a			max = a[0];
29	00001084			}
30				
31	0000108a		_sort	void sort(long *a)
32	0000108e			{
33				long t;

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.



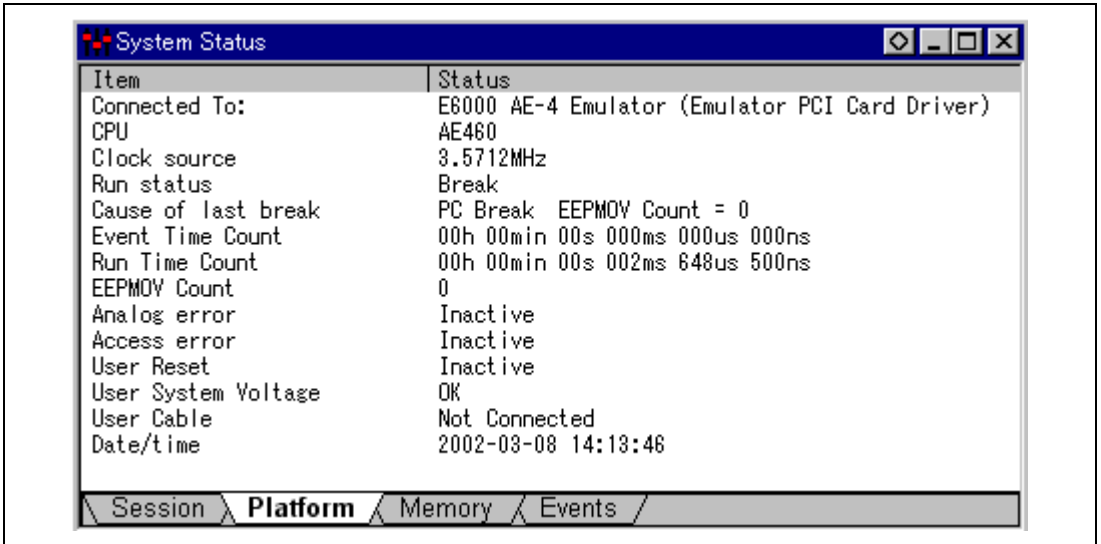


Figure 4.14 System Status Window (Platform Sheet)

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 2648.500 μ s. 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:

A screenshot of the 'Registers' window in a debugger. The window title is 'R1 Registers'. It contains a table with two columns: 'Register' and 'Value'. The registers listed are ER0 through ER7, PC, and CCR. The PC register has a value of 001040. The CCR register has a value of I0---Z--. The PC register is highlighted in blue.

Register	Value
ER0	00FFCFF0
ER1	00000FF6
ER2	00000000
ER3	BF54BC7E
ER4	0000000A
ER5	00FFCFCC
ER6	00000FF6
ER7	00FFCFCC
PC	001040
+ CCR	I0---Z--

Figure 4.15 Registers Window

As expected, the value of the program counter, PC, is the same as the highlighted statement, H' 1040.

(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.

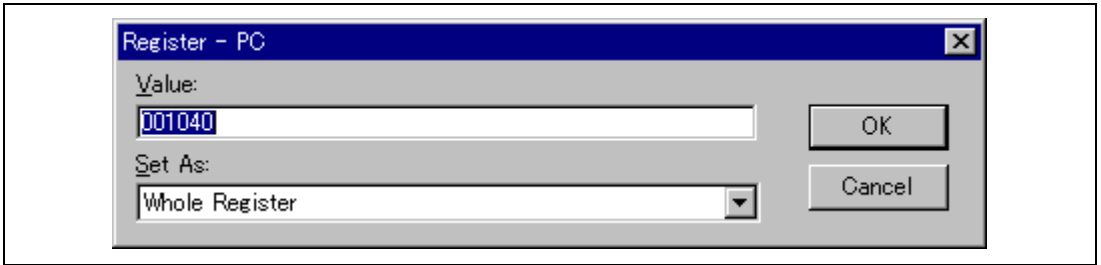


Figure 4.16 Register Dialog Box

- 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:

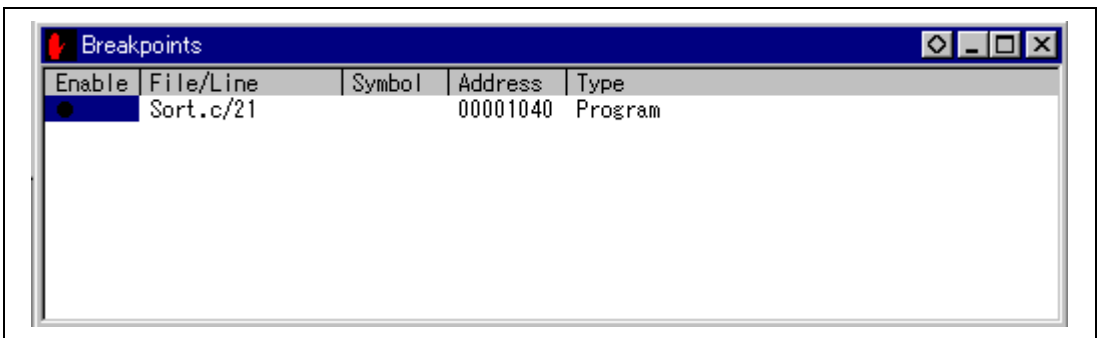


Figure 4.17 Breakpoints Window

The **Breakpoints** window also allows you to enable or disable breakpoints, define new breakpoints, and delete breakpoints by pop-up menu.

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 `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`.

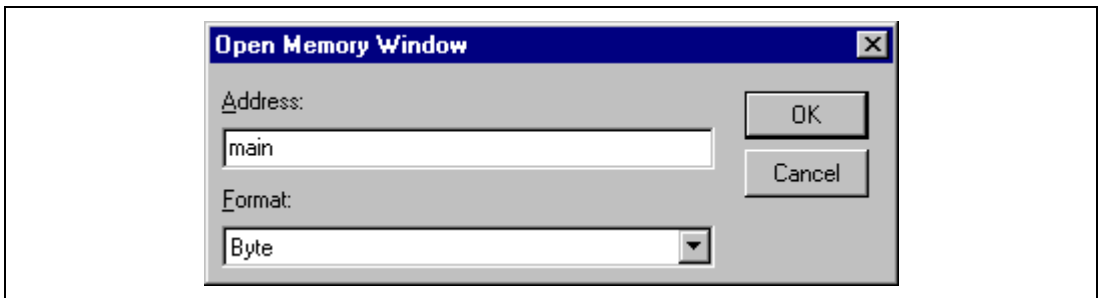


Figure 4.18 Open Memory Window Dialog Box

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

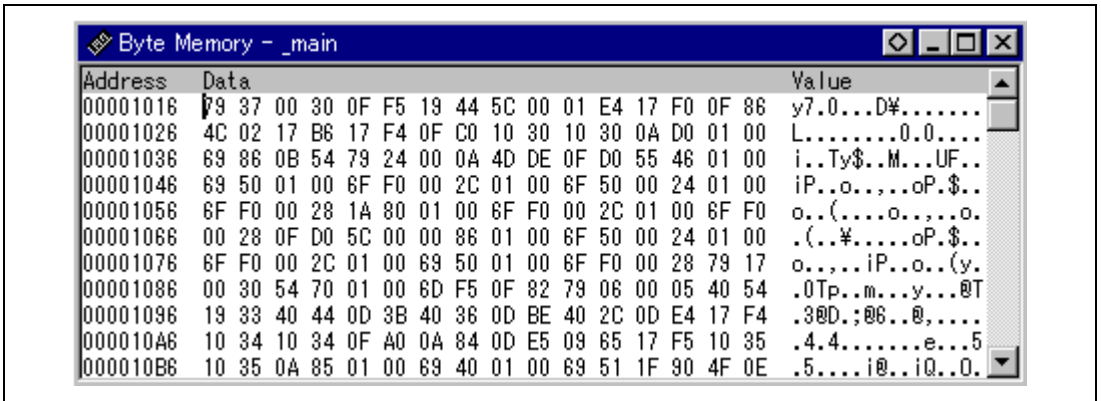


Figure 4.19 Memory Window (Byte)

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.

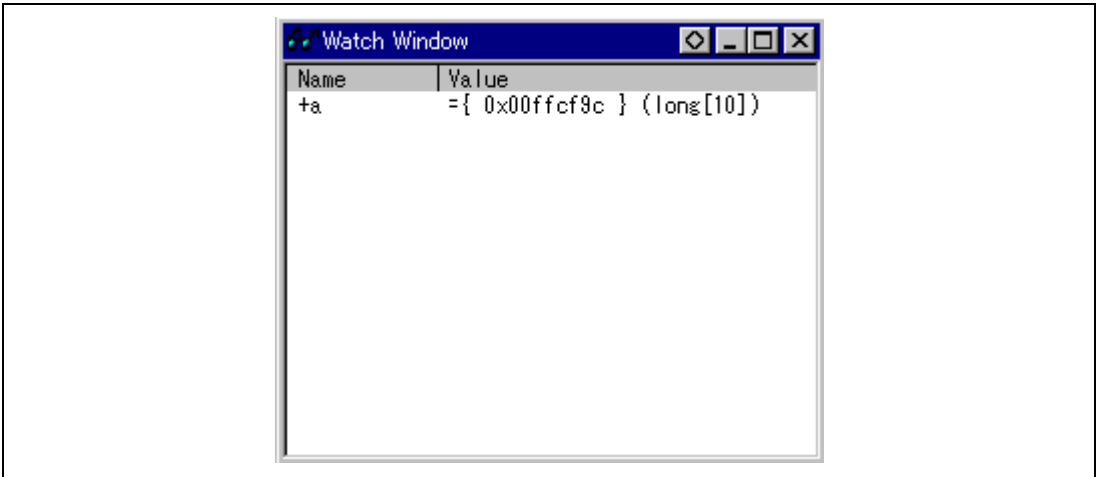


Figure 4.20 Watch Window (After Adding Variables)

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.



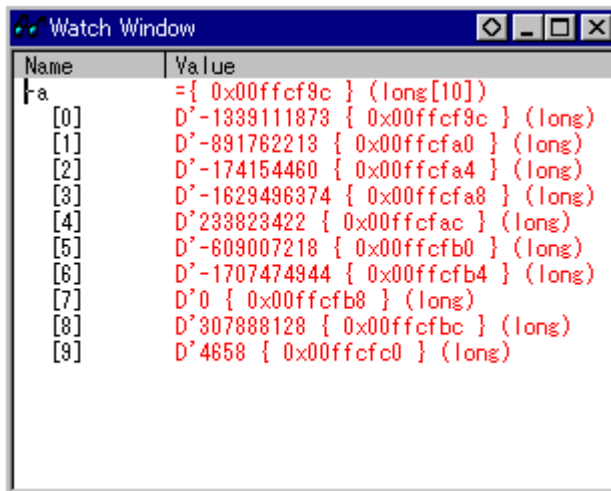


Figure 4.21 Watch Window (Symbol Expansion)

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.

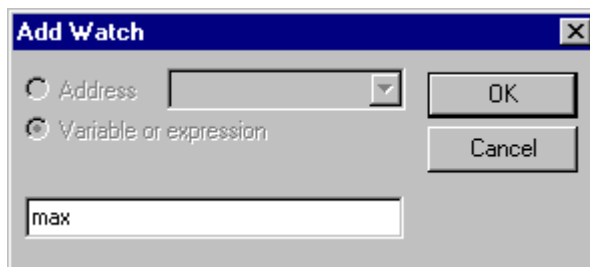


Figure 4.22 Add Watch Dialog Box

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

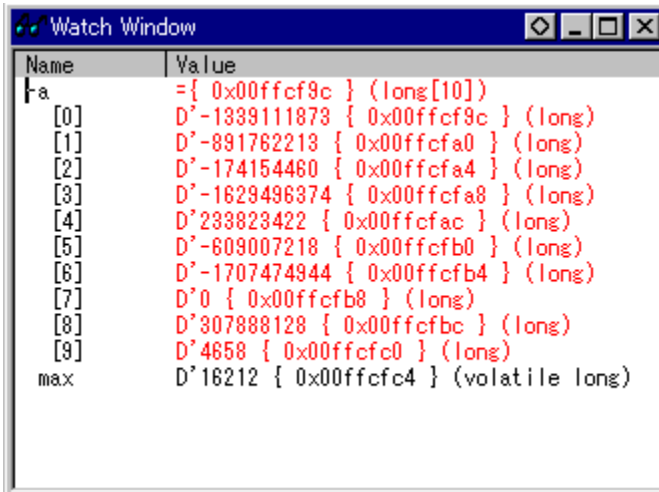


Figure 4.23 Watch Window (Adding Variables)

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

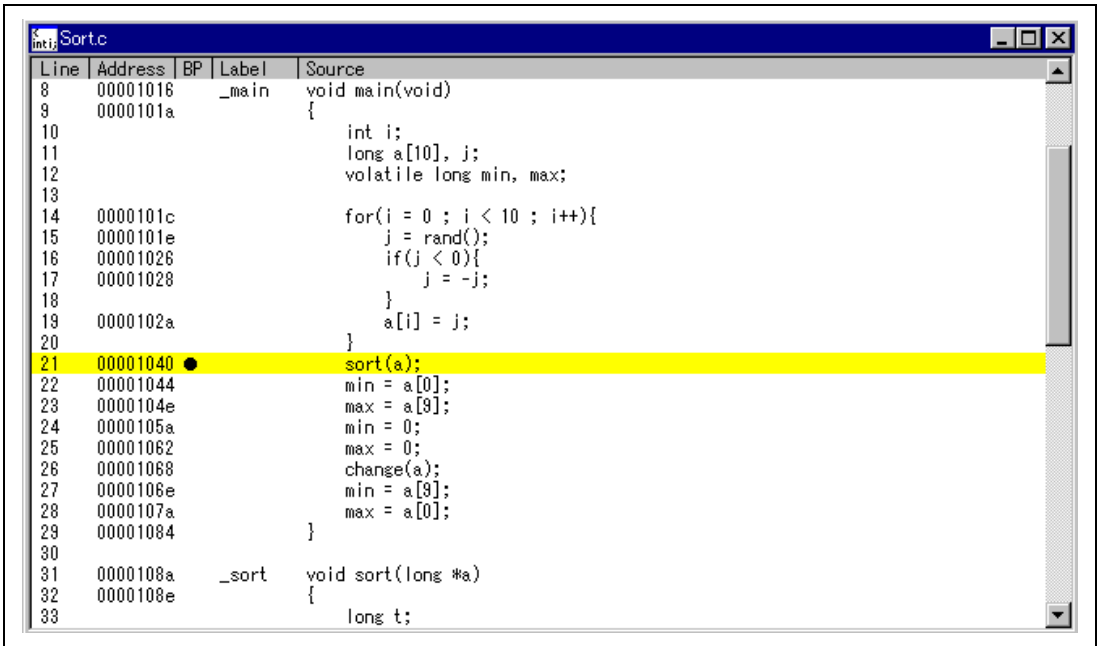
Command	Description
Step in	Executes every statement, including statements within functions.
Step Over	Executes a function call in a single step.
Step out	Exits a function and stops at the next statement of the calling program.
Step...	Allows you to step repeatedly the specified number of times.

4.7.1 Single Stepping

- Confirm that a PC break is set at H' 1040.
- Select **Reset Go** from the **Run** menu or click the **Reset Go** button in the toolbar.



The program is executed and stopped at **H'107a** by set **PC break**. The statement of **sort (a)** ; will be highlighted.



Line	Address	BP	Label	Source
8	00001016		_main	void main(void)
9	0000101a			{
10				int i;
11				long a[10], j;
12				volatile long min, max;
13				
14	0000101c			for(i = 0 ; i < 10 ; i++){
15	0000101e			j = rand();
16	00001026			if(j < 0){
17	00001028			j = -j;
18				}
19	0000102a			a[i] = j;
20				}
21	00001040	●		sort(a);
22	00001044			min = a[0];
23	0000104e			max = a[9];
24	0000105a			min = 0;
25	00001062			max = 0;
26	00001068			change(a);
27	0000106e			min = a[9];
28	0000107a			max = a[0];
29	00001084			}
30				
31	0000108a		_sort	void sort(long *a)
32	0000108e			{
33				long t;

Figure 4.24 Program Window after Executing the Reset Go Command

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



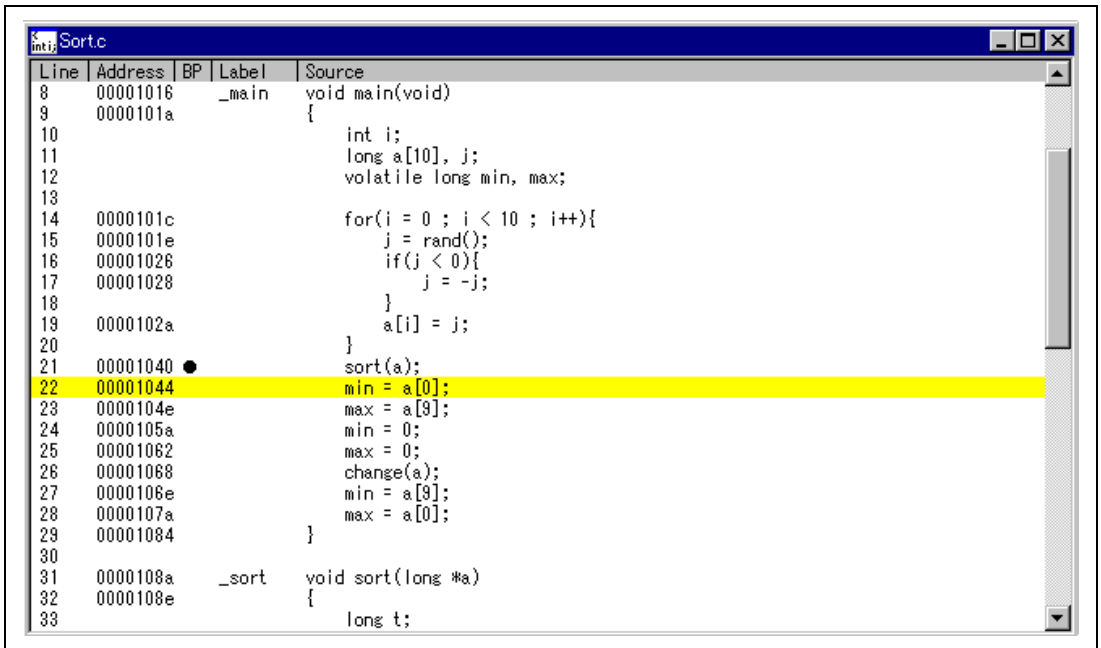
Line	Address	BP	Label	Source
8	00001016		_main	void main(void)
9	0000101a			{
10				int i;
11				long a[10], j;
12				volatile long min, max;
13				
14	0000101c			for(i = 0 ; i < 10 ; i++){
15	0000101e			j = rand();
16	00001026			if(j < 0){
17	00001028			j = -j;
18				}
19	0000102a			a[i] = j;
20				}
21	00001040			sort(a);
22	00001044			min = a[0];
23	0000104e			max = a[9];
24	0000105a			min = 0;
25	00001062			max = 0;
26	00001068			change(a);
27	0000106e			min = a[9];
28	0000107a			max = a[0];
29	00001084			}
30				
31	0000108a		_sort	void sort(long *a)
32	0000108e			{
33				long t;

Figure 4.25 Program Window after Executing the Step In Command (1)

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.



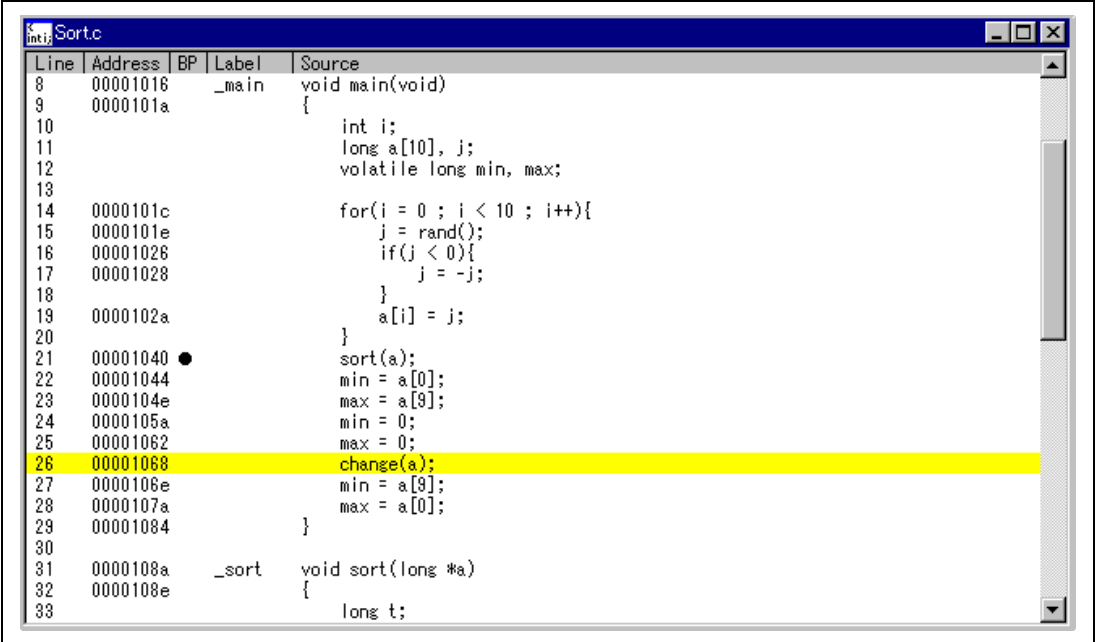
Address H' 1044 will be highlighted showing that the emulator has exit from the function.



```
intj Sort.c
Line Address BP Label Source
8 00001016 _main void main(void)
9 0000101a {
10     int i;
11     long a[10], j;
12     volatile long min, max;
13
14     for(i = 0 ; i < 10 ; i++){
15         j = rand();
16         if(j < 0){
17             j = -j;
18         }
19         a[i] = j;
20     }
21     sort(a);
22     min = a[0];
23     max = a[9];
24     min = 0;
25     max = 0;
26     change(a);
27     min = a[9];
28     max = a[0];
29 }
30
31 0000108a _sort void sort(long *a)
32 0000108e {
33     long t;
```

Figure 4.26 Program Window after Executing the Step Out Command

- Choose **Step In** four times from the **RUN** menu, or click the **Step In** button four times to call the change function.



The screenshot shows a debugger window titled "Sort.c" with a table of source code. The table has four columns: Line, Address, BP, and Source. Line 26, which contains the call to "change(a);", is highlighted in yellow. A black dot is visible in the BP column for line 21. The code includes a main function and a sort function.

Line	Address	BP	Label	Source
8	00001016		_main	void main(void)
9	0000101a			{
10				int i;
11				long a[10], j;
12				volatile long min, max;
13				
14	0000101c			for(i = 0 ; i < 10 ; i++){
15	0000101e			j = rand();
16	00001026			if(j < 0){
17	00001028			j = -j;
18				}
19	0000102a			a[i] = j;
20				}
21	00001040			sort(a);
22	00001044			min = a[0];
23	0000104e			max = a[9];
24	0000105a			min = 0;
25	00001062			max = 0;
26	00001068			change(a);
27	0000106e			min = a[9];
28	0000107a			max = a[0];
29	00001084			}
30				
31	0000108a		_sort	void sort(long *a)
32	0000108e			{
33				long t;

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** three times from the **Run** menu, or click the **Step Over** button three times in the toolbar.



The program executes the change function and stops at the beginning of the address, H'10ac.

Line	Address	BP	Label	Source
8	00001016		_main	void main(void)
9	0000101a			{
10				int i;
11				long a[10], j;
12				volatile long min, max;
13				
14	0000101c			for(i = 0 ; i < 10 ; i++){
15	0000101e			j = rand();
16	00001026			if(j < 0){
17	00001028			j = -j;
18				}
19	0000102a			a[i] = j;
20				}
21	00001040	●		sort(a);
22	00001044			min = a[0];
23	0000104e			max = a[9];
24	0000105a			min = 0;
25	00001062			max = 0;
26	00001068			change(a);
27	0000106e			min = a[9];
28	0000107a			max = a[0];
29	00001084			}
30				
31	0000108a		_sort	void sort(long *a)
32	0000108e			{
33				long t;

Figure 4.28 Program Window after Executing the Step Over Command

4.7.3 Displaying Local Variables

For example, we will examine the local variables in the function `sort`. 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.

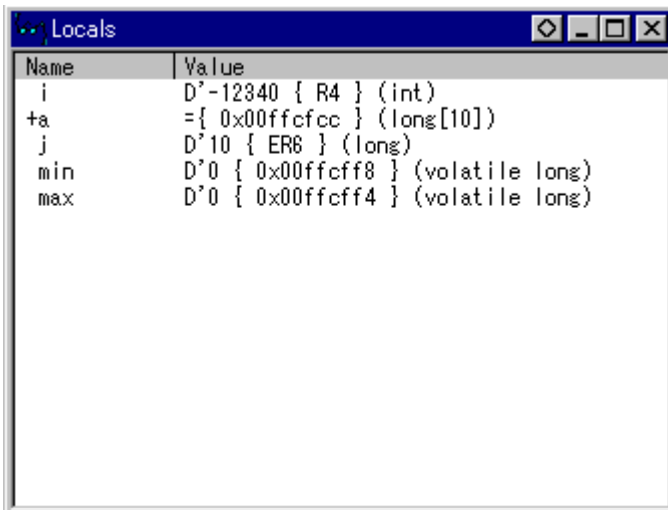


Figure 4.29 Locals Window

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 a, j, i, min and max are changed and their values are displayed.

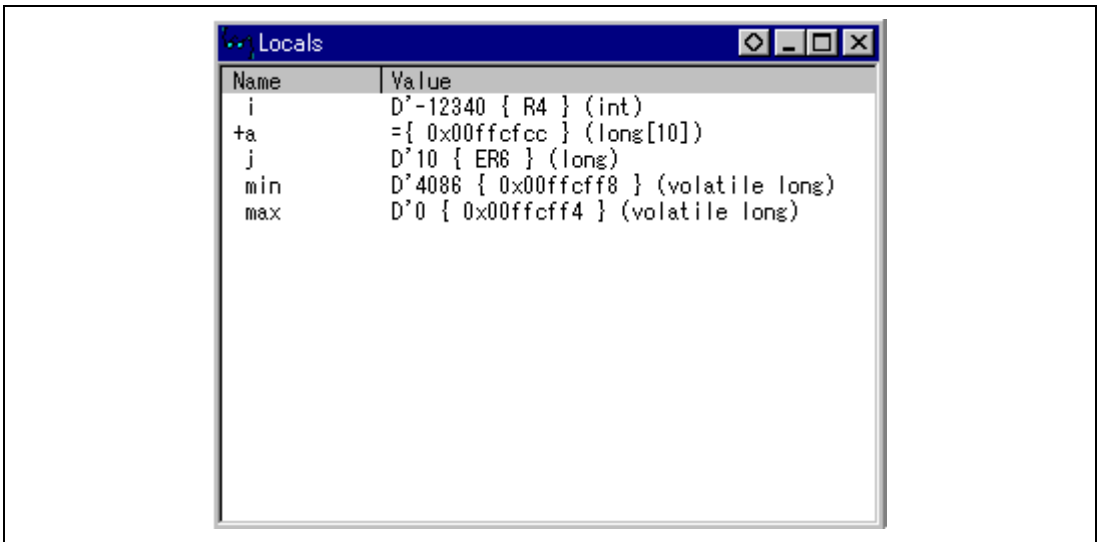


Figure 4.30 Locals Window (After Contents of Variable min are Changed)

- Double click the + symbol in front of array a in the **Locals** window to display the separate elements of array a.
- Refer to the elements of array a before executing the **sort** function, and confirm that random data is sorted in descending order.

Name	Value
i	D'-12340 { R4 } (int)
-a	= { 0x00ffcfc0 } (long[10])
[0]	D'31051 { 0x00ffcfc0 } (long)
[1]	D'23010 { 0x00ffcfd0 } (long)
[2]	D'17515 { 0x00ffcfd4 } (long)
[3]	D'16838 { 0x00ffcfd8 } (long)
[4]	D'16212 { 0x00ffcfdc } (long)
[5]	D'10113 { 0x00ffcfe0 } (long)
[6]	D'7419 { 0x00ffcfe4 } (long)
[7]	D'5758 { 0x00ffcfe8 } (long)
[8]	D'5627 { 0x00fcfec } (long)
[9]	D'4086 { 0x00ffcff0 } (long)
j	D'10 { ER6 } (long)
min	D'4086 { 0x00ffcff8 } (volatile long)
max	D'0 { 0x00ffcff4 } (volatile long)

Figure 4.31 Locals Window (Elements of Array Variable a after Function sort has been Executed)

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 timing when a program accesses address H'10ca.

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:

- 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.

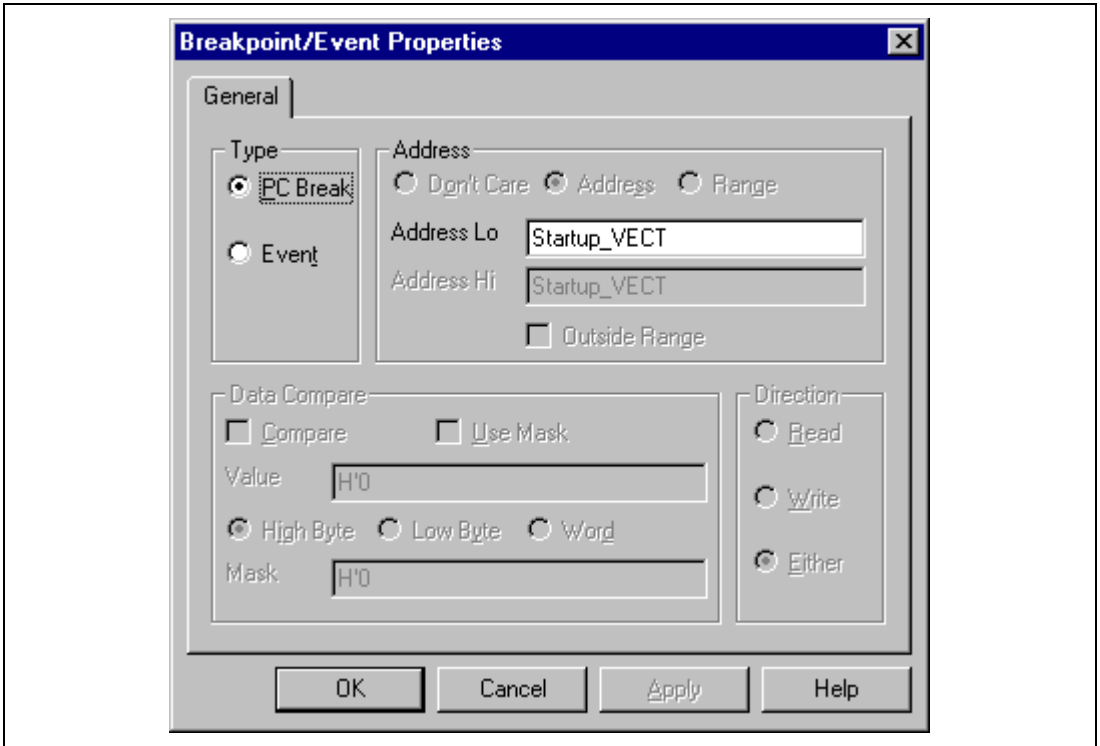


Figure 4.32 Breakpoint/Event Properties Dialog Box

- Set the **Type** to **Event** and enter the address H'11e8 into the **Address Lo** box as a condition.
- Click **OK** to define the breakpoint.

This will cause a break whenever address H'10ca is accessed, either for a read or a write.

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

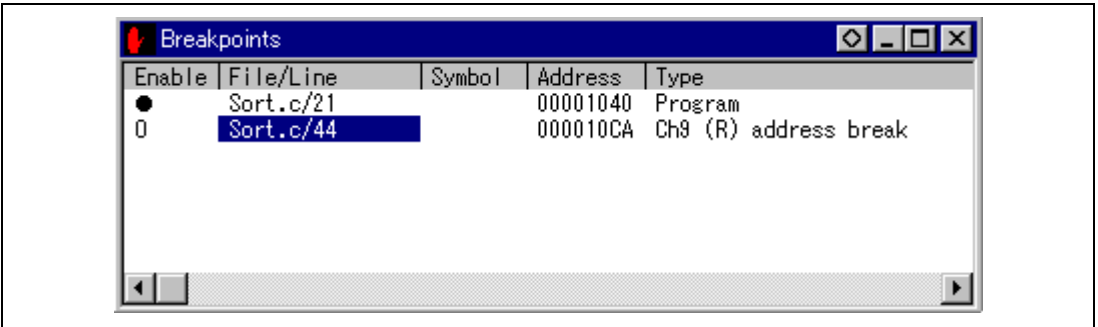


Figure 4.33 Breakpoints Window

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

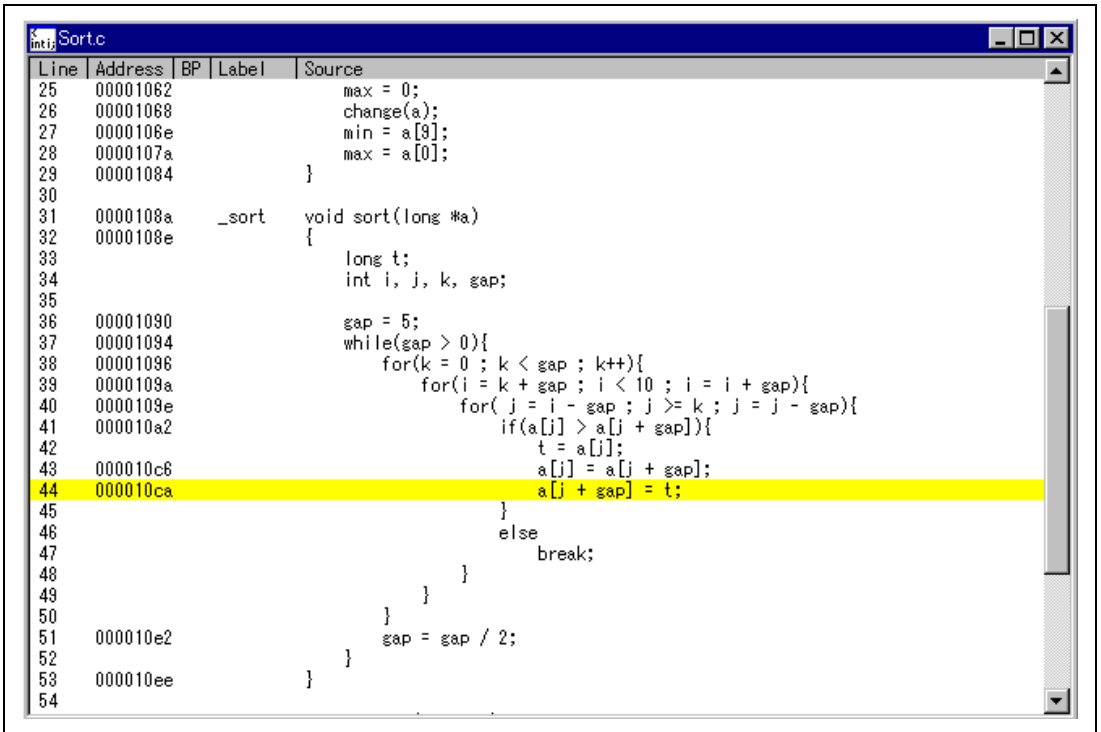


Execution will stop at the PC breakpoint set at address H' 1040.

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



Execution will stop at address H'10ca.



Line	Address	BP	Label	Source
25	00001062			max = 0;
26	00001068			change(a);
27	0000106e			min = a[9];
28	0000107a			max = a[0];
29	00001084			}
30				
31	0000108a		_sort	void sort(long *a)
32	0000108e			{
33				long t;
34				int i, j, k, gap;
35				
36	00001090			gap = 5;
37	00001094			while(gap > 0){
38	00001096			for(k = 0 ; k < gap ; k++){
39	0000109a			for(i = k + gap ; i < 10 ; i = i + gap){
40	0000109e			for(j = i - gap ; j >= k ; j = j - gap){
41	000010a2			if(a[j] > a[j + gap]){
42				t = a[j];
43	000010c6			a[j] = a[j + gap];
44	000010ca			a[j + gap] = t;
45				}
46				else
47				break;
48				}
49				}
50				}
51	000010e2			gap = gap / 2;
52				}
53	000010ee			}
54				

Figure 4.34 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.

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 4.35.

Cycle	Address	Label	Code	Data	R/W	Area	Status	Clock	Probes	IRQ1	IRQ2	Source	
-00026	0010a6		SHLL.L	ER4	1034	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00025	0010a8		SHLL.L	ER4	1034	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00024	0010aa		MOV.L	ER2,ER0	0fa0	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00023	0010ac		ADD.L	ER0,ER4	0a84	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00022	0010ae		MOV.W	ER,ER5	0de6	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00021	0010b0		ADD.W	R6,R5	0965	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00020	0010b2		EXTS.L	ER5	17f5	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00019	0010b4		SHLL.L	ER5	1035	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00018	0010b6		SHLL.L	ER5	1035	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00017	0010b8		ADD.L	ER0,ER5	0a85	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00016	0010ba		MOV.L	0ER4,ER0	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00015	0010bc				6940	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00014	0010be		MOV.L	0ER5,ER1	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00013	ffcfc				0000	RD	RAM	DATA	2	1111	HIGH	HIGH	
-00012	ffcfc				41c8	RD	RAM	DATA	2	1111	HIGH	HIGH	
-00011	0010c0				6951	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00010	0010c2		CMP.L	ER1,ER0	1f90	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00009	ffcfe0				0000	RD	RAM	DATA	2	1111	HIGH	HIGH	
-00008	ffcfe2				15fb	RD	RAM	DATA	2	1111	HIGH	HIGH	
-00007	0010c4		BLE	0H'10D4:8	4f0e	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00006	0010c6		MOV.L	ER1,0ER4	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00005	0010d4				096b	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00004	0010c8				69c1	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00003	0010ca				0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00002	ffcfc				0000	WR	RAM	DATA	2	1111	HIGH	HIGH	
-00001	ffcfc				15fb	WR	RAM	DATA	2	1111	HIGH	HIGH	
+00000	0010cc				69d0	RD	ROM	PROG	2	1111	HIGH	HIGH	

Figure 4.35 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 -00003, you can see that address H'011ca has been accessed.

4.9.2 Setting a Trace Filter

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

- First click in the **Trace** window with the right mouse button and choose **Clear** from the popup menu to clear the existing trace buffer.
- Then choose **Filter** to display the **Trace Filter** dialog box.

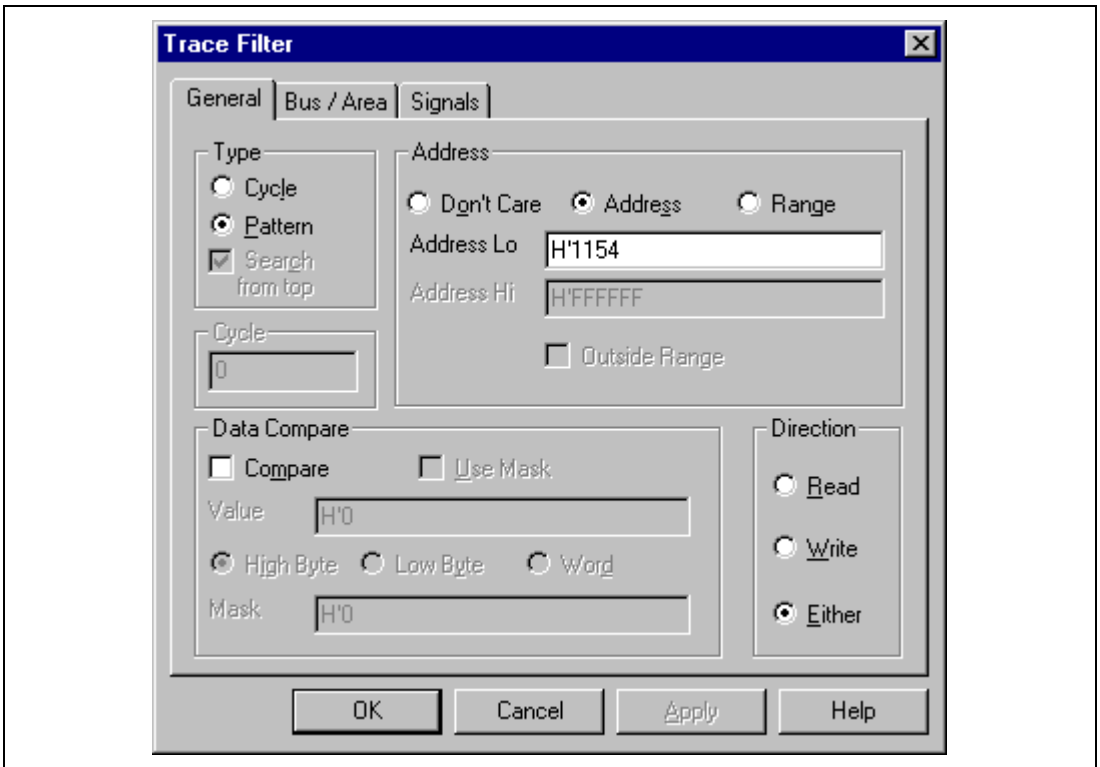


Figure 4.36 General Panel in Trace Filter Dialog Box

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' 10ca in the **Address Lo** field.
- Click **Bus / Area** to display the **Bus / Area** panel.
- Set **Bus State** to CPU Prefetch.

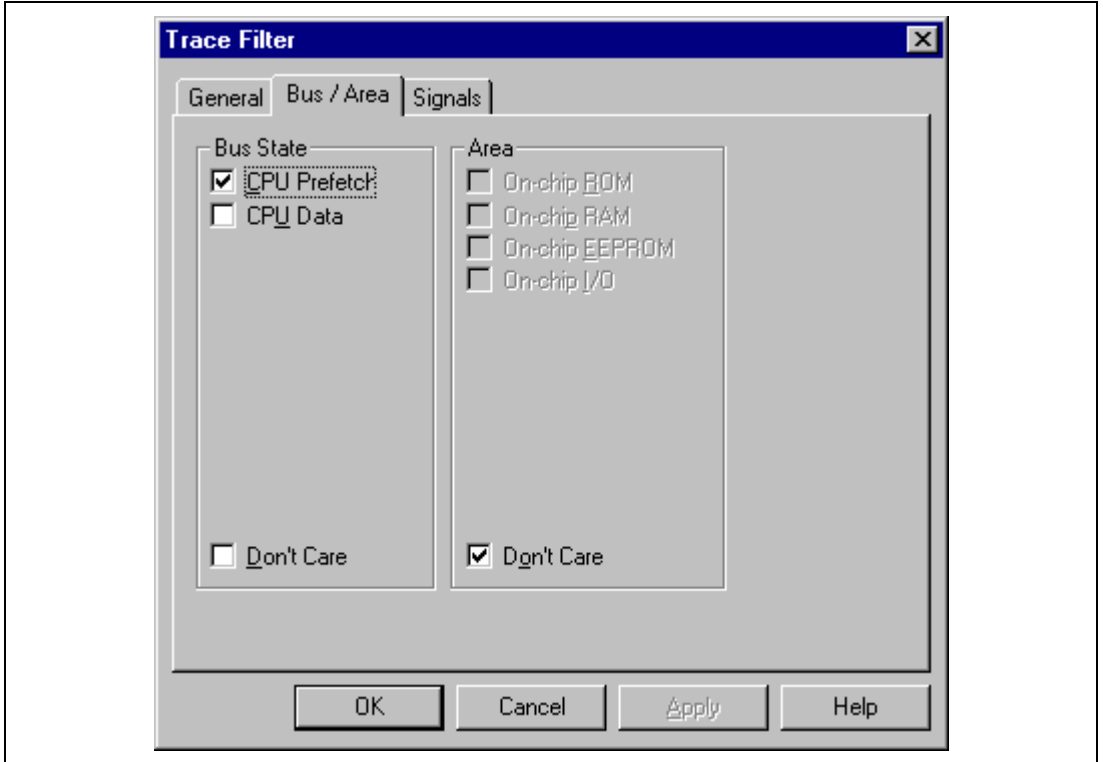


Figure 4.37 Bus / Area Panel in Trace Filter Dialog Box

- Click **OK** to save the trace filter.
- Choose **Breakpoints** from the **View** menu and open the **Breakpoints** window and breakpoints.
- Then choose **Reset Go** from the **Run** menu to execute the program.
- Then choose **Halt** from the **Run** menu to halt execution to see the trace buffer.

In the **Trace** window, only the cycles in which the MCU accessed address H' 10ca are displayed.

Cycle	Address	Label	Code	Data	R/W	Area	Status	Clock	Probes	IRQ1	IRQ2	Source	
-01705	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-01599	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-01540	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-01481	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-01374	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-01339	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-01257	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-01105	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-01070	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00905	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00800	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	
-00660	0010ca		MOV.L	ER0,0ER5	0100	RD	ROM	PROG	2	1111	HIGH	HIGH	

Figure 4.38 Trace Window (When Trace Filter is Specified)

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.

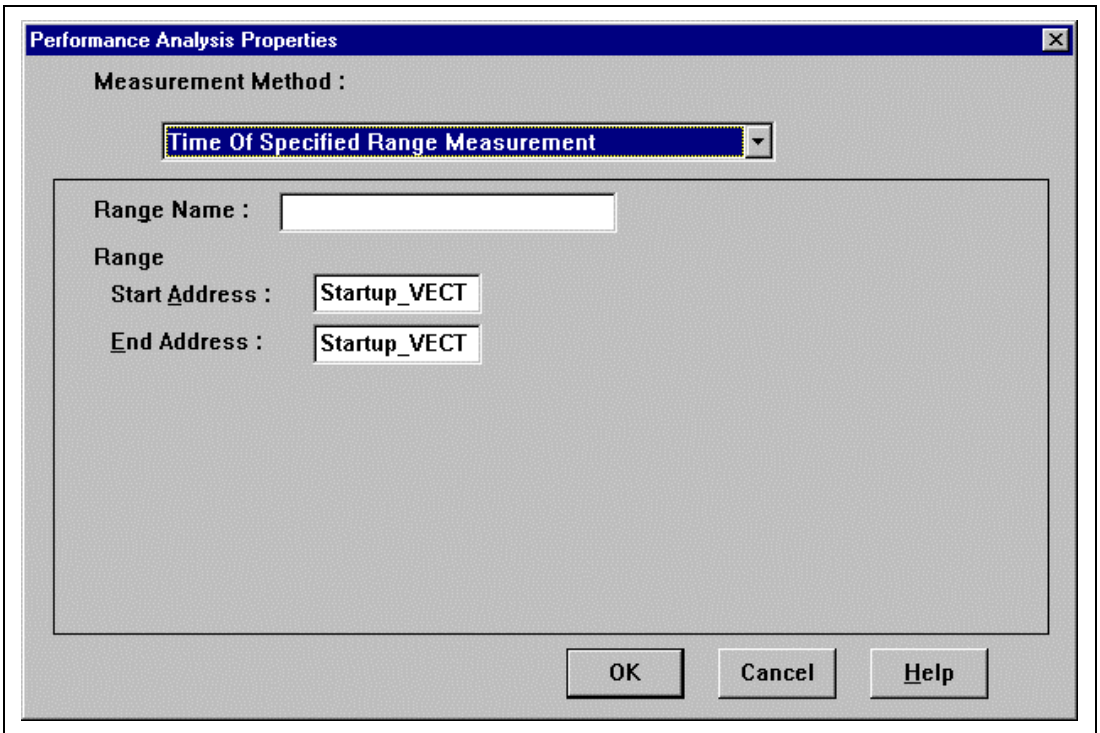


Figure 4.39 Selecting the Conditions for Measurement

- Select **Time Of Specified Range Measurement** from **Measurement Method** to measure the performance over the specified range.
- Input Range1 as the **Range Name**.
- Input address H' 10c8 as the **Start Address** and address H' 120c 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.

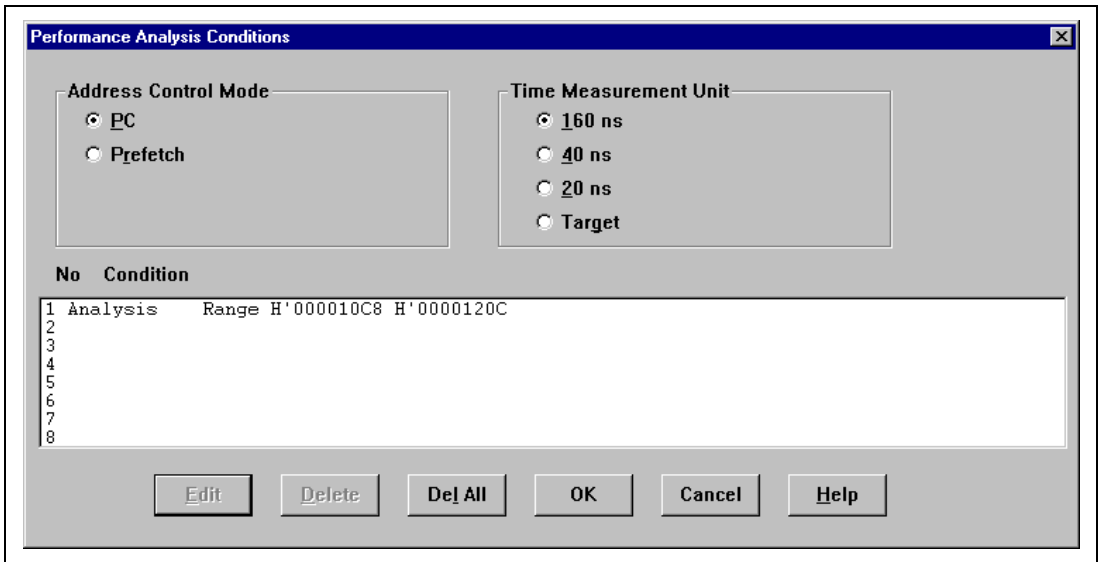


Figure 4.40 Displaying the Measurement Conditions

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

Now, the performance of the execution in the address range H'10c8 to H'120c can be measured.

- Click **Close** and close the **Performance Analysis** dialog box.
- 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'104e.

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.

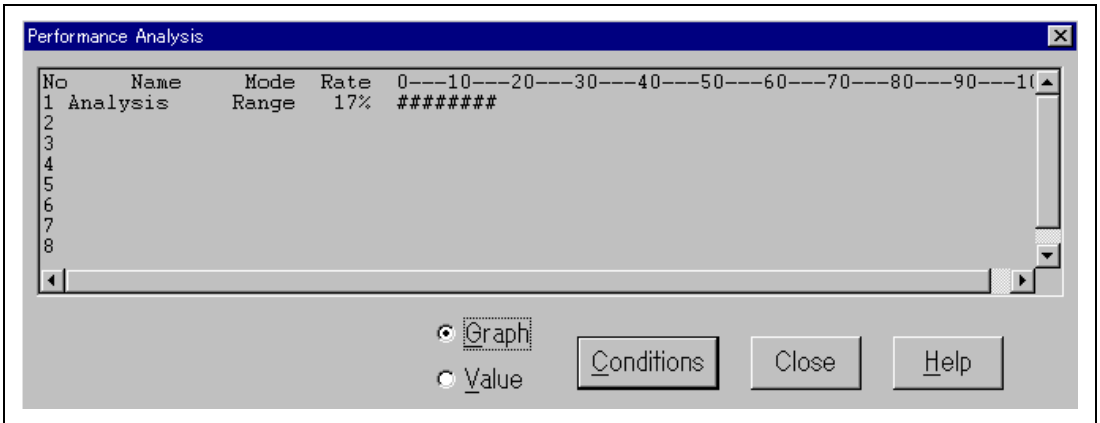


Figure 4.41 Displaying the Analysis Results (1)

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

- Click **Value**.

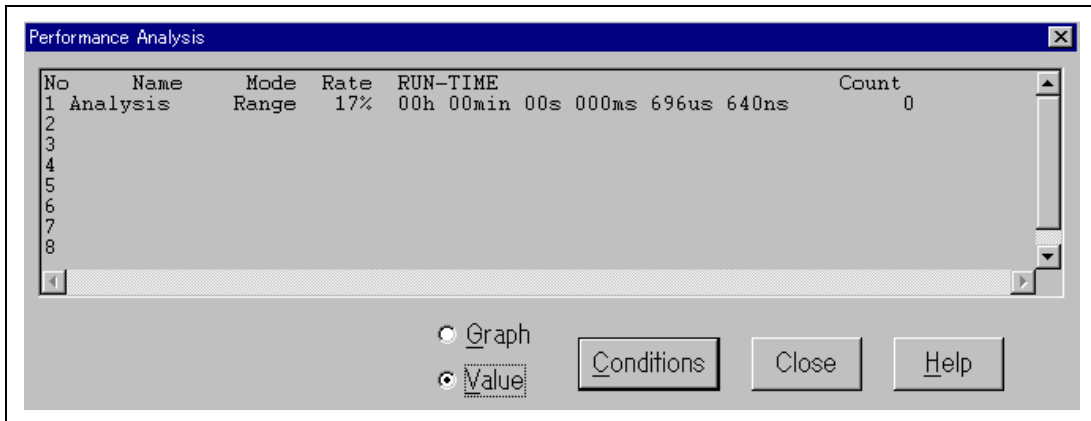


Figure 4.42 Displaying the Analysis Results (2)

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

4.11 Bus Monitor

You will be able to refer to the contents of memory in realtime (minimum time to update the window is 1s) as shown in the example window below when you execute the user program.

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

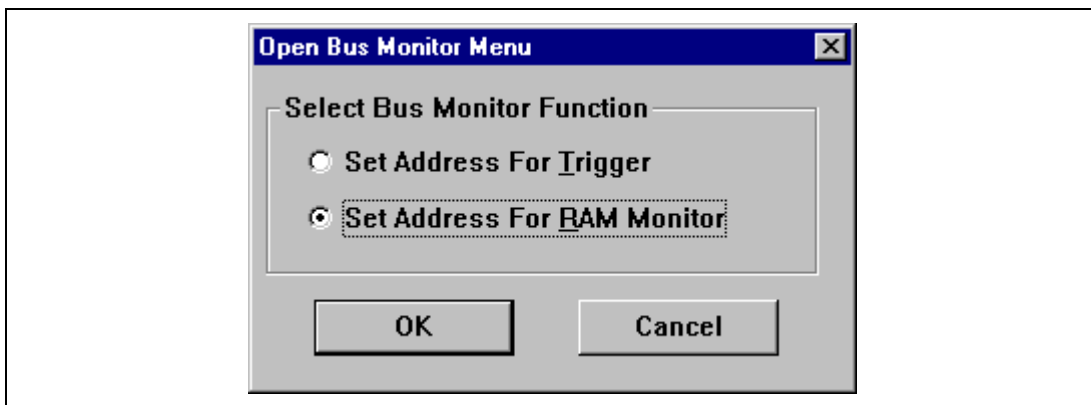


Figure 4.43 Open Bus Monitor Menu Dialog Box

- Choose **Set Address For RAM Monitor** from the **Select Bus Monitor Function**, and click **OK**.

- Select the check box left to **Monitor 1** and input H'00ffc00, choose Access, then click the **OK** button.

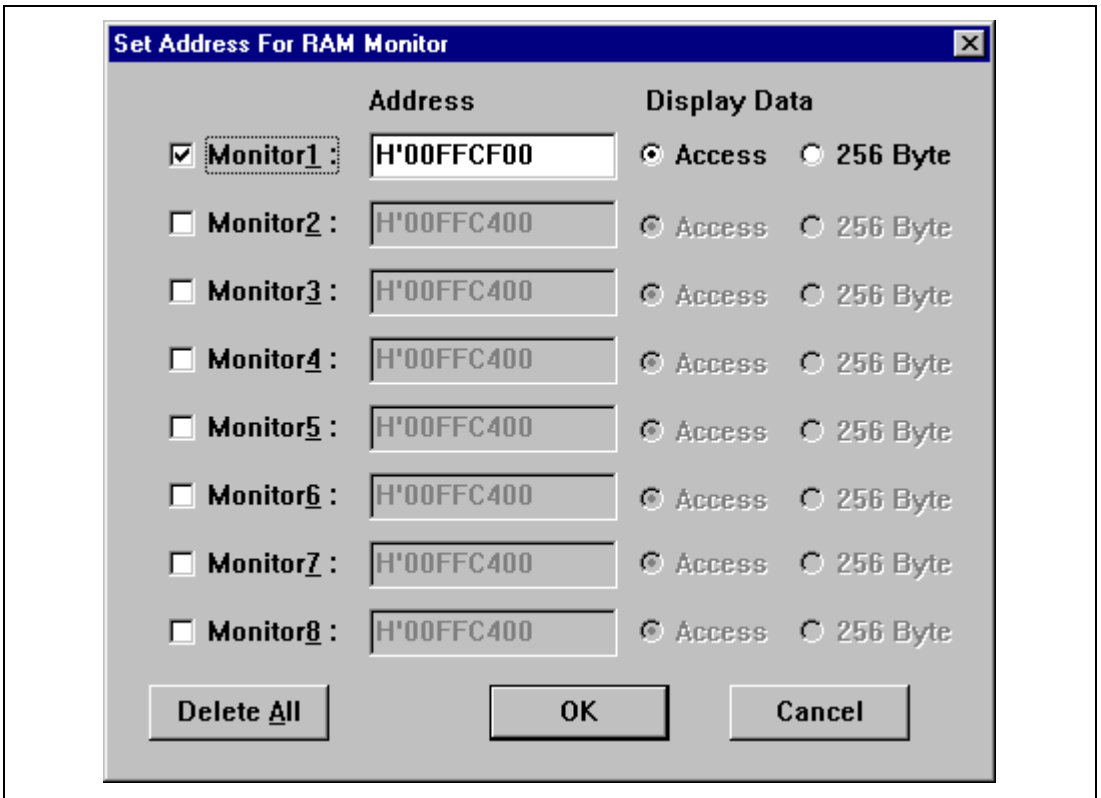


Figure 4.44 Set Address For RAM Monitor Dialog Box

- Choose **Reset Go** from the **Run** menu or click the **Reset Go** button on the tool bar.

You can now watch the memory contents being updated in realtime (minimum time for updating the window is 1s) as shown in the example window below.

In this tutorial, since the user program stops at PC breaks, only accessed addresses can be confirmed.

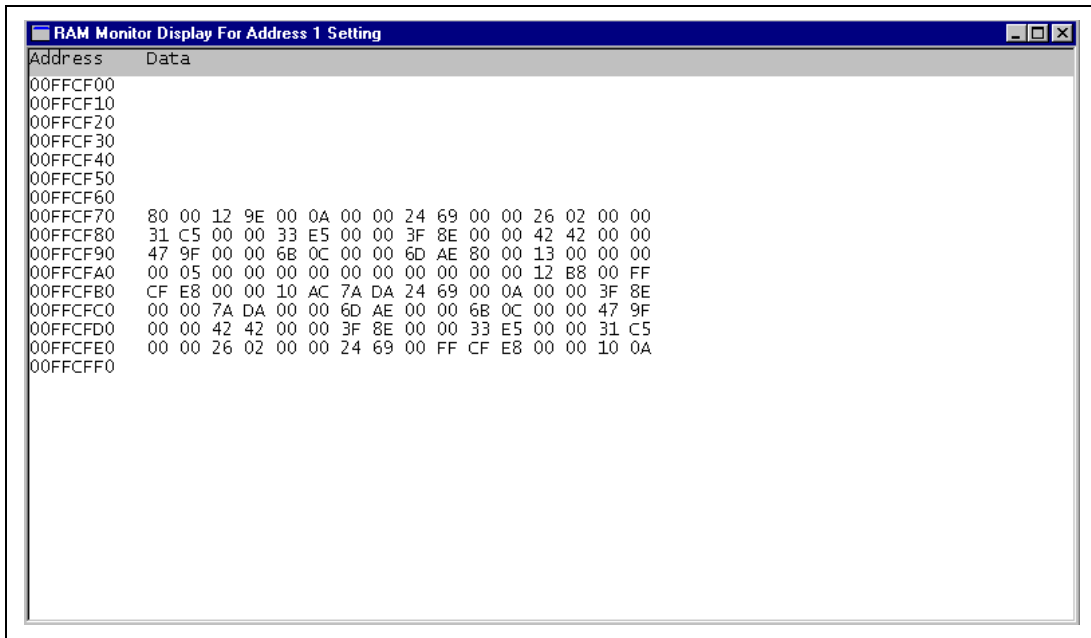


Figure 4.45 RAM Monitor Display

4.12 Stack Trace Function

By using the stack trace function, you can check the history of functions called when user programs stopped and the results of having allocated automatic variables.

- Double click the BP column, including address H'1090, in the **Program** window and set PC Break.
- Select **Reset Go** from the Run menu or click the **Reset Go** button in the toolbar and execute the program from the initial position. The program execution stops at address H'1090 by PC Break that has been set.
- Select **Stack Trace** from the **View** menu and open the **Stack Trace** window.
- Click in the **Stack Trace** window with the right mouse button, select **View Setting** from the pop-up menu, and open the **Stack Trace Setting** window.
- Check **Parameter**, **Local Variable**, and **Hexadecimal** in the **Stack Trace Setting** window and click **OK**.

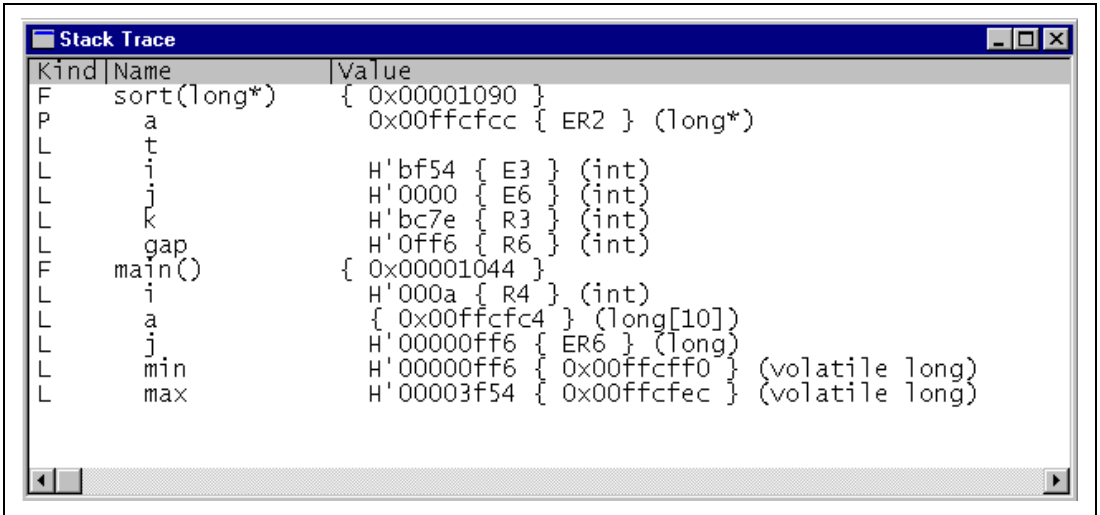


Figure 4.46 Stack Trace Window

You can see that the PC is currently in the sort function and is called from the main function.

Note: This function can be used only when a load module with debugging information in Dwarf2 format has been loaded.

4.13 Coverage Function

When the coverage function is used, you can check what instruction has been executed and what memory has been accessed when the user program stopped.

- Remove all break points by selecting **Delete All** from the pop up menu appeared by clicking the right button of mouse on the **Breakpoints** window.
- Select **Reset Go** from the **Run** menu or click the **Reset Go** button in the toolbar and execute the program from the initial position. When the state of the status bar becomes **Status = SLEEP1**, click the Stop button on the toolbar and stop the program execution.
- Select **Coverage** from the **View** menu and open the **Coverage Main** window.
- Click **Display Coverage Update** button in the **Coverage Main** window, reflect the results of execution, and select **512 bytes** from **Display Scale** drop down list box.
- Input address H'1000 for the display start address on the **Jump to address** dialog box appeared by double-clicking the graph area.

The program execution history is displayed with **Display Key** colors in the graph area.

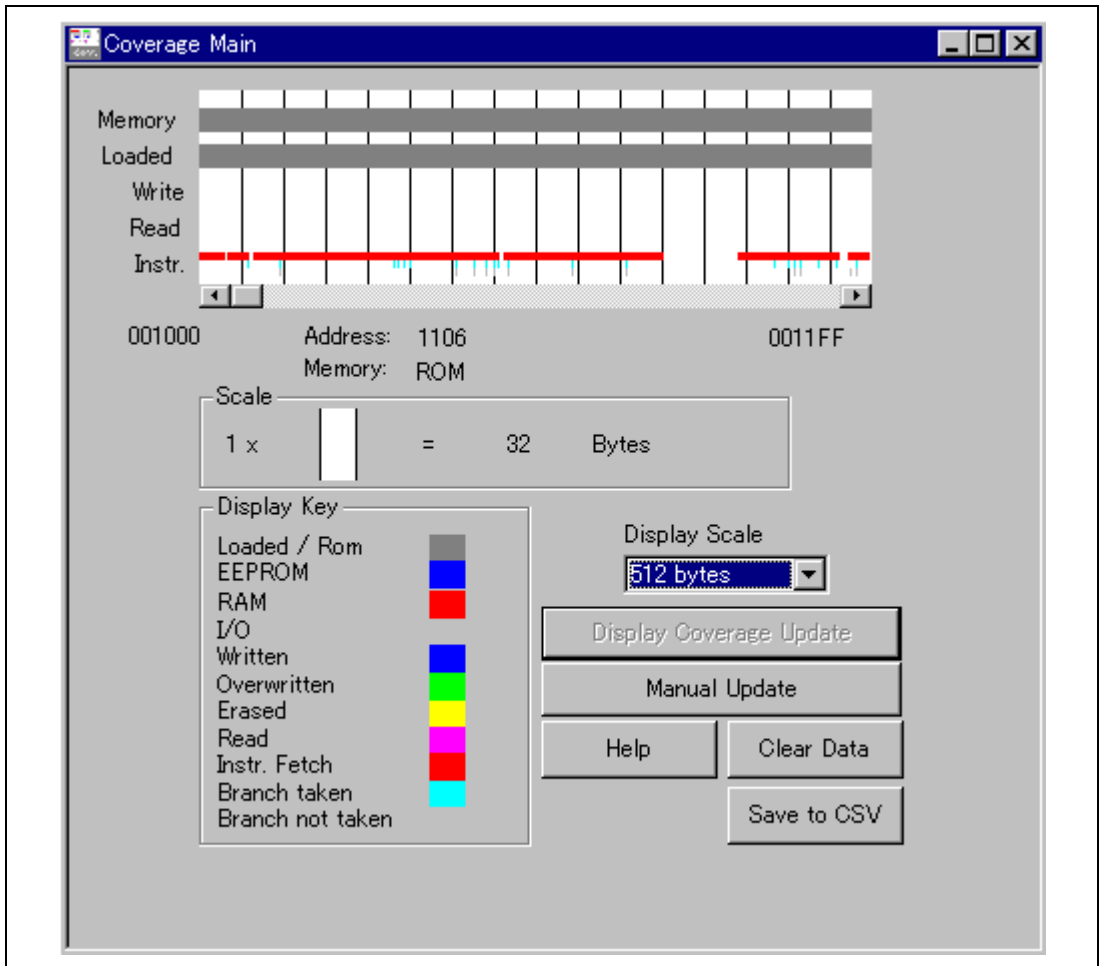


Figure 4.47 Coverage Main Window

Clicking the **Save to CSV** button in the **Coverage Main** window saves this coverage result with the CSV-file format.

For details on the above functions, refer to the online help. Online help can be displayed by clicking the **Help** button in any window or pressing the **F1** key.

4.14 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.

4.15 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.

Section 5 Diagnostic Test Procedure

This section describes the diagnostic test procedure using the E6000 emulator test program.

5.1 System Set-Up for Test Program Execution

To execute the test program, use the following hardware; do not connect the user system interface cable and user system.

- AE-4 E6000 emulator (HS0AE4XEPI61H)
 - Host computer (MS-DOS)
 - E6000 PC interface board (One of the following interface boards is shown in this manual. Prepare a suitable one according to your PC interface specifications.)
 - ISA bus interface board (HS6000EII01H)
 - PCI bus interface board (HS6000EIC01H, HS6000EIC02H)
 - PCMCIA interface card (HS6000EIP01H)
 - LAN interface adapter (HS6000ELN01H)
1. Install the E6000 PC interface board in the host computer and connect the supplied PC interface cable to the board.
 2. Connect the PC interface cable to the AE-4X E6000.
 3. Connect the supplied AC adapter to the AE-4X E6000.
 4. Initiate the host computer to make it enter DOS prompt command input wait state.
 5. Turn on the E6000 switch.

5.2 Diagnostic Test Procedure Using Test Program

Insert the CD-R (HS0AE4EPI61SR supplied with the E6000) into the CD-ROM drive of the host computer by pressing the Shift key, move the current directory to <Drive>:\Diag with a command prompt, and enter one of the following commands according to the PC interface board used to initiate the test program:

1. ISA bus interface board (HS6000EII01H)
>TM0AE4X -ISA (RET)
2. PCI bus interface board (HS6000EIC01H, HS6000EIC02H)
>TM0AE4X -PCI (RET)
3. PCMCIA interface card (HS6000EIP01H)
>TM0AE4X -PCCD (RET)
4. LAN interface adapter (HS6000ELN01H)
>TM0AE4X -ELN (RET)

The HDI must be installed before the test program is executed.

Be sure to initiate the test program from <Drive>:\Diag. Do not initiate it from a directory other than <Drive>:\Diag, such as <Drive>:\Diag\TM0AE4X (RET). If the test program is initiated when the current directory is not <Drive>:\Diag, the test program will not operate correctly.

When -S is added to the command line such as >TM0AE4X -ISA -S (RET), steps 1 to 24 will be repeatedly executed. To stop the execution, enter Q.

- Notes:
1. When the CD-R is inserted into the CD-ROM drive without pressing the Shift key, the HDI installation wizard is automatically started.
In such a case, exit the HDI installation wizard.
 2. <Drive> is a drive name for the CD-ROM drive.
 3. Do not remove the CD-R from the CD-ROM drive during test program execution.

Messages and test contents displayed during the tests are as follows. You can perform tests No. 1 through No. 24. (The testing time is about 20 minutes when you use a PCI interface card on Windows® 98 at 333 MHz with 64 MB of main memory.)

Message	Description
E6000 AE-4X Emulator Tests Vn.m Copyright (c) 2001 Hitachi Ltd.	Test program start message. Vn.m shows the version number.
Loading driver.....OK (Use PCI)	Shows that the driver software for the PC interface board has been normally loaded.
Initializing driver.....OK	Shows that the driver software for the PC interface board has been normally initialized.
Searching for interface cardOK	Shows that the PC interface board is correctly installed in the host computer.
Checking emulator is connectedOK	Shows that the E6000 emulator is correctly connected to the host computer.
Emulator board information: Main board ID: H' 5 Emulation board ID: H' 1c	Shows the ID number of the lower board of the E6000 emulator (always 5). Shows the ID number of the upper board of the E6000 emulator (always 1c).

- 1) Test Register
 - A) IDR0 RegisterOK
 - B) PAGE RegisterOK
 - C) TRACE G/A RegisterOK
 - D) PERFM G/A RegisterOK
 - E) CES G/A RegisterOK
 - F) IDR1 RegisterOK
 - G) IDR2 RegisterOK
 - H) EEMAX RegisterOK
 - I) CHIPTYPE RegisterOK
- 2) Test DPRAM
 - A) Decode TestOK
 - B) Marching TestOK
- 3) Test Firmware RAM
 - A) Decode Test page [H'700 - H'71f]
.....OK
 - B) Marching test page [H'700 - H'71f]
.....OK
- 4) Test Trace memory
 - A) Decode Test page [H'000 - H'04f]
(Lower 32K)OK
 - B) Marching test page [H'000 - H'04f] (Lower 32K)
..... OK
 - C) Decode Test page [H'000 - H'04f]
(Upper 32K).....OK
 - D) Marching test page [H'000 - H'04f]
..... (Upper 32K).....OK
- 5) Test Map control memory
 - A) Decode Test page [H'200 - H'27f]
.....OK
 - B) Marching test page [H'200 - H'27f]
.....OK
 - C) Coverage RAM TestOK
 - D) Coverage RAM Test2 page [H'900 - H'9ff] H'a00

Shows the check results for the registers in the E6000 emulator.

Shows the results of decoding test and marching test for the dual-port RAM in the E6000 emulator (normal completion).

Shows the results of decoding test for the firmware RAM in the E6000 emulator (normal completion).

Shows the results of marching test in the E6000 emulator (normal completion).

Shows the results of decoding test for the trace RAM in the E6000 emulator (normal completion).

Shows the results of marching test in the E6000 emulator (normal completion).

Shows the results of decoding test for the mapping RAM in the E6000 emulator (normal completion).

Shows the results of marching test in the E6000 emulator (normal completion).

Shows the results of decoding test for the Coverage RAM in the E6000

.....	OK
6) Test Internal ROM and RAM	
A) Decode Test (Internal ROM)	OK
B) Marching test (Internal ROM)	OK
C) Decode Test (Internal RAM)	OK
D) Marching Test (Internal RAM)	OK
7) RESERVED	
8) Test Emulation RAM STEP Operation	
A) Step Operation	OK
9) Test Keybreak	
Key Break	OK
10) Test Emulation RAM Hardware Break	
A) GRD Break	OK
B) WPT Break	OK
11) Test Internal ROM Write-Protect	
A) ROMWR Test	OK
12) Test Hardware Break	
A) Break Point Initialized.....	OK
B) Event Detectors CES channel 1-12	OK
C) Test Sequencing 1	OK
D) Check Range Break	OK
E) Range Break Test for Data	OK
F) Check Compare Either	OK

emulator (normal completion).

Shows the results of decoding and marching test for the emulation RAM in the E6000 emulator (normal completion).

Shows the check results for the step execution controlling circuits in the E6000 emulator (normal completion).

Shows the check results for the forced break controlling circuits in the E6000 emulator (normal completion).

Shows the check results for the illegal access break controlling circuits in the E6000 emulator (normal completion).

Shows the check results for internal ROM controlling circuits in the E6000 emulator (normal completion).

Shows the check results for hardware break controlling circuits in the E6000 emulator (normal completion).

13) Test Emulation RAM Trace

- A) Free Trace.....OK
- B) Range Trace.....OK
- C) Point to Point TraceOK
- D) Start and Stop Event TraceOK
- E) Trace memory OverflowOK
- F) Time STAMP Trace.....OK
- G) Start and Stop Event Trace Test2OK
- H) ASEJTC bit Trace Test.....OK

Shows the check results for the trace controlling circuits in the E6000 emulator (normal completion).

14) Test Runtime Counter

- A) Runtime Counter (9.8304MHz) OK
- B) Runtime Counter (7.1424 MHz) OK
- C) Runtime Counter (4.9152 MHz) OK
- D) Runtime Counter (3.5712 MHz) OK
- E) Runtime Counter (2.4576 MHz) OK
- F) Runtime Counter (1.7856 MHz) OK

Shows the check results for execution time measurement circuits in the E6000 emulator (normal completion).

15) Test Emulation Monitor

- A) EMA23 – EMA0 OK
- B) ACST2 – ACST0 OK
- C) ST3 – ST0 OK
- D) ASE BRKACK OK
- E) CNN OK
- F) RESERVED
- G) INST OK
- H) HLT BIT (MONIT2O: D6) OK
- I) EVAGDD BIT (MONIT2O: D5) OK
- J) RF_VCC BIT (MONIT2E: D3) OK

Shows the check results for monitor controlling circuits in the E6000 emulator (normal completion).

16) Testing PERFM G/A

- A) Time Measurement.....OK
- B) RESERVED
- C) Subroutine Count MeasurementOK
- D) Timeout Function (TIMOT Bit.....OK
- E) Timeout Function (TIMOP BitOK

Shows the check results for execution time measurement controlling circuits in the E6000 emulator (normal completion).

17) Test Bus Monitor

- A) Register testOK
- B) Parallel RAM testOK
- C) SPRSEL2 testOK
- D) RAM monitor testOK

Shows the check results for bus monitor circuits in the E6000 emulator (normal termination).

18) RESERVED

19) Test Go Reset

- A) GoReset.....OK
- B) BUS 00>FF Test.....OK

Shows the check results for GoReset controlling circuits in the E6000 emulator (normal termination).

20) Test PC_CVR

- A) RESERVED
- B) ALL Coverage Test 2OK
- C) Area Coverage Test 1OK
- D) Area Coverage Test 2OK
- E) Area Coverage Test 3OK
- F) C1 Coverage Test.....OK
- G) C1 Coverage Test 2OK
- H) LID2BCC Coverage.....OK

Shows the check results for coverage controlling circuits in the E6000 emulator (normal completion).

21) Test EEPROM

- EEPROM TEST (ECR Reg OC1, 0=0, 0) No.1
- A-1) Decode Test (PBMSSEL bit : 0OK
- A-2) Decode Test (PBMSSEL bit : 1OK
- A-3) EEPMOV By UserRunOK
- A-4) Shift Address Test.....OK
- B) EEPROM TEST (ECR Reg OC1, 0=0, 0) No.2OK
- C) EEPROM TEST (ECR Reg OC1, 0=0, 1)OK
- D) EEPROM DELETE TEST (ECR Reg OC1, 0=1, 0) ..OK
- E) EEPROM PROTECT TESTOK
- F) EEMAX TEST 1OK
- G) EEMAX TEST 2OK
- H) EEMOV ON EEPROM TESTOK
- I) EEPMOV TIME MEASUREMENT TEST.....OK
- J) EEPROM WR TESTOK
- K) EMR TESTOK

Shows the check results for writing to EEPROM/rewriting to EEPROM/erasing and its erasing time, EEMAX Break write-protect controlling circuits in the E6000 emulator (normal completion).

22) Test Double Stack, I/O Stack

- A) Double Stack Test OK
- B) I/O Stack Test 1 OK
- C) I/O Stack Test 2 OK
- D) RAM Stack Test 3 OK

Shows the check results for Double Stack, I/O Stacks and RAM Stack controlling circuits in the E6000 emulator (normal completion).

23) Test SUB_CHIP

- A) RAM Test OK
- B) MLTIRQ Line Test OK
- C) MLTMON Line Test OK
- D) MLTCONT Line Test..... OK
- E) MLTRES Line Test..... OK
- F) MLTSLEEP Line Test..... OK
- G) MLTIO Line Test OK
- H) UART,TIMER Register Test OK
- I) RF Register Test..... OK

Shows the check results for sub-chip bus circuit in the E6000 emulator (normal termination).

24) Test DATA_CVR

- A) ALL Coverage Test 1OK
- B) Area Coverage Test 1OK
- D) Area Coverage Test 2OK
- F) RDWR Coverage Test.....OK
- G) EEPWR Coverage TestOK
- H) IO Coverage Test.....OK

Tests run for 0H: 20M: 46S

Tests passed, emulator functioning correctly

Shows the check results for coverage controlling circuits in the E6000 emulator (normal completion).

When -S is added to the command line, steps 1 to 24 will be repeatedly executed.

When detecting an error, the test program displays ERROR and stops execution. In this case, the emulator hardware may be malfunctioning. Inform a Hitachi sales agency of the test results in detail.

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
COVERAGE	CV	Specific	Displays coverage information
COVERAGE_CLEAR	CC	Specific	Clears coverage information
DEVICE_TYPE	DE	Specific	Selects the target device in the E6000 emulator
DISASSEMBLE	DA	General	Disassembles and displays a program
EEPROM_DISPLAY	EP	Specific	Displays protected areas of EEPROM
EEPROM_MAP	EM	Specific	Sets or cancels protected areas of EEPROM
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 platforms
LOG	LO	General	Operation of logging files

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
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_ALL	TL	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.