User's Manual

SuperH™ Family E10A-USB Emulator for Multi-core Microcomputers

User's Manual

SuperH™ Family E10A-USB

HS0005KCU04HE

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corporation without notice. Please review the latest information published by Renesas Electronics Corporation through various means, including the Renesas Electronics Corporation website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev. 5.00 Aug. 2012

Regulatory Compliance Notices

● European Union regulatory notices

This product complies with the following EU Directives. (These directives are only valid in the European Union.)

CE Certifications:

• Electromagnetic Compatibility (EMC) Directive 2004/108/EC

EN 55022 Class A

WARNING: This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

EN 55024

• Information for traceability

• Authorised representative

Name:	Renesas Electronics Corporation
Address:	1753, Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, 211-8668, Japan

• Manufacturer

Name:	Renesas Solutions Corp.
Address:	Nippon Bldg., 2-6-2, Ote-machi, Chiyoda-ku, Tokyo 100-0004, Japan

• Person responsible for placing on the market

Name:	Renesas Electronics Europe GmbH
Address:	Arcadiastrasse 10, 40472 Dusseldorf, Germany

• Trademark and Type name

Trademark:	Renesas
Product name:	E10A-USB Emulator
Type name:	HS0005KCU04H / HS0005KCU14H

Environmental Compliance and Certifications:

• Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment (RoHS) Directive 2002/95/EC

• Waste Electrical and Electronic Equipment (WEEE) Directive 2002/96/EC

● United States Regulatory notices

This product complies with the following EMC regulation. (This is only valid in the United States.)

FCC Certifications:

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

CAUTION: Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

IMPORTANT INFORMATION

READ FIRST

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

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

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

Emulator Product:

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

- Emulator
- User system interface cable

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

Purpose of the Emulator Product:

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

Limited Applications:

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

Improvement Policy:

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

Target User of the Emulator Product:

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

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

LIMITED WARRANTY

Renesas warrants its emulator products to be manufactured in accordance with published specifications and free from defects in material and/or workmanship. Renesas, at its option, will replace any emulator products returned intact to the factory, transportation charges prepaid, which Renesas, upon inspection, shall determine to be defective in material and/or workmanship. The foregoing shall constitute the sole remedy for any breach of Renesas' warranty. See the Renesas 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. Renesas is not liable for any claim made by a third party or made by you for a third party.

DISCLAIMER

RENESAS 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 RENESAS BE LIABLE FOR ANY DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE, OR LOSSES OR EXPENSES RESULTING FROM ANY DEFECTIVE EMULATOR PRODUCT, THE USE OF ANY EMULATOR PRODUCT, OR ITS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCEPT AS EXPRESSLY STATED OTHERWISE IN THIS WARRANTY, THIS EMULATOR PRODUCT IS SOLD "AS IS ", AND YOU MUST ASSUME ALL RISK FOR THE USE AND RESULTS OBTAINED FROM THE EMULATOR PRODUCT.

State Law:

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

The Warranty is Void in the Following Cases:

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

All Rights Reserved:

Other Important Things to Keep in Mind:

1. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Renesas' semiconductor products. Renesas 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 Renesas.

Figures:

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

Device names:

This user's manual uses SHxxxx as an example of the device names.

Limited Anticipation of Danger:

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

SAFETY PAGE

READ FIRST

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

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

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

DEFINITION OF SIGNAL WORDS

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

DANGER

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

WARNING

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

CAUTION

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

CAUTION

CAUTION used without the safety alert symbol indicates a potentially hazardous situation which, if not avoided, may result in property damage.

NOTE emphasizes essential information.

⚠ WARNING

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

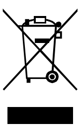
1. **Do not repair or remodel the emulator product by yourself for electric shock prevention and quality assurance.**
2. **Always switch OFF the host computer and user system before connecting or disconnecting any CABLES or PARTS.**
3. **Connect the connectors in the user system and in the user interface cable by confirming the correct direction.**

CAUTION

Caution to Be Taken for Disposal:

Penalties may be applicable for incorrect disposal of this waste, in accordance with your national legislation.

European Union regulatory notices:



The WEEE (Waste Electrical and Electronic Equipment) regulations put responsibilities on producers for the collection and recycling or disposal of electrical and electronic waste. Return of WEEE under these regulations is applicable in the European Union only. This equipment (including all accessories) is not intended for household use. After use the equipment cannot be disposed of as household waste, and the WEEE must be treated, recycled and disposed of in an environmentally sound manner.

Renesas Electronics Europe GmbH can take back end of life equipment, register for this service at “http://www.renesas.eu/weee”.

Warnings on Emulator Usage

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

WARNING

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

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

CAUTION

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

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

About This Manual

This manual describes preparation before using the emulator, emulator functions, debugging functions specific to the emulator, tutorial, and emulator's hardware and software specifications.

Refer to the High-performance Embedded Workshop User's Manual for details on the information on the basic usage of the High-performance Embedded Workshop, customization of the environment, build functions, and debugging functions common to each High-performance Embedded Workshop product.

This manual does not intend to explain how to write C/C++ or assembly language programs, how to use any particular operating system or how best to tailor code for the individual devices. These issues are left to the respective manuals.

Document Conventions

This manual uses the following typographic conventions:

Table 1 Typographic Conventions

Convention	Meaning
[Menu->Menu Option]	Bold text with '->' is used to indicate menu options (for example, **[File->Save As...]**).
FILENAME.C	Uppercase names are used to indicate filenames.
"<u>enter this string</u>"	Used to indicate text that must be entered (excluding the "" quotes).
Key + Key	Used to indicate required key presses. For example, **CTRL+N** means press the **CTRL** key and then, whilst holding the **CTRL** key down, press the **N** key.
➲ (The "how to" symbol)	When this symbol is used, it is always located in the left hand margin. It indicates that the text to its immediate right is describing "how to" do something.

User Registration

When you install debugger software, a text file for user registration is created on your PC. Fill it in and email it to your local distributor. If you have replaced an emulator main unit or emulation probe, rewrite an emulator name and serial number in the text file you filled in earlier to register your new hardware products.

Your registered information is used for only after-sale services, and not for any other purposes. Without user registration, you will not be able to receive maintenance services such as a notification of field changes or trouble information. So be sure to carry out the user registration.

For more information about user registration, please contact your local distributor.

Table of Contents

Section 1 Overview

The E10A-USB emulator (hereafter referred to as the emulator) is a support tool for developing the hardware and software of application systems to run on Renesas original microcomputers.

The main unit of the emulator is connected, through the dedicated debugging interface, to the user system. The user system can be debugged under the conditions similar to the actual application conditions. The emulator enables debugging anywhere indoors or out. The host computer for controlling the emulator must be an IBM PC compatible machine with USB 1.1/2.0 (Full-Speed).

The High-performance Embedded Workshop provides a graphical user interface that eases the development and debugging of applications written in the C/C++ programming languages or assembly language for Renesas microcomputers. Its aim is to provide a powerful yet intuitive way of accessing, observing and modifying the debugging platform on which the application is running.

Figure 1.1 shows the system configuration using the emulator.

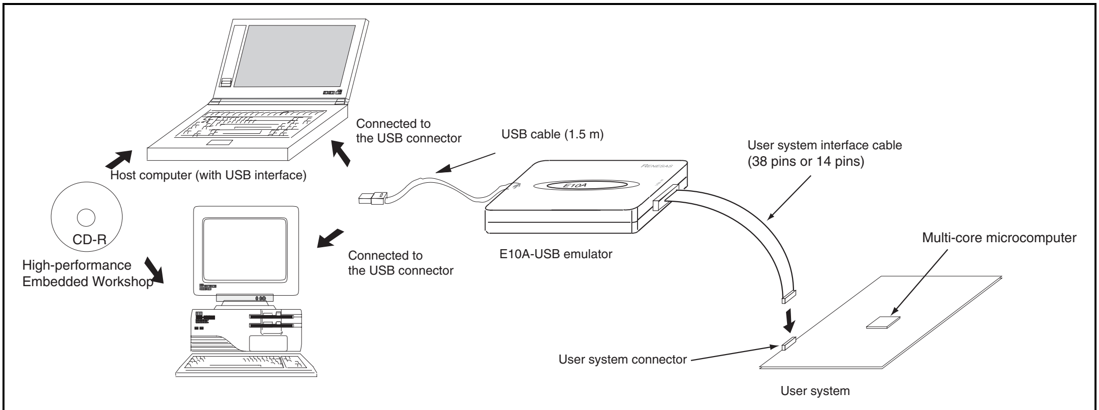


Figure 1.1 System Configuration with the Emulator

The emulator provides the following features:

- Excellent cost-performance emulator

  Compactness and connection to the USB are implemented.
- Realtime emulation

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

  Using the High-performance Embedded enables user program debugging using a pointing device such as a mouse. The High-performance Embedded Workshop enables high-speed downloading of load module files.
- Various debugging functions

  Various break and trace functions enable efficient debugging. Breakpoints and break conditions can be set by the specific window, trace information can be displayed on a window, and command-line functions can be used.
- Debugging of the user system in the final development stage

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

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

  The AUD trace function enables realtime trace.

Note: The AUD is an abbreviation of the Advanced User Debugger. Support for the AUD varies with the product.

1.1 Warnings

CAUTION

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

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

1.2 Environmental Conditions

CAUTION

Observe the conditions listed in tables 1.1 and 1.2 when using the emulator. Failure to do so will cause illegal operation in the user system, the emulator product, and the user program.

Table 1.1 Environmental Conditions

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

Table 1.2 lists the acceptable operating environments.

Table 1.2 Operating Environments

Item	Description	
Operating system	Windows® XP (32-bit version)	Windows Vista® (32-bit version), Windows® 7 (32-bit/64-bit version)
Host computer	IBM PC or compatible machine with USB 1.1/2.0 (Full-Speed).	
CPU	Core™ 2 Duo (2 GHz), or higher recommended	Core™ 2 Duo (3.16 GHz), or higher recommended
Memory capacity	1 Gbytes or more (at least 10 times the file size of load modules is recommended)	2 Gbytes or more (at least 10 times the file size of load modules is recommended)
Hard-disk capacity	Installation disk capacity: 600 Mbytes or more. (Prepare an area at least double the memory capacity (four-times or more recommended) as the swap area.)	
Pointing device such as mouse	Connectable to the host computer; compatible with Windows® XP, Windows Vista®, or Windows® 7.	
Display	Monitor resolution: 1024 x 768 or higher	
Power voltage	5.0 ± 0.25 V (USB-bus power type)	
Current consumption	500 mA (max.)	
CD-ROM drive	Required to install the High-performance Embedded Workshop for the emulator or refer to the emulator user's manual.	

Note: Microsoft, Windows, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and or other countries. All other brand and product names are trademarks, registered trademarks or service marks of their respective holders.

1.3 Components

Check that all of the components are present when unpacking the product. For details on the emulator components, refer to section 1.1 in the additional document, ″Supplementary Information on Using the SHxxxx″. If all of the components are not present, contact your nearest Renesas sales office or contact center (csc@renesas.com)

Section 2 Emulator Functions

This section describes the emulator functions. They differ according to the device supported by the emulator. For the usage of each function, refer to section 6, Tutorial [SH-2A] or section 7, Tutorial [SH-4A].

2.1 Overview

Table 2.1 gives a functional overview of the emulator.

For details on the functions of each product, refer to the online help.

Table 2.1 Emulator Functions

No.	Item	Function
1	User program execution function	• Executes a program with the operating frequency within a range guaranteed by devices. • Reset emulation • Step functions: Single step (one step: one instruction) Source-level step (one step: one-line source) Step over (a break did not occur in a subroutine) Step out (when the PC points to a location within a subroutine, execution continues until it returns to the calling function) • Synchronized functions: Synchronized execution (execution by both of the CPUs proceeds at the same time and is synchronized with execution by one CPU) Synchronized step functions. (All of the CPUs execute with synchronizing the one of stepping for the CPU.) Synchronized break functions. (All of the CPUs break by synchronizing the one of a break for the CPU.)
2	Reset function	• Issues a power-on reset from the High-performance Embedded Workshop to the device during break.
3	Trace functions	• Trace function incorporated in the device • AUD trace: Branch trace or memory access trace • Memory output function of trace data
4	Break functions	• Hardware break condition (conditions and the number of conditions differ according to the device) • PC break condition (255 points) • Forced break function

Table 2.1 Emulator Functions (cont)

No.	Item	Function
5	Performance measurement function	• Uses a counter in the device to measure the number of cycles that passes during point-to-point execution.
6	Memory access functions	• Downloading to RAM • Downloading to flash memory • Single-line assembly • Reverse assembly (disassembly) • Reading of memory • Writing to memory • Automatic updating of a display of selected variables during user program execution • Fill • Search • Move • Copy • Monitor (physical address)
7	General/control register access function	• Reads or writes the general/control registers.
8	Internal I/O register access function	• Reads or writes the internal I/O registers.*
9	Source-level debugging function	• Various source-level debugging functions.
10	Command line function	• Supports command input. • Batch processing is enabled when a file is created by arranging commands in input order.
11	Help function	• Describes the usage of each function or command syntax input from the command line window.

Note: The [IO] window displays the contents defined in [SHxxxx.io]. Editing those contents adds or deletes the registers to be displayed. For the contents to be described as [SHxxxx.io], refer to reference 6, I/O File Format, in the High-performance Embedded Workshop V.4.09 User's Manual.
The following directory contains [SHxxxx.io] (xxxx means the name of emulator device group.):
<High-performance Embedded Workshop folder>:
\Tools\Renesas\DebugComp\Platform\E10A-USBM\xxxx\IOFiles

The specific functions of the emulator are described in the next section.

2.2 Trace Functions

The emulator has two trace functions.

2.2.1 Internal Trace Function

The branch source and branch destination addresses, mnemonics, operands, and source lines are displayed. This function uses the trace buffer built into the device.

Notes: 1. The number of branch instructions that can be acquired by a trace differs according to the product. For the number that can be specified for each product, refer to the online help.
2. The internal trace function is not supported for all products. For details on the specifications of each product, refer to the online help.
3. The internal trace function is extended for some products. For details on the specifications of each product, refer to the online help.

2.2.2 AUD Trace Function

This is the large-capacity trace function that is enabled when the AUD pins are connected to the emulator. If an event occurs to acquire a trace, trace information is output in realtime from the AUD pins.

When a set of the branch source and branch destination instructions is one branch, the maximum amount of information acquired by a trace is 32,767.

(1) Trace acquisition event

The following events can be acquired by the AUD trace function.

(a) Branch generation information

The branch source and branch destination addresses are acquired.

(b) Memory access information within the specified range

Memory access in the specified range can be acquired by trace.

Two memory ranges can be specified for channels A or B. The read, write, or read/write cycle can be selected as the bus cycle for trace acquisition.

This function is called the window trace function.

(c) Software trace

When a specific instruction is executed, the PC value at execution and the contents of one general register are acquired by trace. Describe the Trace(x) function (x is a variable name) to be compiled and linked beforehand. For details, refer to the SHC/C++ compiler manual.

When the load module is loaded on the emulator and a valid software trace function is executed, the PC value that has executed the Trace(x) function, the variable for x, and the source lines are displayed.

Note: The types of events acquired by a trace differ depending on the product. For details on the specifications of each product, refer to the online help.

(2) Trace acquisition mode

The AUD trace function has the following modes to acquire a trace.

Table 2.2 shows the AUD trace acquisition mode that can be set in each trace function.

Table 2.2 AUD Trace Acquisition Mode

Type	Mode	Description
Continuous trace occurs	Realtime trace	When the trace information is being generated intensely that the output from the AUD pin incapable of keeping up, the CPU temporarily suspends the output of trace information. Therefore, although the user program is run in real time, the acquisition of some trace information might not be possible.
	Non realtime trace	When trace information is being generated so intensely that the output from the AUD pin is incapable of keeping up, CPU operations are temporarily suspended and the output of trace information takes priority. In such cases, the realtime characteristics of the user program are lost.
Trace buffer full	Trace continue	This function overwrites the latest trace information to store the oldest trace information.
	Trace stop	After the trace buffer becomes full, the trace information is no longer acquired. The user program is continuously executed.

(3) Trace display contents

When the program breaks, the following trace results are displayed in the [Trace] window.

- PTR: The trace-buffer pointer (+0 from the last instruction to have been executed)
- IP: Indicates the number of cycles that have elapsed since the latest trace information was gathered. For branch instructions, the branch source and destination are counted together as one.
- Type: Displays the type of trace acquisition information.
- Address: Displays the addresses from which the trace data was acquired.
- Data: Displays the data acquired in the trace. For information without data, displays '********'.
- Instruction, Source, Label: Displays the mnemonic of the instruction at the trace acquisition address, along with the corresponding source code and label information. Double-clicking on the [Source] column moves the cursor to the corresponding position in the [Editor] window.

The Type, Address, and Data columns have different meanings according to the type of AUD trace that has been selected.

Table 2.3 [Trace Window] Display Contents

Trace Type	Type Column	Address Column	Data Column
Branch trace	BRANCH	Branch source address	No display
	DESTINATION	Branch destination address	No display
Window trace[*1]	MEMORY	Memory access address	Memory access data
Software trace[*1]	S_TRACE	Trace(x) function execution address	Variable x data
Data lost[*1, *2]	LOST	No display	No display
CPU wait generation[*1, *2]	CPU-WAIT	No display	No display

Notes: 1. Not displayed in the internal trace.

2. According to the device being debugged, there may be no output for the [Lost] or [CPU-WAIT] type. In such a case, it is not possible to clarify whether the trace data was not output in time or the CPU generated a wait state for the output trace data.

The following items will be displayed, according to the device to be debugged.

For specifications of the individual products, refer to the additional document, ″Supplementary Information on Using the SHxxxx″, or the online help.

- PTR: The trace-buffer pointer (+0 from the last instruction to have been executed)
- IP: Indicates the number of cycles that have elapsed since the latest trace information was gathered. For branch instructions, the branch source and destination are counted together as one.
- Master: Type of bus master that accessed the memory.
- Type: Displays the type of trace acquisition information.
- Branch Type: Branch type (only displayed for a branch trace)
  For an AUD trace, this item is only displayed if the PPC option has been enabled.
- Bus: Displays which bus was accessed.
- R/W: Displays whether the access involved reading or writing.
- Address: Displays the addresses from which the trace data was acquired.
- Data: Displays the data acquired in the trace.
- PPC: Output from a performance counter
- Instruction, Source, Label: Displays the mnemonic of the instruction at the trace acquisition address, along with the corresponding source code and label information. Double-clicking on the [Source] column moves the cursor to the corresponding position in the [Editor] window.

The Type, BUS, R/W, Address, and Data columns have different meanings according to the type of AUD trace that has been selected.

Table 2.4 [Trace Window] Display Contents

Trace Type	Type Column	BUS Column	R/W Column	Address Column	Data Column
Branch trace	BRANCH*1	No display	No display	Branch source address*1	No display
	DESTINATION	No display	No display	Branch destination address	No display
Memory-range access trace	MEMORY	Bus through which access is proceeding	Read/write	Memory access address	Memory access data*1
Software trace	S_TRACE	No display	No display	Trace(x) function execution address	Variable x data
System bus trace	MEMORY	No display	Read/write	Memory access address	Memory access data (write only)*1
Data lost*2	LOST	No display	No display	No display	No display
CPU wait generation*2	CPU-WAIT	No display	No display	No display	No display

Notes: 1. Not displayed when the PPC option is in use.

2. According to the device being debugged, there may be no output for the [Lost] or [CPU-WAIT] type. In such a case, it is not possible to clarify whether the trace data was not output in time or the CPU generated a wait state for the output trace data.

2.2.3 Memory Output Function of Trace Data

In some devices to be debugged, trace data can be written to the specified memory range. The data is read from the memory range written in the [Trace] window and the result is then displayed.

Note: Do not specify the program area as the memory in the specified range is overwritten.

2.2.4 Useful Functions of the [Trace] Window

The trace window provides the following useful functions.

(1) Searches for the specified data.

(2) Extracts the specified data.

(3) Filters and displays again the specified data.

(4) Supplements the information from the branch destination address to the next branch source address.

For the usage of those functions, refer to section 5.7, Viewing the Trace Information.

(5) Changes the trace settings during user program execution.

In some devices to be debugged, trace settings can be changed during user program execution. For details on the specifications of each product, refer to the online help.

2.3 Break Function

The emulator has the following three break functions.

(1) Hardware break function

Uses a break controller incorporated in the device.

The access address, instruction fetch address, data, or bus cycle condition can be set. The logical address is the address condition.

This function can be also set from the [Event] column in the [Editor] or [Disassembly] window. For the setting, refer to section 5.3, Downloading a Program.

Note: In some devices to be debugged, hardware break settings can be changed during user program execution. For details on the specifications of each product, refer to the online help.

(2) PC break function (BREAKPOINT)

Breaks when the dedicated instruction at the specified address that has been replaced is executed. This function cannot be set at a place other than RAM or internal flash memory area since a memory write occurs.

It can also be set when the [S/W breakpoint] column for the line to be set is double-clicked in the [Editor] or [Disassembly] window.

(3) Forced break function

Forcibly breaks the user program.

2.4 Performance Measurement Function

The emulator has a following performance measurement function.

2.4.1 Function for Measuring the Number of Cycles from Point to Point

This function applies a counter in the device to measure the number of cycles from one specified condition being satisfied until a next specified condition is satisfied.

Not only the number of cycles but also various items such as the number of cache misses or of TLB misses can be measured according to the supported devices.

Note: Items to be measured differ according to the product and some products do not support this function. For details on the specifications of each product, refer to the online help.

2.5 Memory Access Functions

The emulator has the following memory access functions.

(1) Memory read/write function

[Memory] window: The memory contents are displayed in the window. Only the amount specified when the [Memory] window is opened can be read. Since there is no cache in the emulator, read cycles are always generated. If the memory is written in the [Memory] window, read cycles in the range displayed in the [Memory] window will occur for updating the window. When the [Memory] window is not to be updated, change the setting in [Lock Refresh] from the popup menu.

me command: A command line function that reads or writes the specified amount of memory at the specified address.

(2) User program downloading function

A load module registered in the workspace can be downloaded. Such module can be selected from [Download Module] in the [Debug] menu. Downloading is also possible by a popup menu that is opened by right-clicking on the mouse at the load module in the workspace. The user program is downloaded to the RAM or internal flash memory.

When downloading to the flash memory that has not been within the MPU, select [Emulator] from the [Setup] menu, open the [Configuration] window, and perform required settings on the [Loading flash memory] page.

This function also downloads information required for source-level debugging such as debugging information.

(3) Memory data uploading function

The specified amount of memory from the specified address can be saved in a file.

(4) Memory data downloading function

The memory contents saved in a file can be downloaded. Select [Load] from the popup menu in the [Memory] window.

(5) Displaying the variable contents

The variable contents specified in the user program are displayed.

(6) Monitoring function

In some devices to be debugged, memory contents can be monitored during user program execution. For details on the specifications of each product, refer to the online help.

(7) Other memory operation functions

Other functions are as follows:

- Memory fill
- Memory copy
- Memory save
- Memory verify
- Memory search
- Internal I/O display
- Cache table display and edit (only for devices incorporating caches)
- TLB table display or edit (only for devices incorporating MMU)
- Displaying label and variable names and their contents

For details, refer to the online help.

Notes: 1. Memory access during user program execution:
When memory is accessed from the memory window, etc. during execution of the user program, execution stops for the memory access and is then resumed. Therefore, realtime emulation cannot be performed.

The stopping time of the user program is as follows:

Environment:

Host computer: CORE™2 CPU T7600 2.33 GHz

SH7265: CPU clock 66.6 MHz

JTAG clock: 2.5 MHz

When a one-byte memory is read from the command-line window, the stopping time will be about 70 ms.

2. Memory access during user program break:
The program can also be downloaded for the flash memory area by the emulator. Other memory write operations are enabled for the RAM area and the internal flash memory. Therefore, an operation such as memory write or BREAKPOINT should be set only for the RAM area and the internal flash memory. When the memory area can be read by the MMU, do not perform memory write, BREAKPOINT setting, or downloading.

3. Cache operation during user program break:
When cache is enabled in the device incorporating a cache, the emulator accesses the memory by the following methods:
   - At memory write: Writes through the cache, then writes to the memory or uses the OCBWB instruction.
   - At memory read: Does not change the cache write mode that has been set.
   - At memory verify: Disables the cache for verification read.

   Therefore, when memory read or write is performed during user program break, the cache state will be changed.
   In some devices to be debugged, the emulator accesses the memory by the following methods:
   - At memory write: Writes to the cache, then issues an external single write. The LRU is not updated.
   - At memory read: Reads memory from the cache. The LRU is not updated.

2.6 Stack Trace Function

The emulator uses the information on the stack to display the names of functions in the sequence of calls that led to the function to which the program counter is currently pointing. This function can be used only when the load module that has the Dwarf2-type debugging information is loaded. For the usage of this function, refer to section 6.21 and 7.22 Stack Trace Function.

2.7 User-interrupt Open Function during User Program Break

Some devices to be debugged open all interrupts while executing emulation to users. During a user program break, it is possible to specify the mode whether or not the interrupt processing is executed.

2.8 Online Help

An online help explains the usage of each function or the command syntax that can be entered from the command line window.

Select [Emulator Help] from the [Help] menu to view the emulator help.

Section 3 Preparation before Use

3.1 Emulator Preparation

Unpack the emulator and prepare it for use as follows:

⚠ WARNING

READ the reference sections shaded in figure 3.1 before using the emulator product. Incorrect operation will damage the user system and the emulator product. The USER PROGRAM will be LOST.

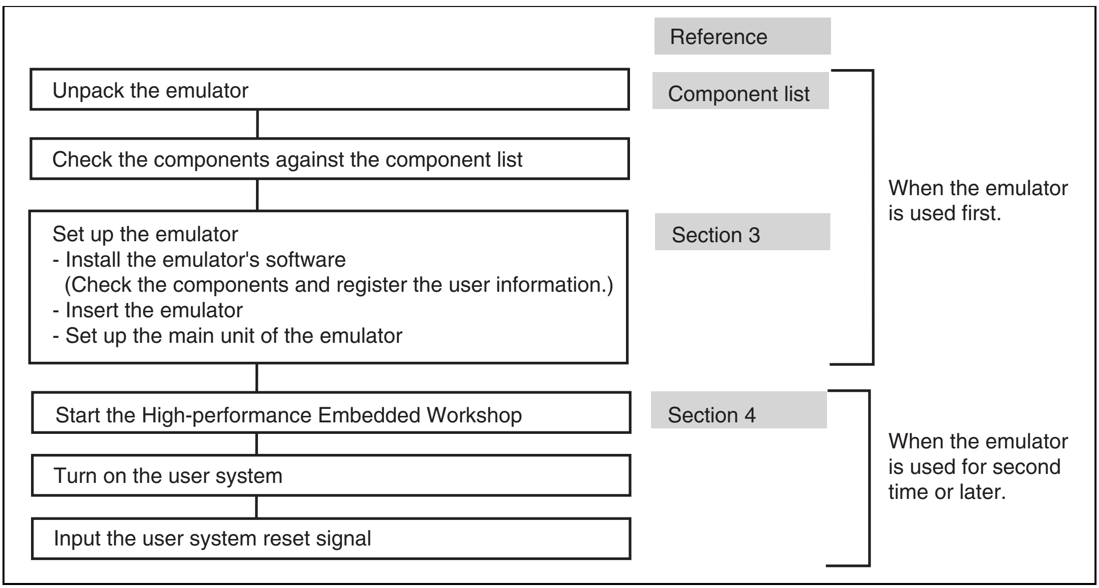


Figure 3.1 Emulator Preparation Flow Chart

3.2 Emulator Hardware Configuration

As shown in figure 3.2, the emulator consists of an emulator, a USB cable, and a user system interface cable. The emulator is connected to the host computer via USB 1.1, and also to the USB port conforming to USB 2.0.

Figure 3.2 Emulator Hardware Configuration (when the 38-pin Type Cable is Used)

The names of each section of the emulator are explained next.

Emulator Top View:

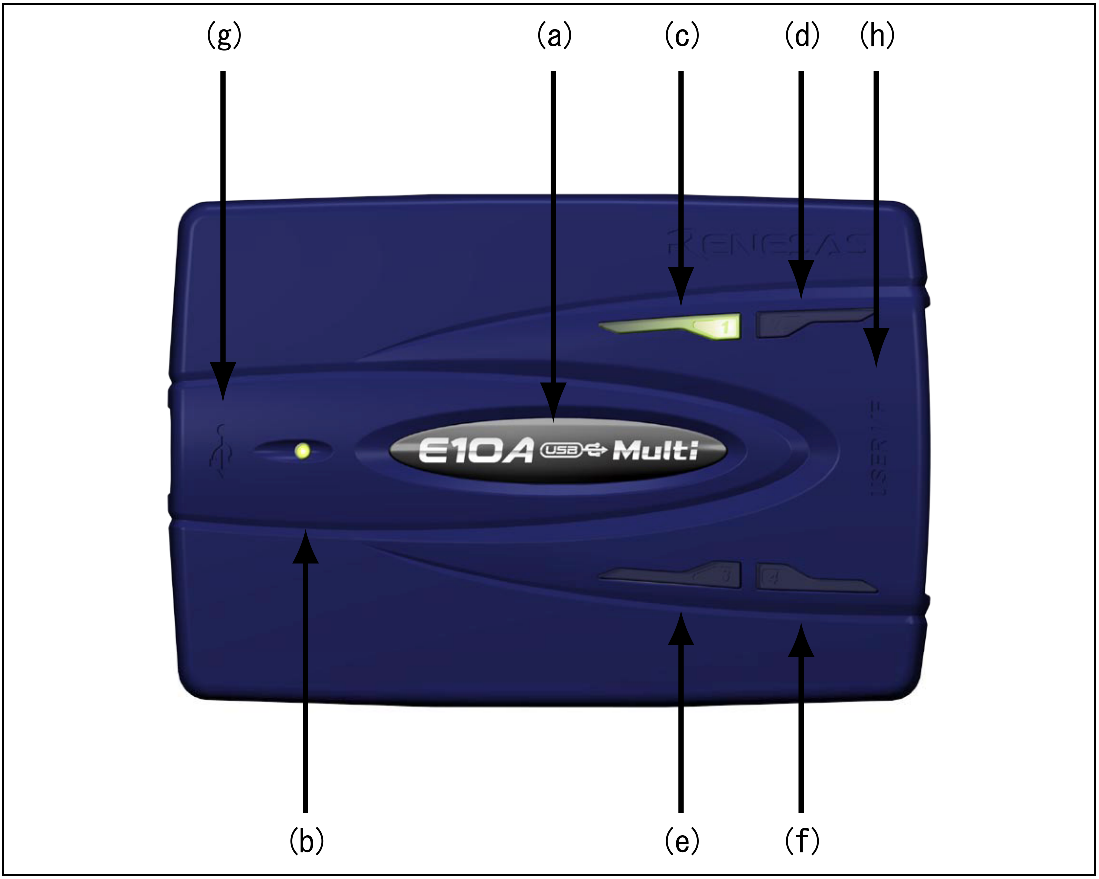


Figure 3.3 Emulator Top View

(a) E10A-USB logo plate: A black plate is dedicated for the emulator is provided to be easily distinguished from other E-series emulators.

(b) ACTION LED: A circled LED. Marked 'ACT'. When this LED is lit, the E10A-USB control software is in operation.

(c) RUN LED: Marked '1'. When this LED is lit, the user program is in operation.

(d) ACT LED: Marked '2'. When this LED is lit, the E10A-USB is connecting to the micro computer.

(e) Core switch LED: Marked '3'. When this LED is lit, the E10A-USB switches the micro computer to be controlled.

(f) UVCC LED: Marked '4'. When this LED is lit, the E10A-USB is supplied the UVCC .

(g) Host connector: Marked '⭠'. A connector for the host computer is provided at the side of this mark.

(h) User connector: Marked 'USER I/F'. A connector for the user system interface cable is provided at the side of this mark.

Note: Even if the LED is not lit, the USB is not disconnected or malfunctioned.

Emulator Host-side View:

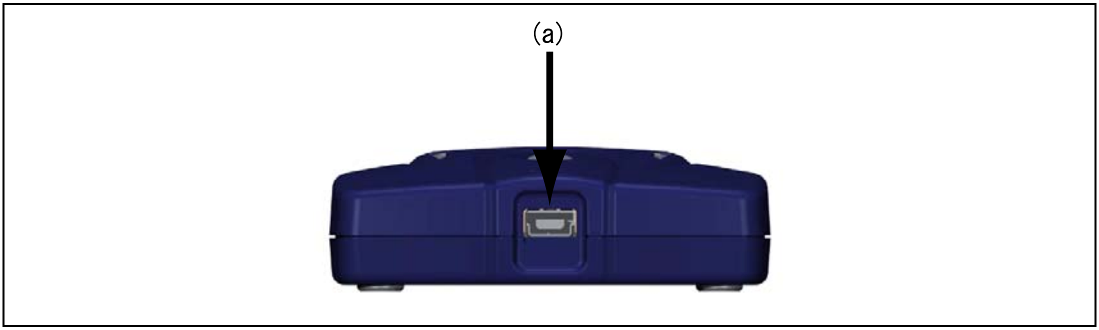


Figure 3.4 Emulator Host-side View

(a) Host-side connector: A USB connector for the host computer. Be sure to connect the provided USB cable.

Emulator User-side View:

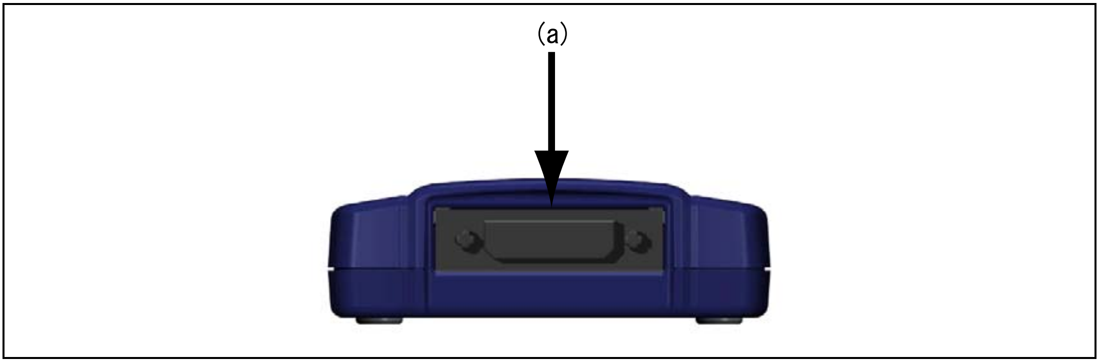


Figure 3.5 Emulator User-side View

(a) User-side connector: A user system interface cable is connected.

Emulator Bottom View:

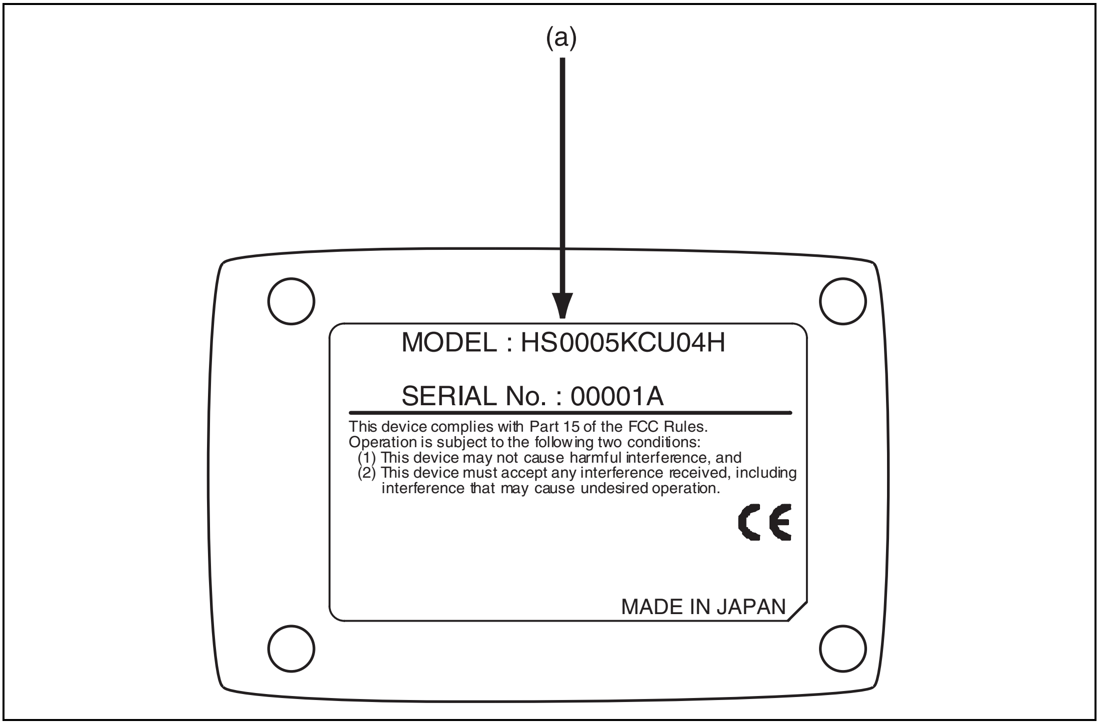


Figure 3.6 Emulator Bottom View

(a) Label for product management: The serial number, revision, and safety standard, etc. of the emulator are written to. The contents differ depending on the time when you purchased the product.

3.3 CD-R

The root directory of the CD-R contains a setup program for installing the emulator's software. The folders contain the files and programs listed below.

Table 3.1 Contents of the CD-R Directories

Directory Name	Contents	Description
Dlls	Microsoft® runtime library	A runtime library for the High-performance Embedded Workshop. The version is checked at installation and this library is copied to the hard disk as part of the installation process.
Drivers	E10A-USB emulator driver	USB drivers for the E10A-USB emulator.
Help	Online help for the E10A-USB emulator	An online help file. This is copied to the hard disk as part of the installation process.
Manuals	E10A-USB emulator manuals	E10A-USB emulator user's manuals. They are provided as PDF files.

3.4 Installing Emulator's Software

Launch the installation manager by executing HewInstMan.exe from the root directory of the CD-R. Install the software in accord with the cues given by the installation manager.

Note: When a driver is installed in Windows® XP, a warning message on the Windows® logo test may be displayed, but it is not a problem. Select [Continue Anyway] to proceed with driver installation.

3.5 Connecting the Emulator to the Host Computer

This section describes how to connect the emulator to the host computer. For the position of each connector of the emulator, refer to section 3.2, Emulator Hardware Configuration.

Notes: 1. When the [Add New Hardware] wizard is displayed, select the [Search for the best driver for your device. (Recommended)] radio button.

2. Be sure to install the software for the emulator before putting the emulator in place.

⚠ WARNING
Always switch OFF the emulator product and the user system before connecting or disconnecting any CABLES except for the USB interface cable. 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.

The emulator is connected to the host computer via the USB 1.1, and also to the USB port conforming to USB 2.0. Figure 3.7 shows the system configuration.

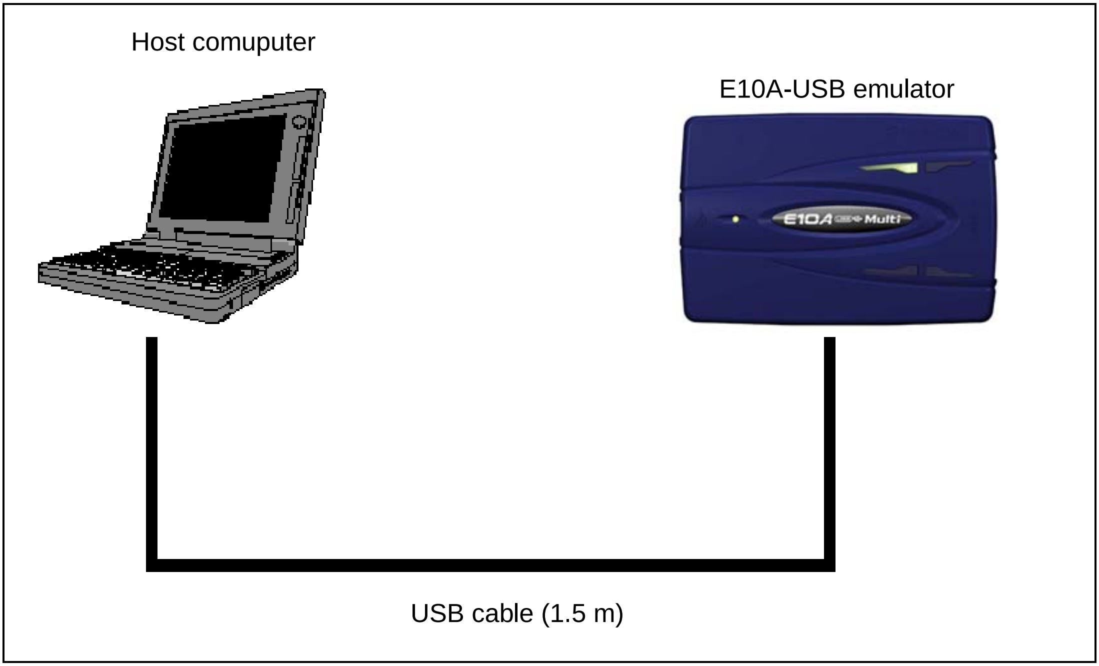


Figure 3.7 System Configuration when Connecting the Emulator to the Host Computer

3.6 Connecting the Emulator to the User System

Use the procedure below to connect the emulator to the user system with the user system interface cable, or to disconnect them when moving the emulator or the user system.

1. Check that the host computer is turned off or the emulator is not connected to the host computer with the USB cable.
2. Connect the user system interface cable to the user-side connector of the emulator.
3. Connect the USB cable to the host-side connector of the emulator.

Figure 3.8 shows the position of the connector.

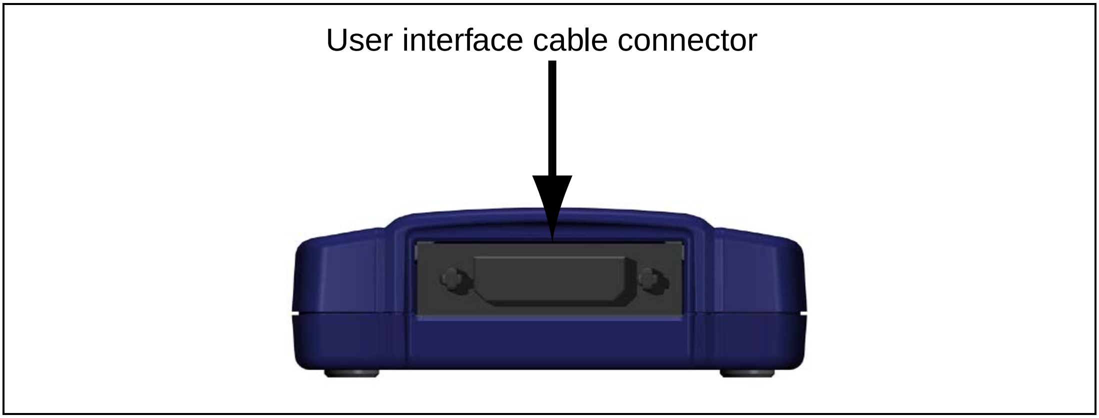


Figure 3.8 Position of the Connector

(1) The connector must be installed to the user system. Table 3.2 shows the recommended connector for the emulator.

Table 3.2 Recommended H-UDI Port Connector

Connector	Type Number	Manufacturer	Specifications
14-pin connector	2514-6002	Minnesota Mining & Manufacturing Ltd.	14-pin straight type
38-pin connector	2-5767004-2	Tyco Electronics Corporation	38-pin Mictor connector

Notes: 1. When designing the 14-pin connector layout on the user board, do not place any components within 3 mm of the H-UDI port connector.
When designing the 36-pin connector layout on the user board, do not connect other signal lines to the H-UDI port connector.
2. The H-UDI is an interface compatible with the Joint Test Action Group (JTAG) specifications.

(2) The pin assignments of the connector are shown in section 2, in the additional document, ″Supplementary Information on Using the SHxxxx″.

(3) Connect pins 5 and GND bus read located in the center of the H-UDI port connector (when using the 38-pin user system interface cable) and pins 9, 10, 12, 13, and 14 (when using the 14-pin user system interface cable) of the H-UDI port connector to GND firmly on the PCB. These pins are used as electrical GND and to monitor the connection of the H-UDI port connector. Note the pin assignments of the H-UDI port connector.

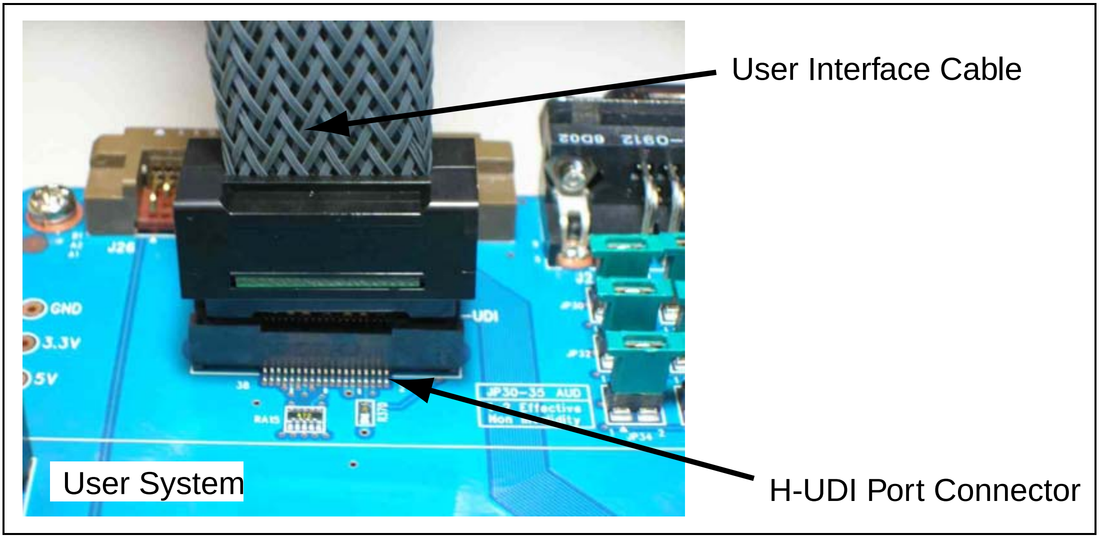


Figure 3.9 Connecting the User System Interface Cable to the User System when the 38-pin Type Connector is Used

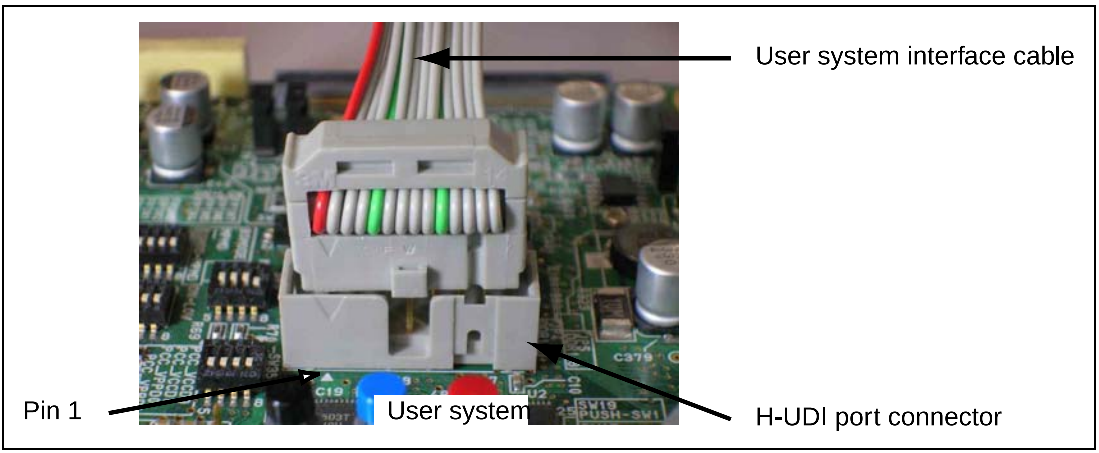


Figure 3.10 Connecting the User System Interface Cable to the User System when the 14-pin Type Connector is Used

CAUTION

Note that the pin number assignments of the connector differ from those of the connector manufacturer.

Notes: 1. Connection of the signals differs depending on the package. For details, refer to the device's pin assignments.

2. The range of communication that the emulator operates at is different depending on the device used.

3. To connect the signals from the connector, refer to section 1, in the additional document, ″Supplementary Information on Using the SHxxxx″.

4. When developing user systems, do not connect the TDI and TDO signals of the device to the boundary scan loop, or separate them by using a switch (figure 3.11).

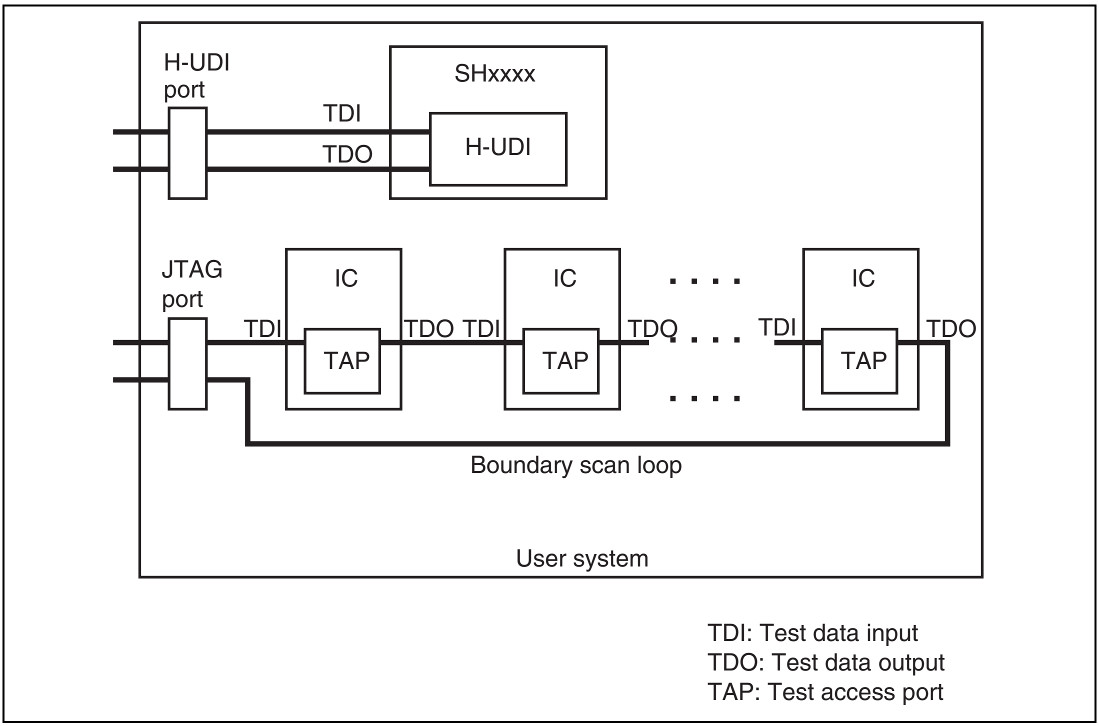


Figure 3.11 User System Example

3.7 Connecting System Ground

⚠ WARNING

Separate the frame ground from the signal ground at the user system. 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 emulator's signal ground is connected to the user system's signal ground. In the emulator, the signal ground and frame ground are connected. In the user system, connect the frame ground only; do not connect the signal ground to the frame ground (figure 3.12).

If it is difficult to separate the frame ground from the signal ground in the user system, set the GND for DC power input (AC adapter) of the host computer and the frame ground of the user system as the same potential. If the GND potential is different between the host computer and the target system, an overcurrent will flow in the low-impedance GND line and thin lines might be burned.

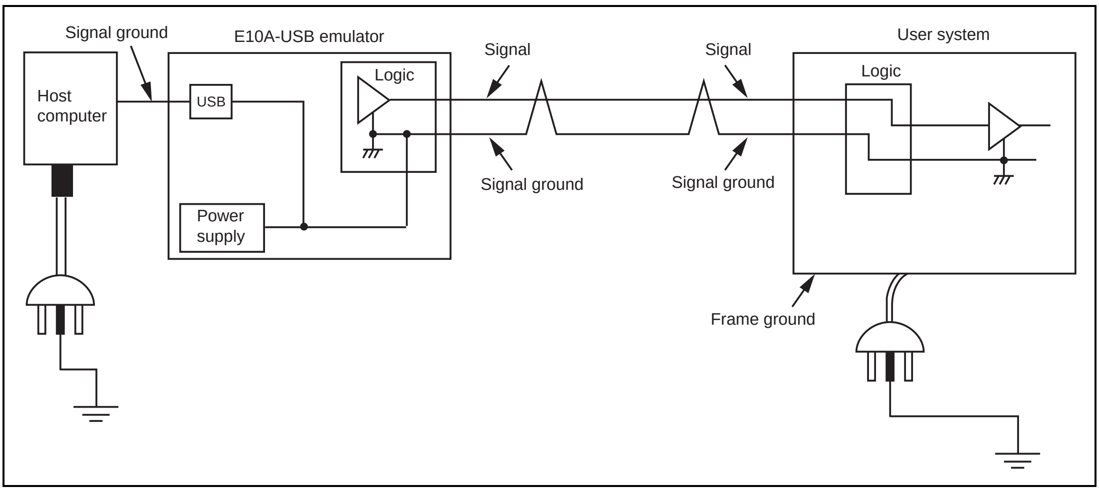


Figure 3.12 Connecting System Ground

3.8 Interface Circuits in the Emulator

Figures 3.13 and 3.14 show interface circuits in the emulator. Use them as a reference to determine the value of the pull-up resistance.

Note: The 74LVC2G125 and 74LVC2T45 operate at VCC (1.8 to 5.0 V) from the H-UDI port connector.

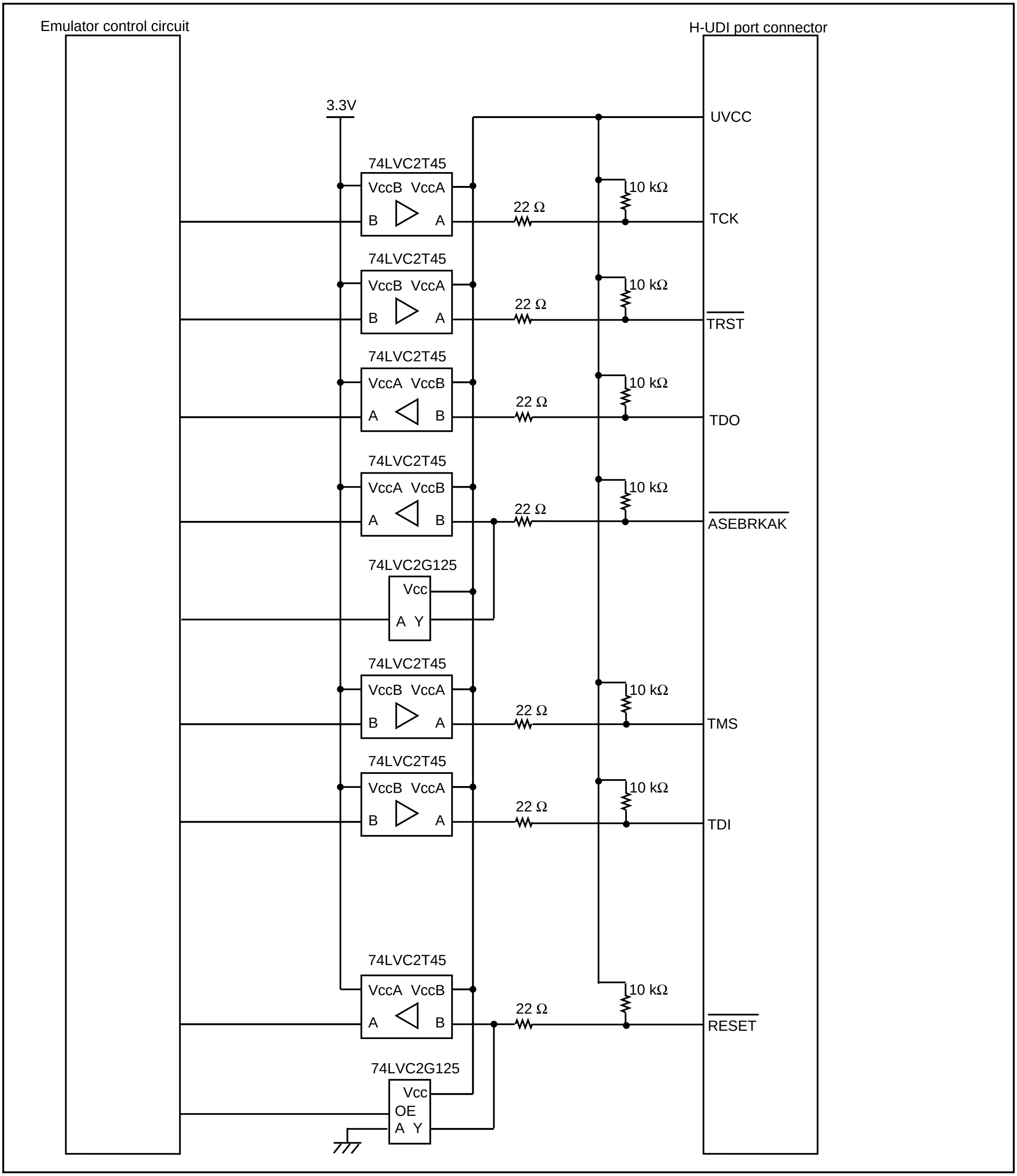


Figure 3.13 Interface Circuits in the Emulator (H-UDI)

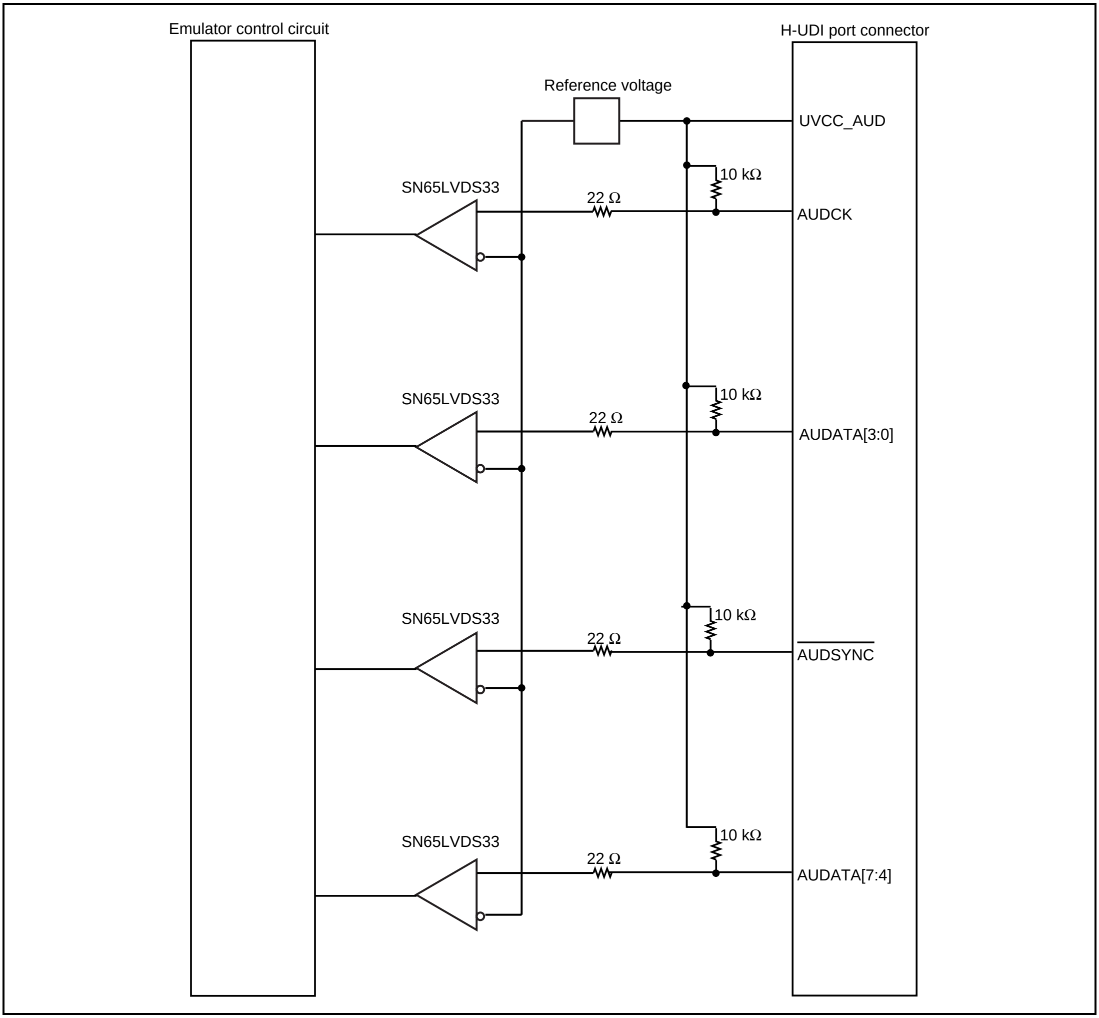


Figure 3.14 Interface Circuits in the Emulator (AUD)

3.9 System Check

Before executing software, follow the procedure below to check that the emulator is connected correctly. At this point, use the tutorial workspace that is provided with the product and start up the High-performance Embedded Workshop for CPU0 only.

Refer to section 4, Preparations for Debugging, for the other activating method to create a new project or use an existing workspace.

1. Connect the emulator to the host computer.
2. Connect the user system interface cable to the connector of the emulator.
3. Connect the user system interface cable to the connector in the user system.
4. Select [Renesas] → [High-performance Embedded Workshop] → [High-performance Embedded Workshop] from the [Program] item in the [Start] menu.
5. [Welcome!] dialog box is displayed.

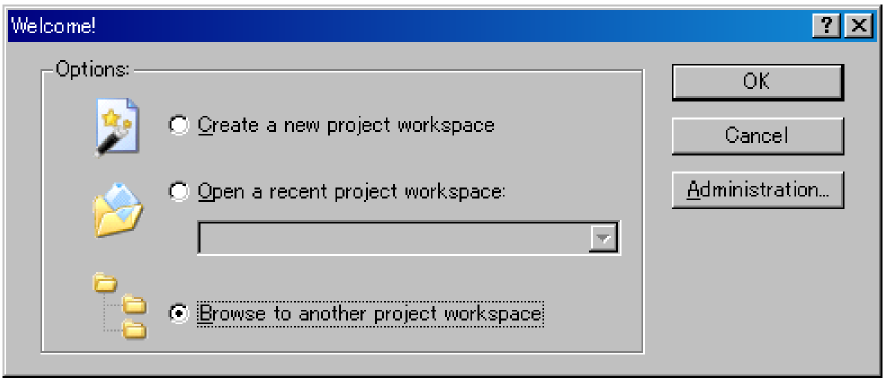


Figure 3.15 [Welcome!] Dialog Box

[Create a new project workspace] radio button:	Creates a new workspace.
[Open a recent project workspace] radio button:	Uses an existing workspace and displays the history of the opened workspace.
[Browse to another project workspace] radio button:	Uses an existing workspace; this radio button is used when the history of the opened workspace does not remain.

To use a workspace for the tutorial, select the [Browse to another project workspace] radio button and click the [OK] button.

When the [Open workspace] dialog box is opened, specify the following directory:
<Drive where the OS is installed>:

\WorkSpace\Tutorial\E10A-USBM\xxxx\xxxx\Tutorial\CPU0

(xxxx represents the device group name).

After the directory has been specified, select the following file and click the [Open] button.

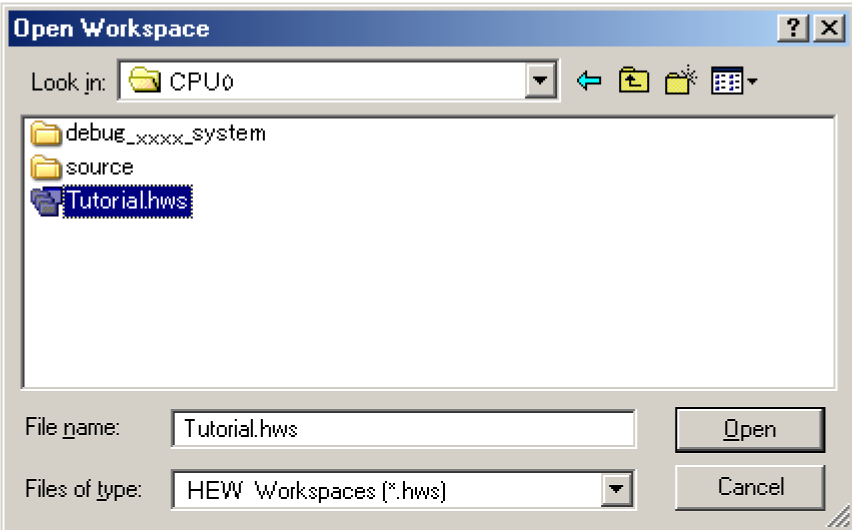


Figure 3.16 [Open The Workspace] Dialog Box

6. The [CPU Select] dialog box or [Select Emulator mode] dialog box is displayed.

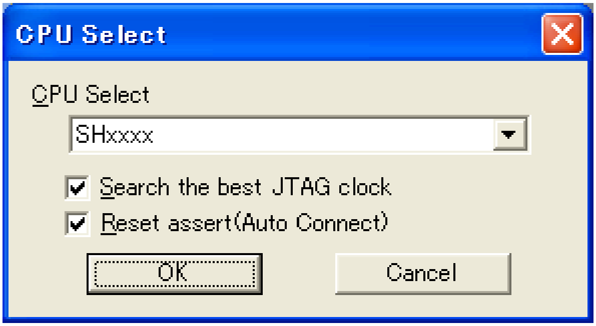


Figure 3.17 [CPU Select] Dialog Box

The [CPU Select] dialog box has the following options.

- [Search the best JTAG clock] check box

  Selects finding of the fastest usable JTAG clock value, and starting up with this as the initial value.
- [Reset assert (Auto Connect)] check box

  A reset signal is generated by the emulator, and steps 12 and 14 are skipped.
  If the [Reset assert (Auto Connect)] option is in use, turn on the power to the user system at this stage.

CAUTION

Do not to use the [Reset assert (Auto Connect)] option unless the port connector is properly connected as shown in Section 1.5, Recommended Circuit between the H-UDI Port Connector and the MPU of the additional document, ″Supplementary Information on Using the SHxxxx″. Using the [Auto connect] option without the proper connection will damage the user system.

Select the device name in use from the [CPU Select] drop-down list box and click on OK.

7. The [Select Emulator mode] dialog box is displayed and indicates the device group to be used.

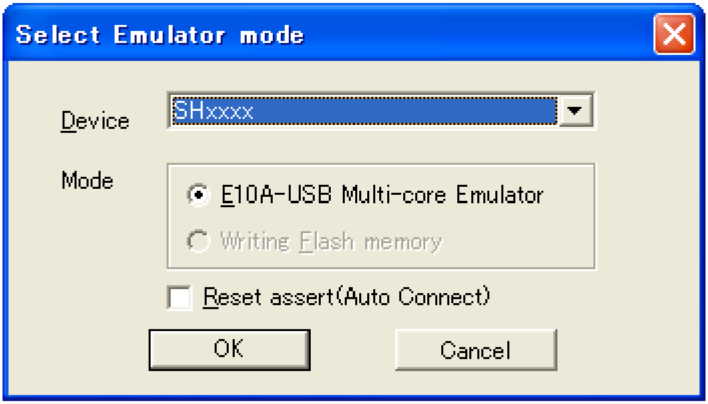


Figure 3.18 [Select Emulator mode] Dialog Box

The [Select Emulator mode] dialog box has the following options.

- [Reset assert (Auto Connect)] check box
  A reset signal is generated by the emulator, and steps 12 and 14 are skipped.

CAUTION

Do not to use the [Reset assert (Auto Connect)] option unless the port connector is properly connected as shown in Section 1.5, Recommended Circuit between the H-UDI Port Connector and the MPU of the additional document, ″Supplementary Information on Using the SHxxxx″. Using the [Auto connect] option without the proper connection will damage the user system.

Select the device name in use from the [Device] drop-down list box and click on OK.

8. The [Connecting] dialog box is displayed and connection of the emulator starts.

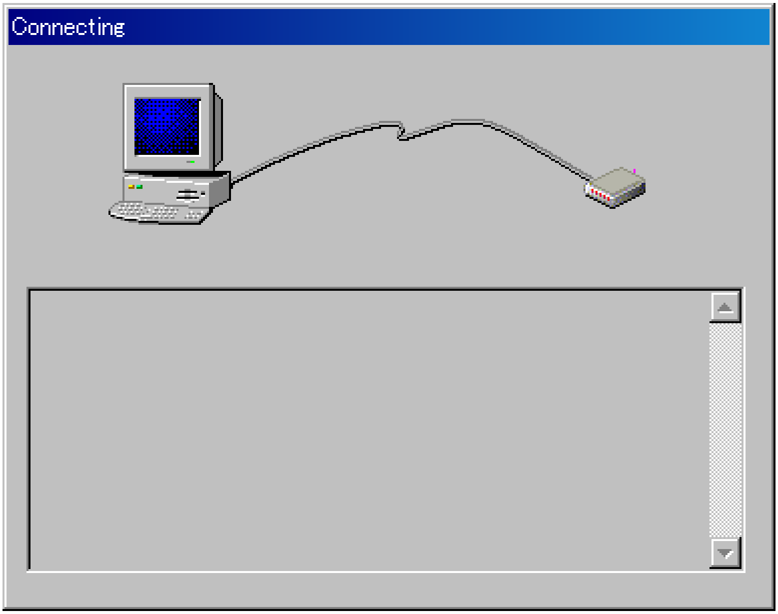


Figure 3.19 [Connecting] Dialog Box

9. The dialog box shown in figure 3.21 is displayed if no product groups have been installed in the emulator at the time of purchase or the emulator firmware has been set up for a different device group. The dialog box shown in figure 3.22 is displayed if an old version of the emulator firmware has been set up in the emulator. Clicking the [OK] button sets up the emulator firmware.

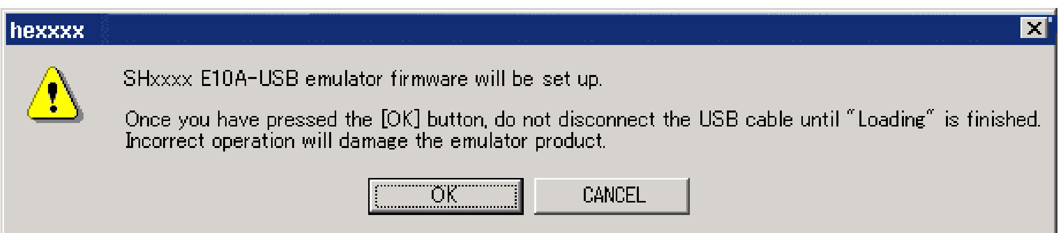


Figure 3.20 Dialog Box to Confirm Setting up of the Emulator Firmware

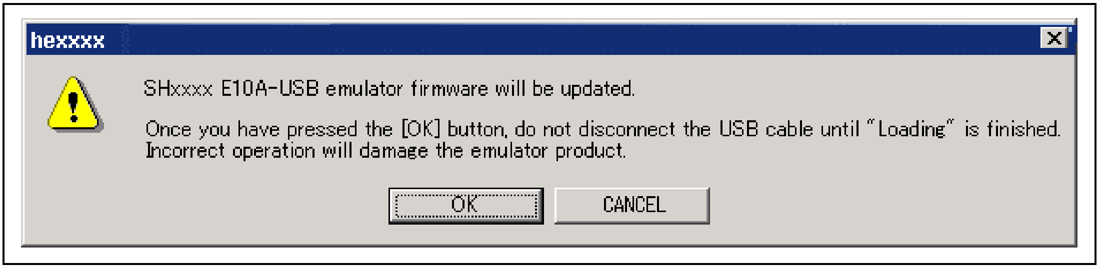


Figure 3.21 Dialog Box to Confirm Updating of the SHxxxx Emulator Firmware

CAUTION

The USB cable must not be disconnected until writing is complete.Early disconnection may damage the emulator.

10. When the Reset assert (Auto Connect) option is in use, the dialog box shown in figure 3.22 is displayed. When the power of the user system is turned on, figure 3.22 is not displayed.

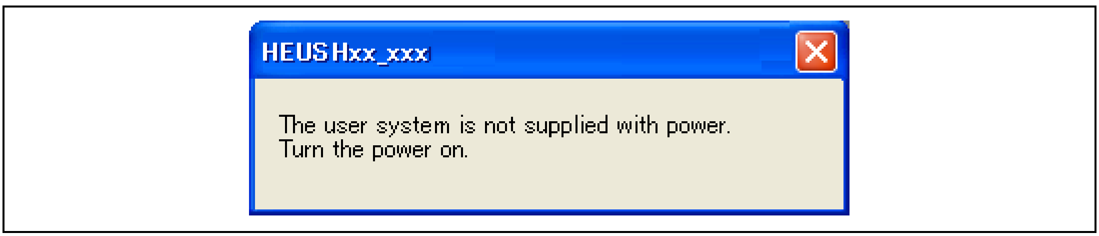


Figure 3.22 Dialog Box of the Turn the Power On Message

11. Power on the user system.
12. The dialog box shown in figure 3.23 is displayed.

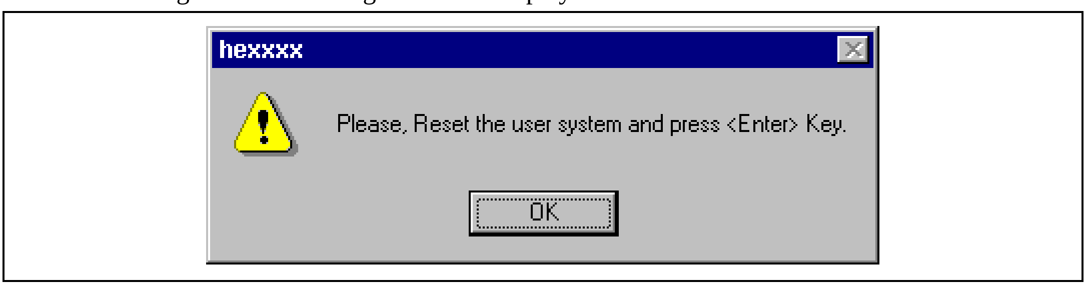


Figure 3.23 Dialog Box of the RESET Signal Input Request Message

13. Power on the user system.

14. Input the reset signal from the user system, and click the [OK] button.
15. If no reset signal is detected, the following dialog box is displayed.

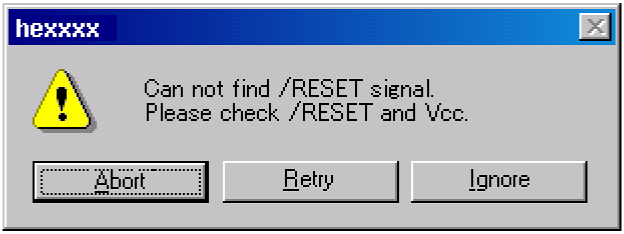


Figure 3.24 [Can not find /RESET signal] Dialog Box

16. If the "Connected" is displayed in the [Output] window of the High-performance Embedded Workshop for CPU0, starting up the emulator is completed.

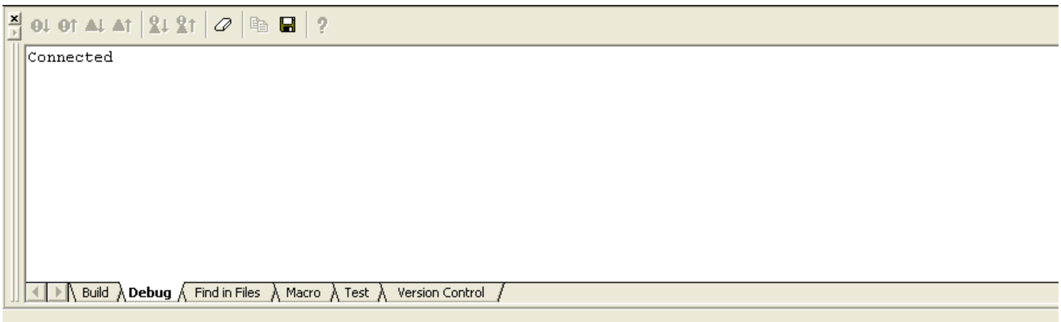


Figure 3.25 [Output] Window

Notes: 1 If the emulator is not initiated, the following dialog boxes shown in figures 3.26 through 3.32 will be displayed.

(a) If the following dialog box is displayed and the method 12 above is unavailable, the power of the user system may not be input or the RESET signal may not be input to the device. Check the input circuits for the power of the user system and the reset pin.

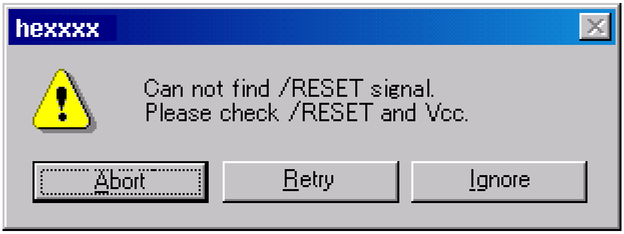


Figure 3.26 [Can not find /RESET signal] Dialog Box

(b) If the following dialog box is displayed, check that the H-UDI port connector on the user system is correctly connected.

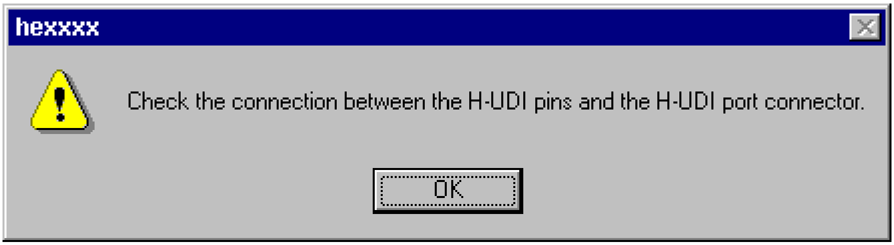


Figure 3.27 [Check the connection] Dialog Box

(c) If the following dialog box is displayed, the device may not correctly operate. Check if there are reasons for illegal device operation.

Figure 3.28 [COMMUNICATION TIMEOUT ERROR] Dialog Box

Figure 3.29 [INVALID ASERAM FIRMWARE!] Dialog Box

Figure 3.30 [Error JTAG boot] Dialog Box

(d) The following dialog box is displayed when the MCU cannot communicate with the emulator. The MCU may not operate correctly; check the MCU settings.

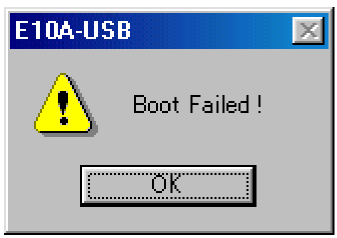


Figure 3.31 [Boot Failed!] Dialog Box

2. If an incorrect driver has been selected, the following dialog box will appear.

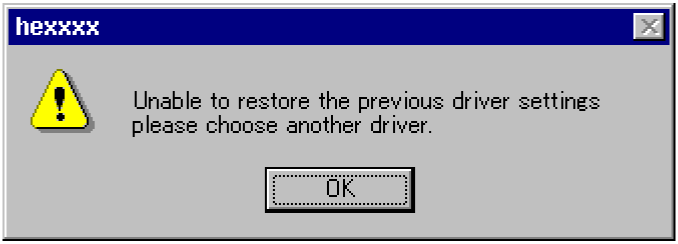


Figure 3.32 [Unable to restore the previous driver settings] Dialog Box

3. If the emulator is not activated due to other reasons, a message box corresponding to the status is displayed. Use the message as a reference to check the wiring on the board.

Section 4 Preparations for Debugging

4.1 Method for Activating High-performance Embedded Workshop

To activate the High-performance Embedded Workshop, follow the procedure listed below.

1. Connect the emulator to the host computer and the user system, then turn on the user system.
2. Select [High-performance Embedded Workshop] from [Renesas] -> [High-performance Embedded Workshop] of [Programs] in the [Start] menu.
3. The [Welcome!] dialog box is displayed.

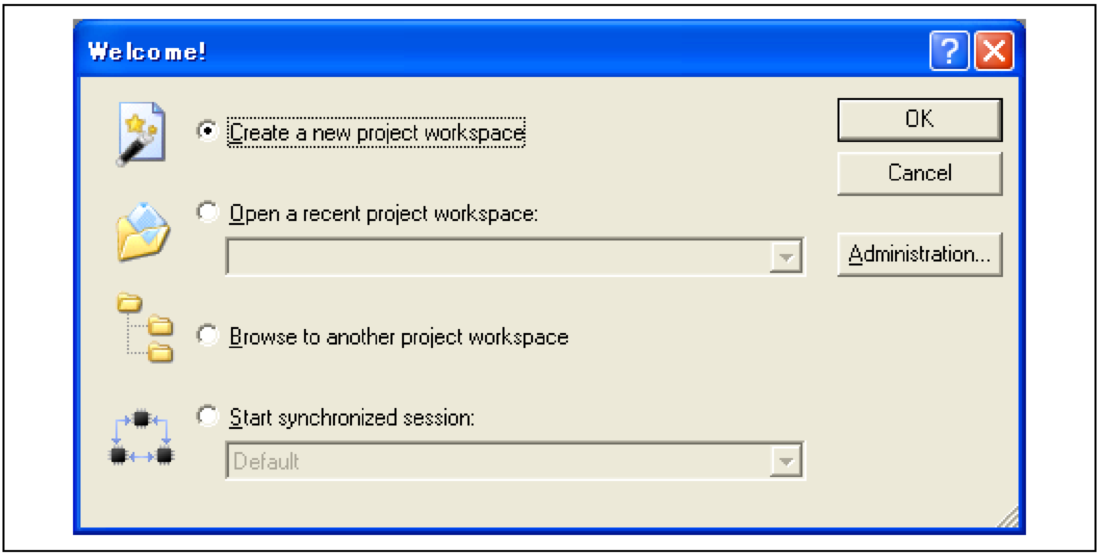


Figure 4.1 [Welcome!] Dialog Box

[Create a new project workspace] radio button:	Creates a new workspace.
[Open a recent project workspace] radio button:	Uses an existing workspace and displays the history of the opened workspace.
[Browse to another project workspace] radio button:	Uses an existing workspace; this radio button is used when the history of the opened workspace does not remain.
[Start synchronized session] radio button:	This button is selected for multi-core debugging and is only displayed when there is a history of multi-core debugging.

The following describes how to activate the High-performance Embedded Workshop when selecting [Create a new project workspace] without any toolchain, [Create a new project workspace] with a toolchain, and [Browse to another project workspace]. The [Open a recent project workspace] radio button is used to omit the operation for specifying the workspace file when [Browse to another project workspace] is selected.

4.1.1 Creating a New Workspace (Toolchain Not Used)

1. In the [Welcome!] dialog box that is displayed when the High-performance Embedded Workshop is activated, select [Create a new project workspace] radio button and click the [OK] button.

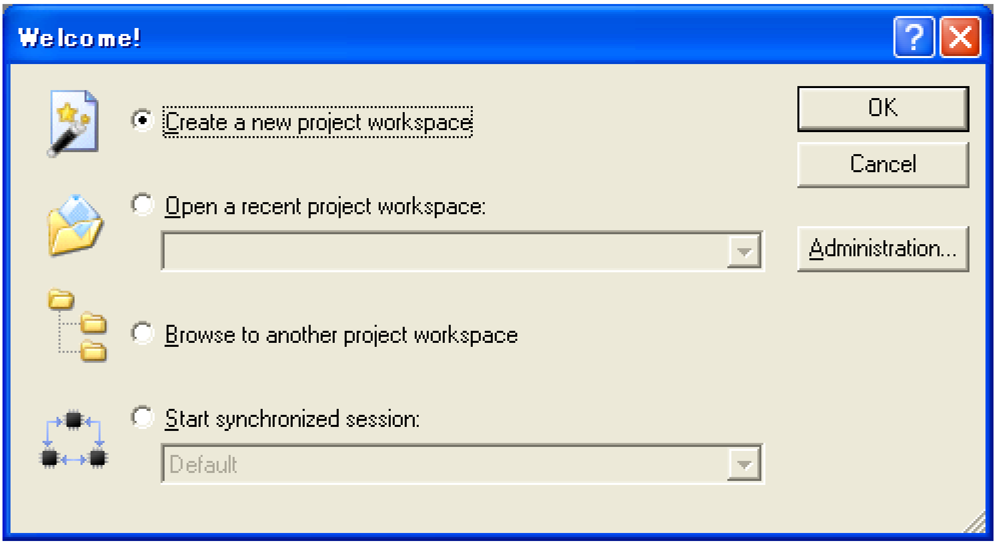


Figure 4.2 [Welcome!] Dialog Box

2. The Project Generator is started. In this section, we omit description of the settings for the toolchain.
   If you have not purchased the toolchain, the following dialog box is displayed.

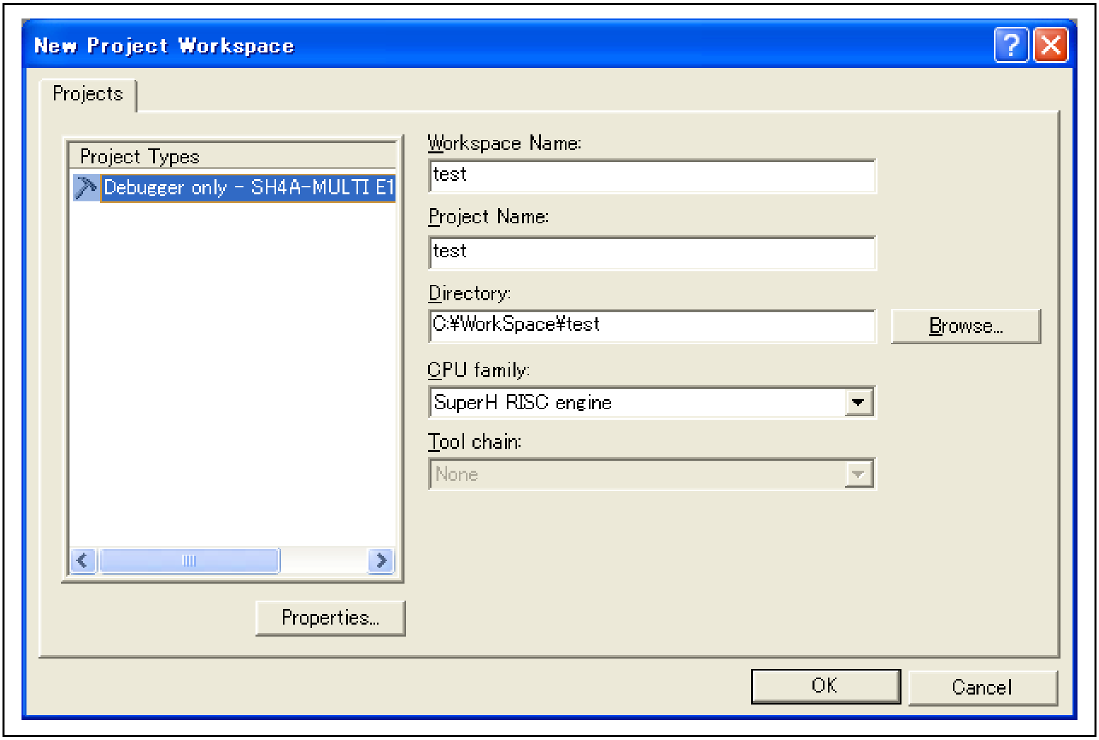


Figure 4.3 [New Project Workspace] Dialog Box

[Workspace Name] edit box: Enter the new workspace name. Here, for example, enter 'test'.

[Project Name] edit box: Enter the project name. When the project name is the same as the workspace name, it needs not be entered.

Other list boxes are used for setting the toolchain; the fixed information is displayed when the toolchain has not been installed.

3. Make the required setting for the toolchain. When the setting has been completed, the following dialog box is displayed.

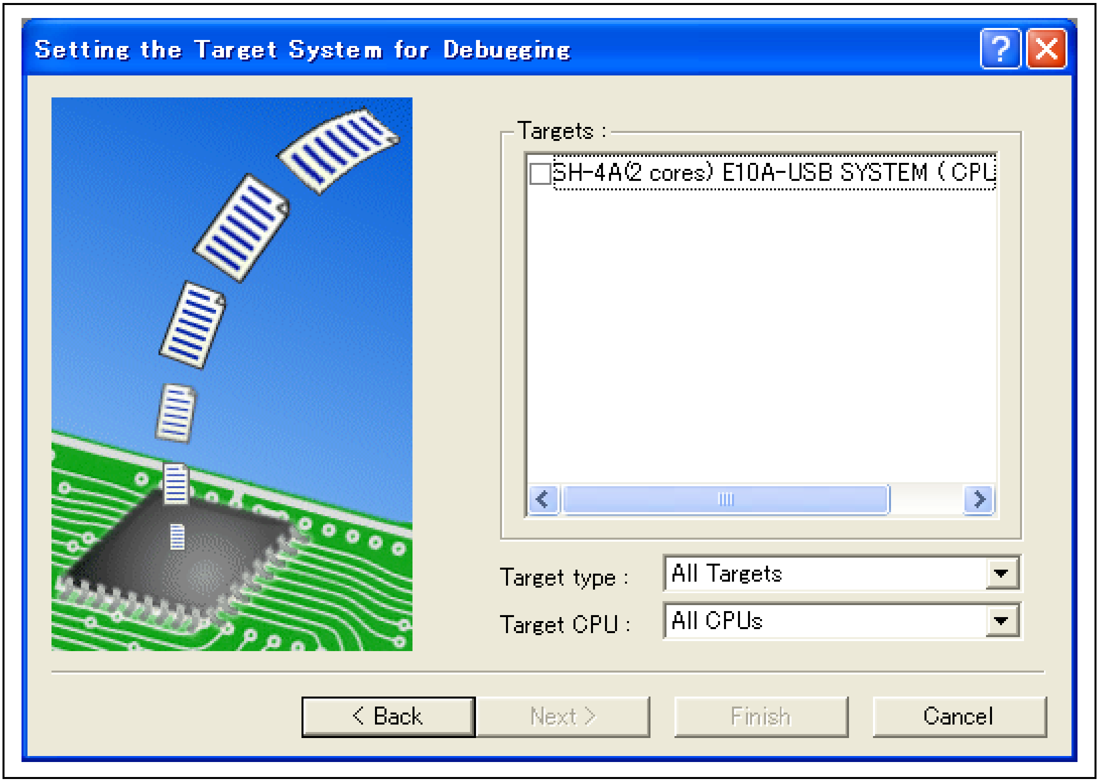


Figure 4.4 [Setting the Target System for Debugging] Dialog Box

Check the target emulator and click the [Next] button.

4. Set the configuration file name. The configuration file saves the state of High-performance Embedded Workshop except for the emulator.
   Select the CPU core numbers for debugging from the [Core] drop-down list box.

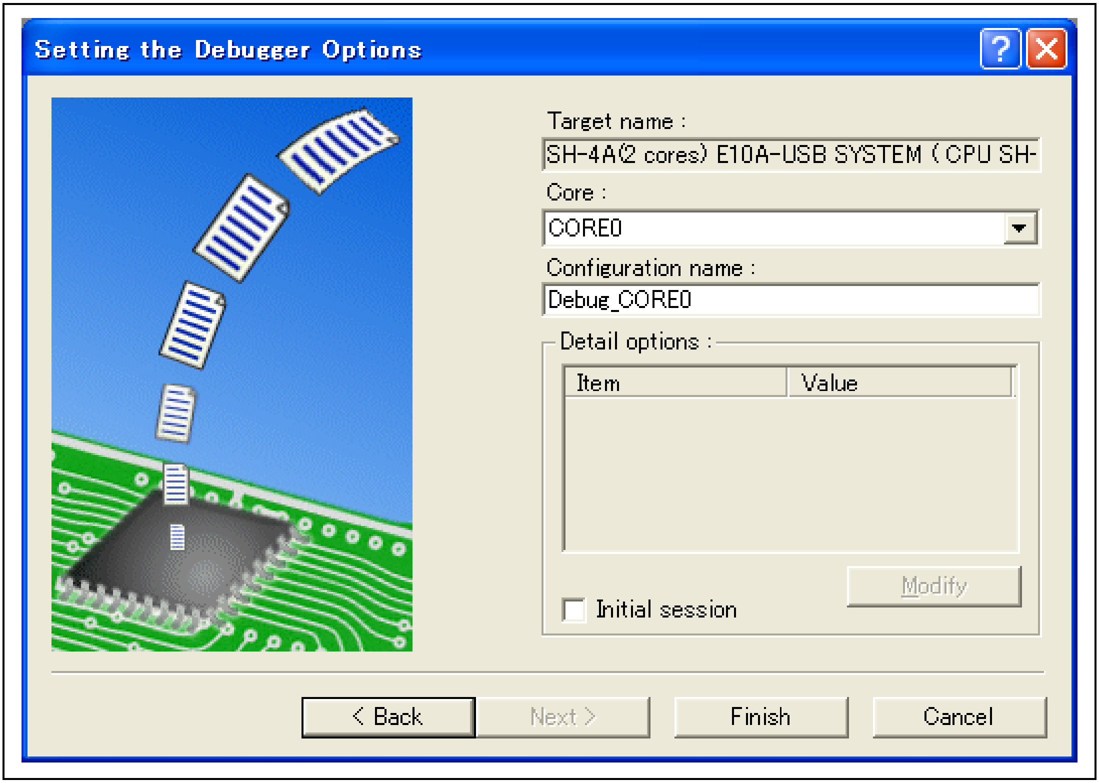


Figure 4.5 [Setting the Debugger Options] Dialog Box

This is the end of the emulator setting.

Click the [Finish] button to exit the Project Generator. The High-performance Embedded Workshop is activated.

5. After the High-performance Embedded Workshop has been activated, the emulator is automatically connected. For operation during connection, refer to section 3.9, System Check.

4.1.2 Creating a New Workspace (Toolchain Used)

1. In the [Welcome!] dialog box that is displayed when the High-performance Embedded Workshop is activated, select [Create a new project workspace] radio button and click the [OK] button.

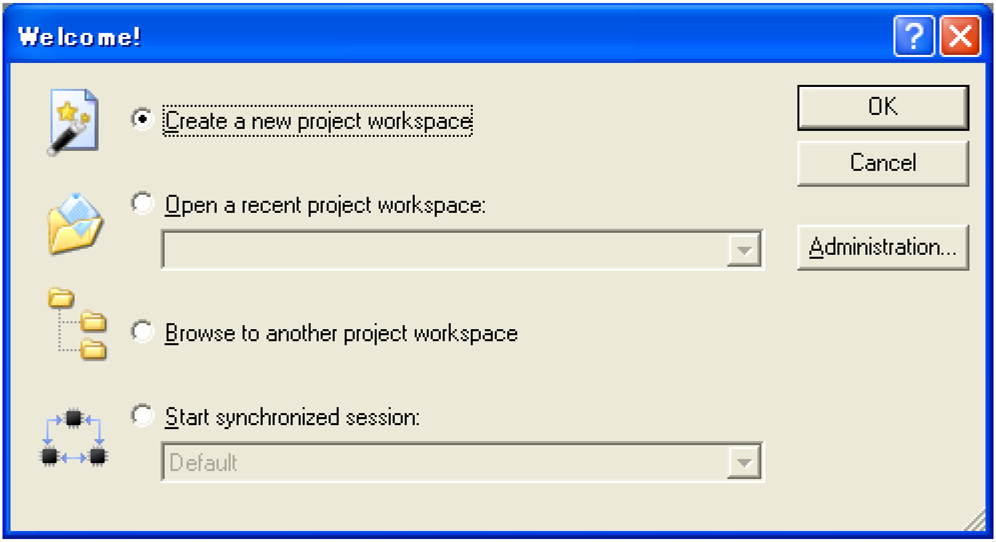


Figure 4.6 [Welcome!] Dialog Box

2. The Project Generator is started.
   If you have purchased the toolchain, the following dialog box is displayed.

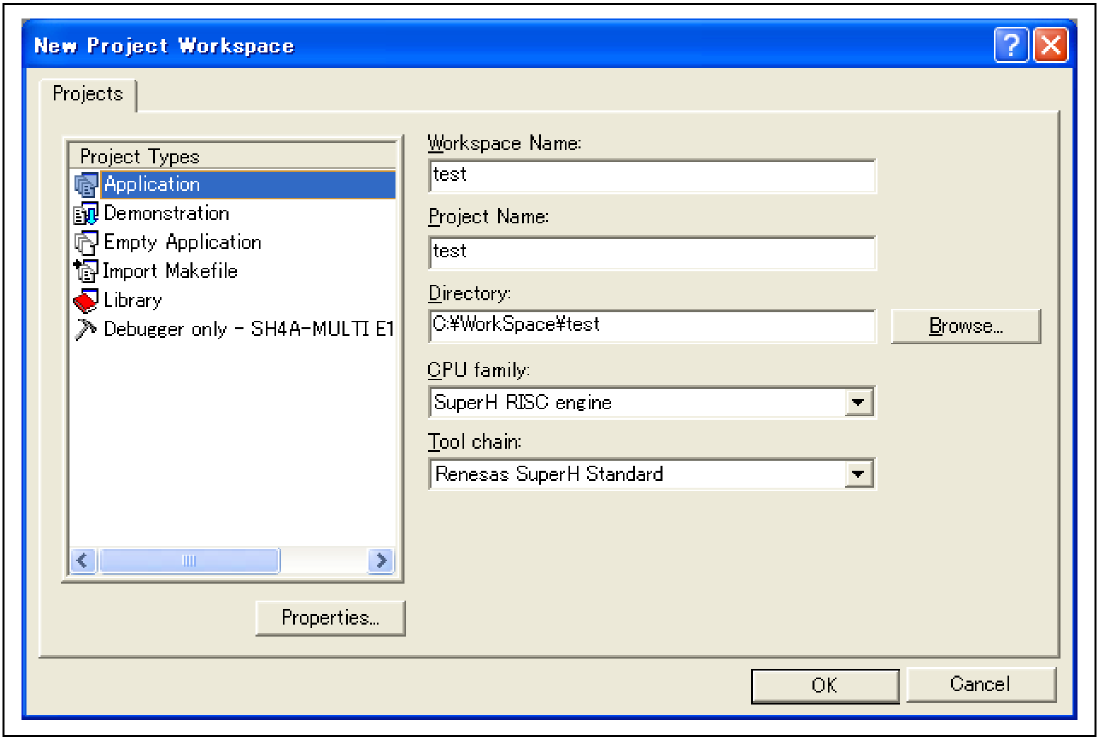


Figure 4.7 [New Project Workspace] Dialog Box

[Workspace Name] edit box:	Enter the new workspace name. Here, for example, enter 'test'.
[Project Name] edit box:	Enter the project name. When the project name is the same as the workspace name, it needs not be entered.
[CPU family] drop-down list box:	Select the target CPU family.
[Tool chain] drop-down list box:	Select the target toolchain name when using the toolchain. Otherwise, select [None].
[Project type] list box:	Select the project type to be used.

Note: When [Demonstration] is selected in the emulator, note the following:
The [Demonstration] is a program for the simulator. When the generated program is used by the emulator, delete the Printf statement.

3. Make the required setting for the toolchain. When the setting has been completed, the following dialog box is displayed.

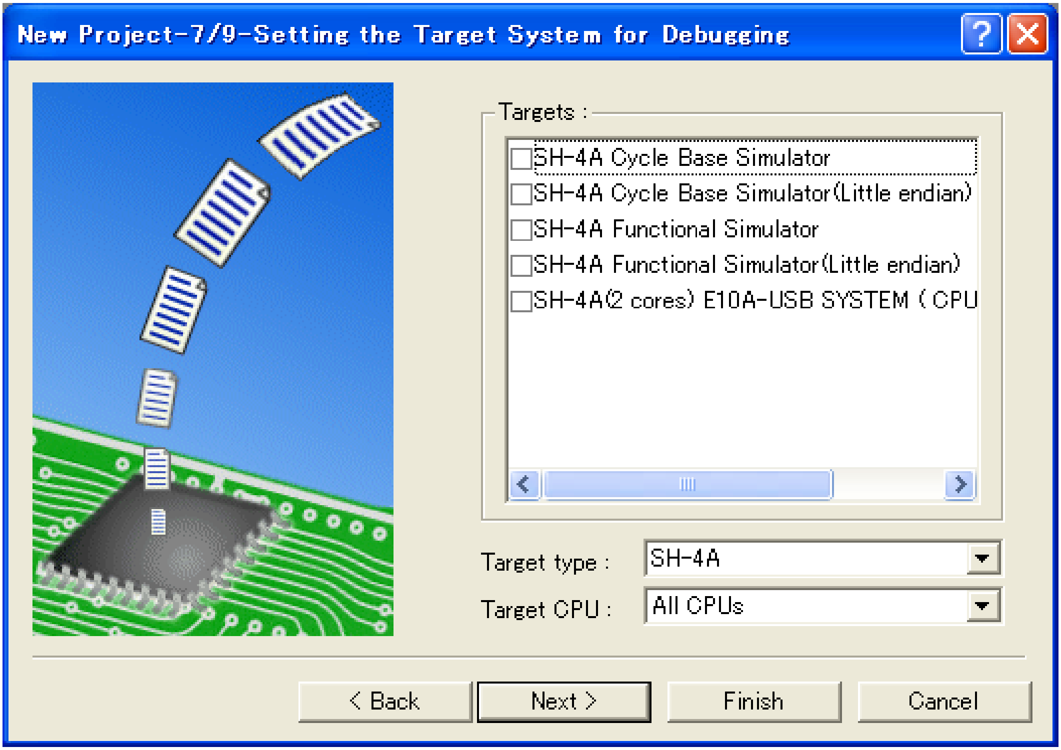


Figure 4.8 [New Project –7/9– Setting the Target System for Debugging] Dialog Box

Check the target emulator and click the [Next] button. Mark other products as required.

4. Set the configuration file name. The configuration file saves the state of High-performance Embedded Workshop except for the emulator.
   Selects the CPU core numbers for debugging from the [Core] drop-down list box.

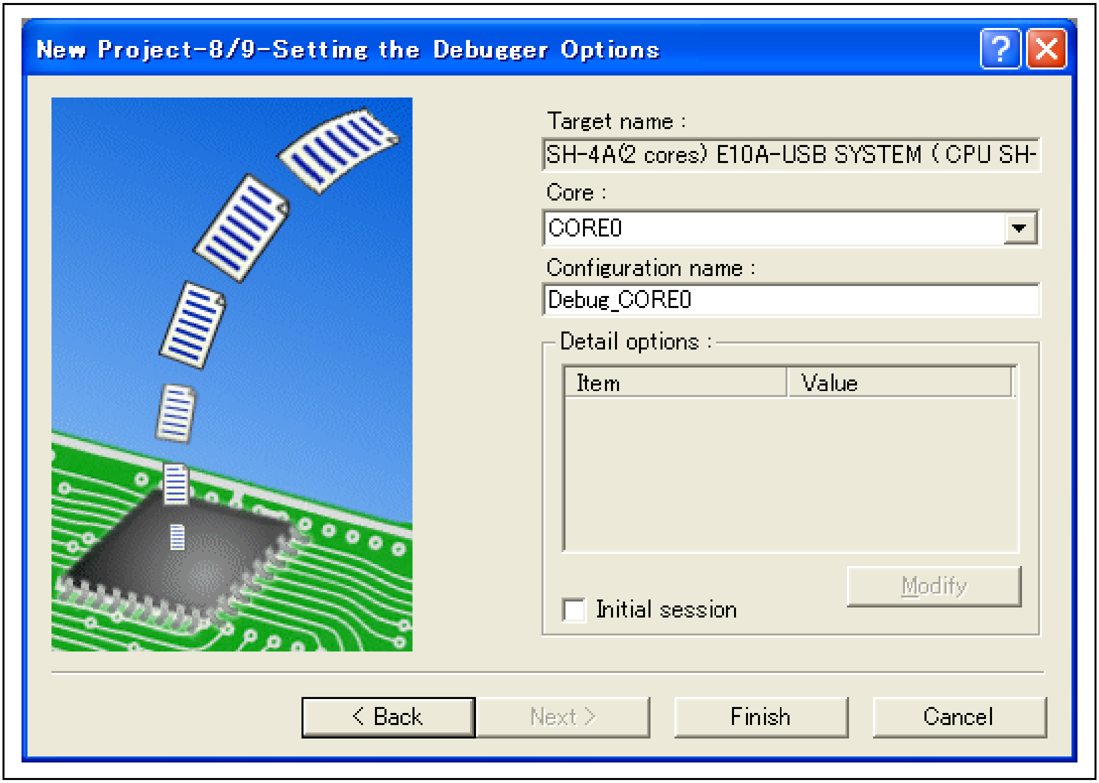


Figure 4.9 [New Project –8/9– Setting the Debugger Options] Dialog Box

This is the end of the emulator setting.

Exit the Project Generator according to the instructions on the screen. The High-performance Embedded Workshop is activated.

5. After the High-performance Embedded Workshop has been activated, connect the emulator. However, it is not needed to connect the emulator immediately after the High-performance Embedded Workshop has been activated.
   To connect the emulator, use one of the methods (a) and (b) below. For operation during connection, refer to section 3.9, System Check.

(a) Connecting the emulator after the setting at emulator activation

Select [Debug settings] from the [Debug] menu to open the [Debug Settings] dialog box. It is possible to register the download module or the command chain that is automatically executed at activation. For details on the [Debug Settings] dialog box, refer to section 4.3, Setting at Emulator Activation.

After the [Debug Settings] dialog box has been set, when the dialog box is closed, the emulator is connected.

(b) Connecting the emulator without the setting at emulator activation

The emulator can be easily connected by switching the session file that the setting for the emulator use has been registered.

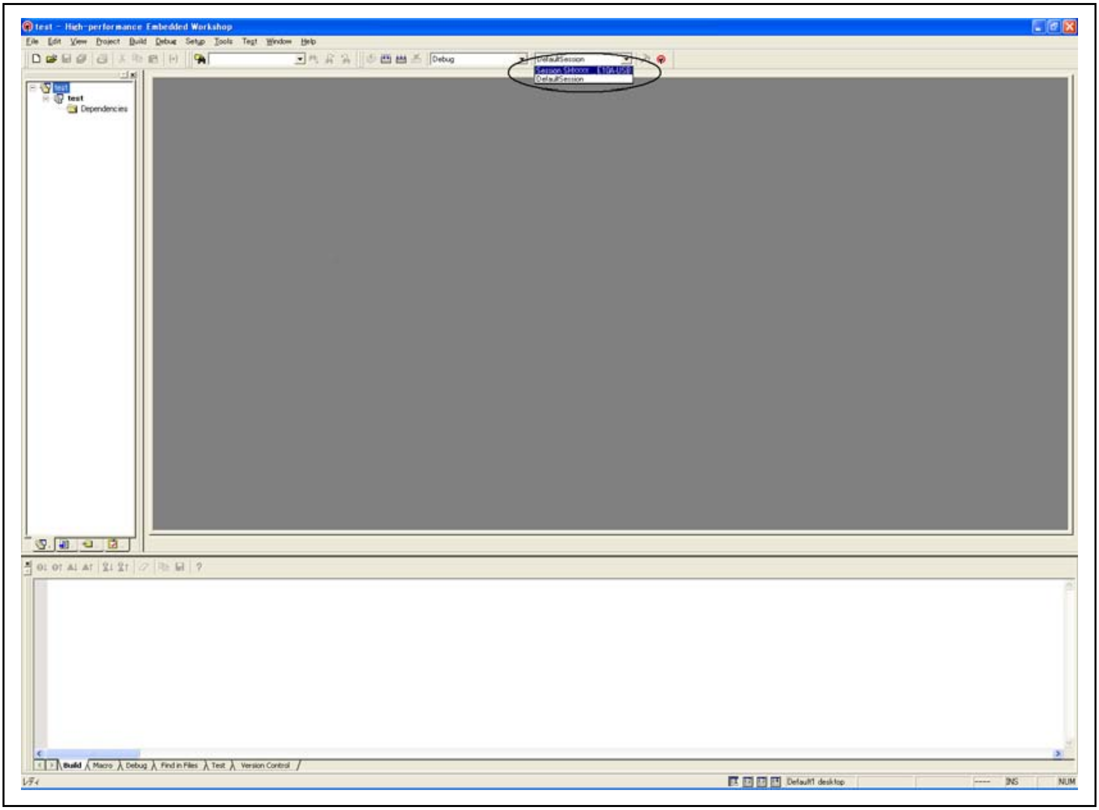


Figure 4.10 Selecting the Session File

In the list box that is circled in figure 4.10, select the session file name including the character string that has been set in the [Target name] text box in figure 4.9, [New Project –8/9– Setting the Debugger Options] dialog box. The setting for using the emulator has been registered in this session file.

After selected, the emulator is automatically connected.

4.1.3 Selecting an Existing Workspace

1. In the [Welcome!] dialog box that is displayed when the High-performance Embedded Workshop is activated, select [Browse to another project workspace] radio button and click the [OK] button.

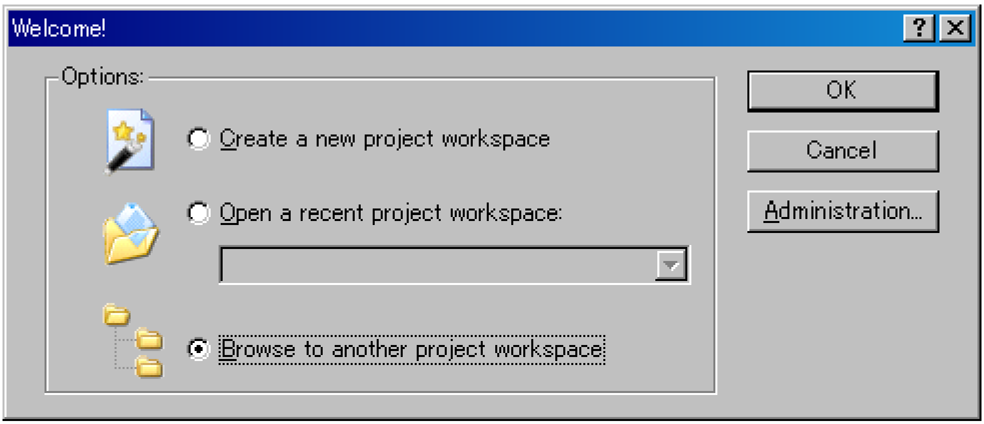


Figure 4.11 [Welcome!] Dialog Box

2. The [Open Workspace] dialog box is displayed. Select a directory in which you have created a workspace.
   After that, select the workspace file (.hws) and press the [Open] button.

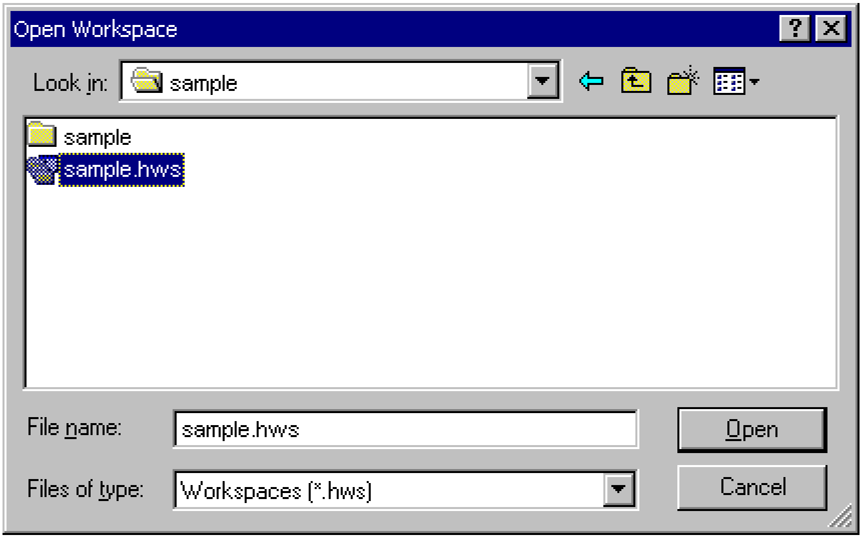


Figure 4.12 [Open Workspace] Dialog Box

3. This activates the High-performance Embedded Workshop and recovers the state of the selected workspace at the time it was saved.
   When the saved state information of the selected workspace includes connection to the emulator, the emulator will automatically be connected. To connect the emulator when the saved state information does not include connection to the emulator, refer to section 4.5, Connecting the Emulator.

4.2 Creating a Project for Synchronized Debugging

For synchronized debugging, the workplace must include as many projects as the number of cores to be debugged.

4.2.1 Adding a New Project

1. Select the [Insert Project] dialog box from the [Project] menu of a workspace opened or created by following the procedure in Section 4.1.1, Creating a New Workspace (Toolchain Not in Use), Section 4.1.2, Creating a New Workspace (Toolchain in Use), or Section 4.1.3, Selecting an Existing Workspace.

The [Insert Project] dialog box is opened.

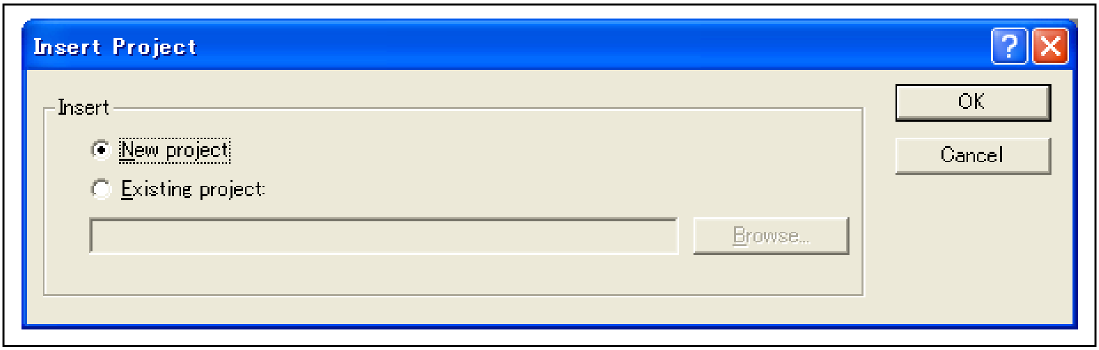


Figure 4.13 [Insert Project] Dialog Box

2. Select the [New project] radio button and click on [OK].
   This opens the [Insert New Project] dialog box.

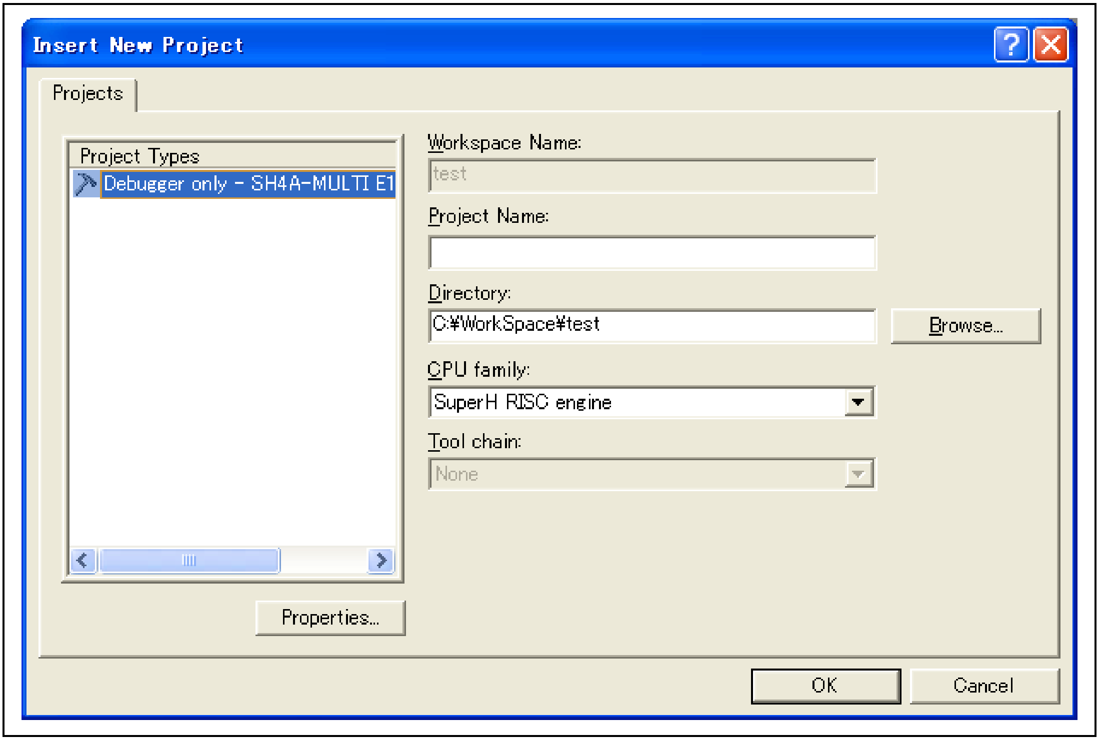


Figure 4.14 [Insert New Project] Dialog Box

3. Create a new project by following the procedure in Section 4.1.1 Creating a New Workspace (Toolchain Not in Use) or Section 4.1.2 Creating a New Workspace (Toolchain in Use).

4.2.2 Adding an Existing Project

1. Select the [Insert Project] item from the [Project] menu of a workspace opened or created by following the procedure in Section 4.1.1, Creating the New Workspace (Toolchain Not in Use), Section 4.1.2, Creating the New Workspace (Toolchain in Use), or Section 4.1.3, Selecting an Existing Workspace. The [Insert Project] dialog box is opened.

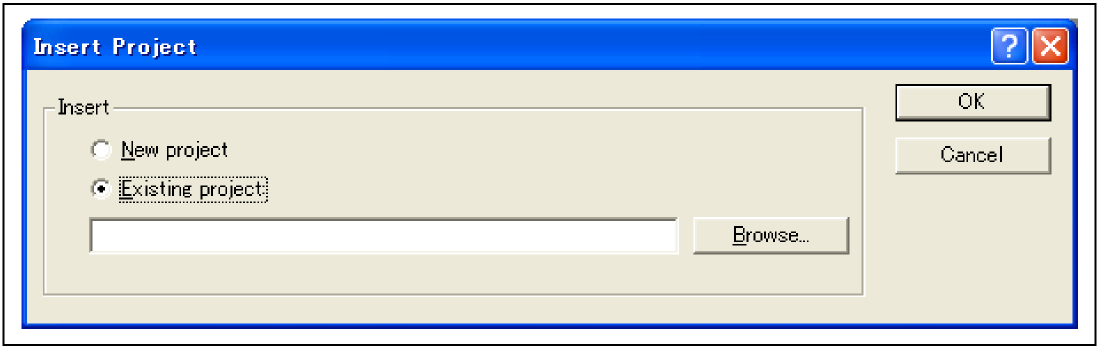


Figure 4.15 [Insert Project] Dialog Box

2. Select the [Existing project] radio button, click on the browse button, find and select the project to be added to the workspace, and click on the [OK] button.

4.3 Setting at Emulator Activation

4.3.1 Setting at Emulator Activation

When the emulator is activated, the command chain can be automatically executed. It is also possible to register multiple load modules to be downloaded. The registered load modules are displayed on the workspace window.

1. Select [Debug settings] from the [Debug] menu to open the [Debug Settings] dialog box.

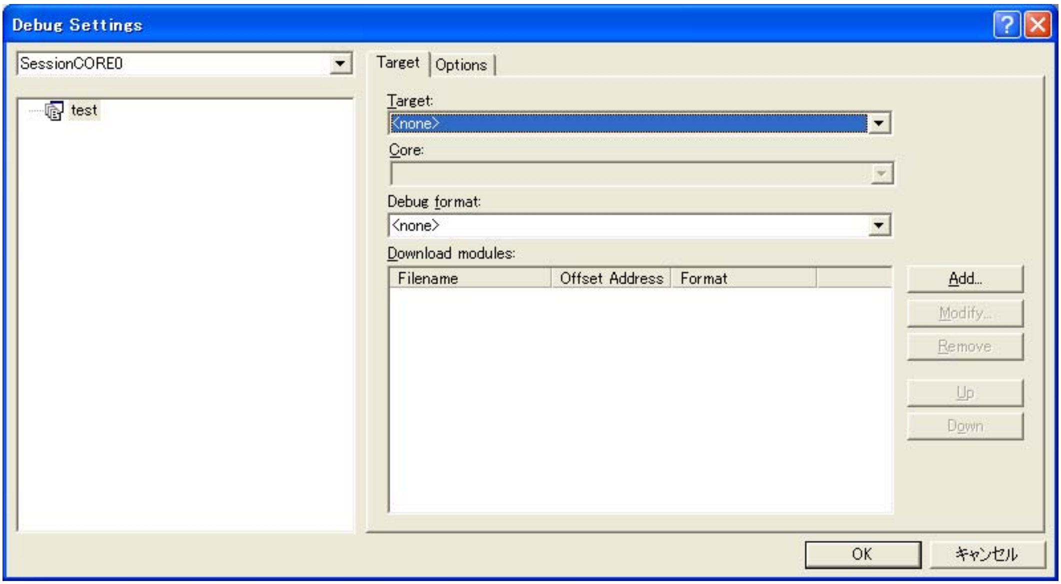


Figure 4.16 [Debug Settings] Dialog Box ([Target] Page)

2. Select the product name to be connected in the [Target] drop-down list box.
   Select the core to be connected from the [Core] drop-down list box.
3. Select the format of the load module to be downloaded in the [Default Debug Format] drop-down list box, then register the corresponding download module in the [Download Modules] list box.

   Note: Here, no program has been downloaded. For downloading, refer to section 5.3, Downloading a Program.
4. Click the [Options] tab.

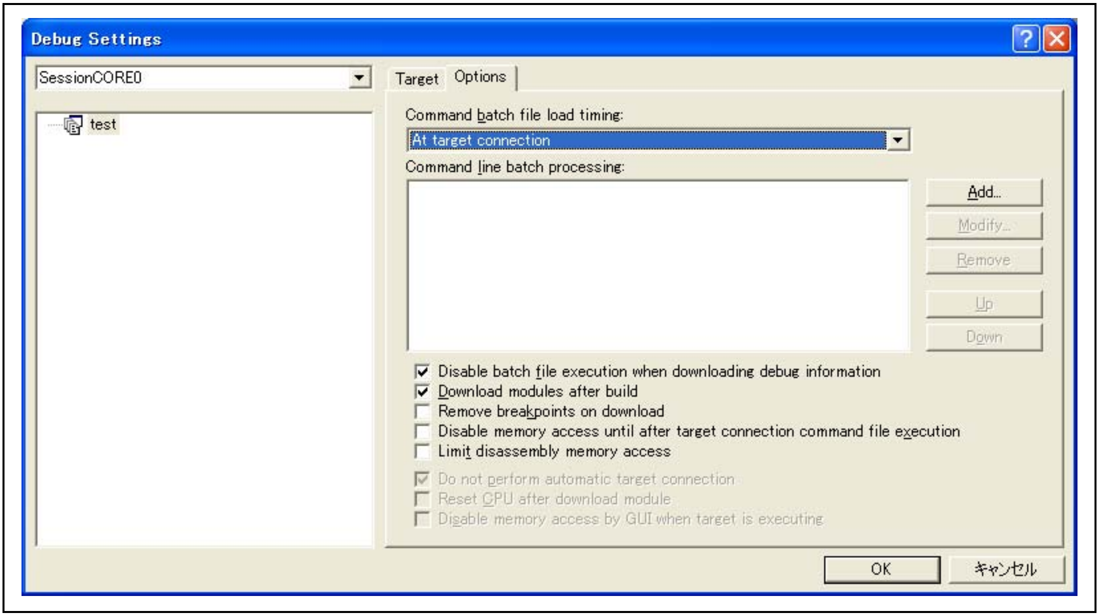


Figure 4.17 [Debug Settings] Dialog Box ([Options] Page)

The command chain that is automatically executed at the specified timing is registered. The following four timings can be specified:

- At connecting the emulator
- Immediately after reset
- Immediately before downloading
- Immediately after downloading

Specify the timing for executing the command chain in the [Command batch file load timing] drop-down list box. In addition, register the command-chain file that is executed at the specified timing in the [Command Line Batch Processing] list box.

4.3.2 Downloading a Program

A download module is added under [Download modules] in the [Workspace] window.

Open the load module of [Download modules] in the [Workspace] window by clicking the right-hand mouse button and select [Download module] to start downloading the module.

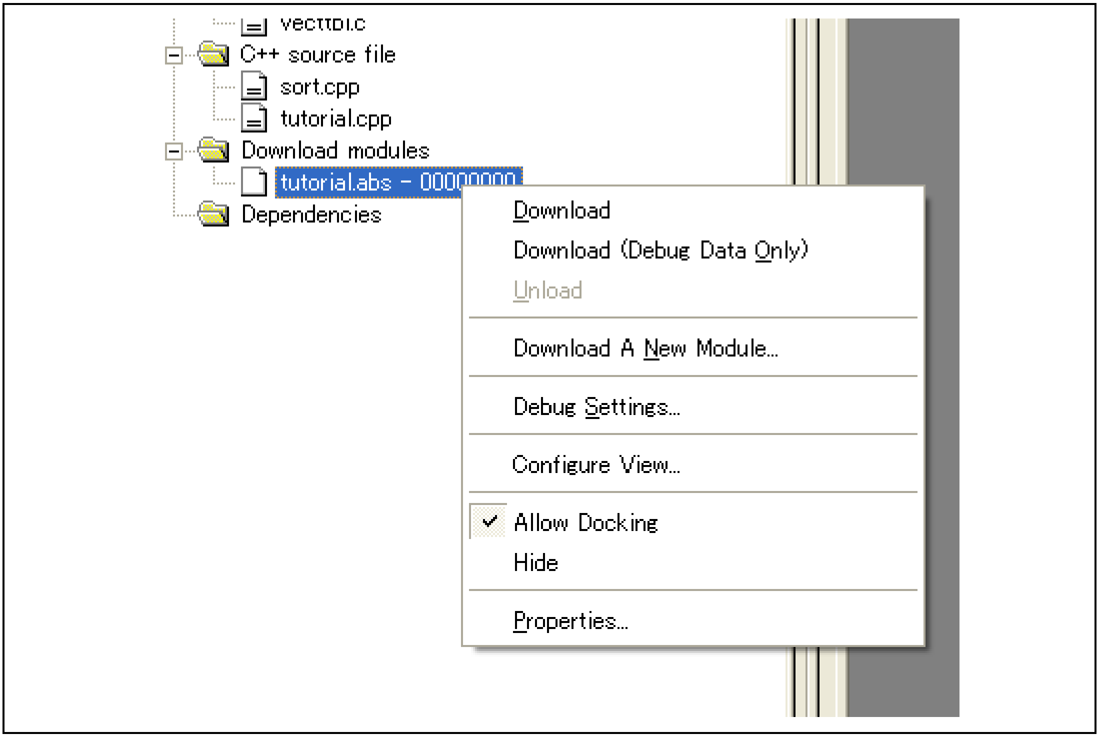


Figure 4.18 Download Menu of the [Workspace] Window ([Projects])

Notes: 1. When load modules are downloaded, select [Debug] -> [Download] -> [All DownLoad Modules].

2. To proceed with source-level synchronized debugging, download the debugging information file for the corresponding CPU. If a module with the same file name has been registered, synchronized downloading is possible. Regarding synchronized downloading, refer to Section 5.2, Setting the Environment for Emulation.

4.4 Debug Sessions

The High-performance Embedded Workshop stores all of your builder options into a configuration. In a similar way, the High-performance Embedded Workshop stores your debugger options in a session. The debugging platforms, the programs to be downloaded, and each debugging platform's options can be stored in a session.

Sessions are not directly related to a configuration. This means that multiple sessions can share the same download module and avoid unnecessary program rebuilds.

Each session's data should be stored in a separate file in the High-performance Embedded Workshop project. Debug sessions are described in detail below.

4.4.1 Selecting a Session

The current session can be selected in the following two ways:

- From the toolbar
  Select a session from the drop-down list box (figure 4.19) in the toolbar.

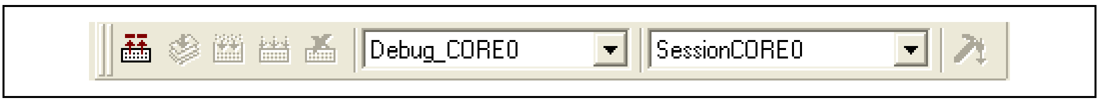


Figure 4.19 Toolbar Selection

- From the dialog box
  1. Select [Debug -> Debug Sessions…]. This will open the [Debug Sessions] dialog box (figure 4.17).

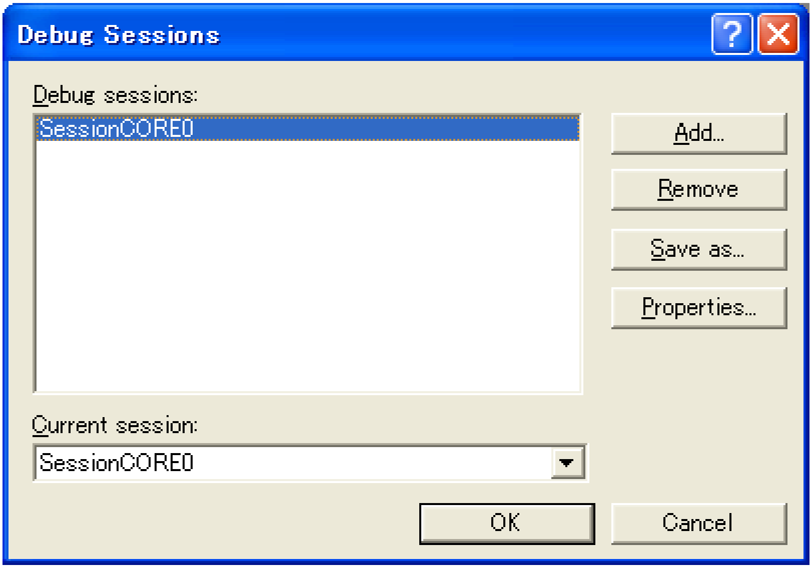


Figure 4.20 [Debug Sessions] Dialog Box

  2. Select the session you want to use from the [Current session] drop-down list box.
  3. Click the [OK] button to set the session.

4.4.2 Adding and Removing Sessions

A new session can be added by copying settings from another session or removing a session.

- To add a new empty session
  1. Select [Debug -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 4.17).
  2. Click the [Add…] button to display the [Add new session] dialog box (figure 4.21).
  3. Check the [Add new session] radio button.
  4. Enter a name for the session.
  5. Click the [OK] button to close the [Debug Sessions] dialog box.
  6. This creates a file with the name entered in step 4. If a file with this name already exists, an error is displayed.

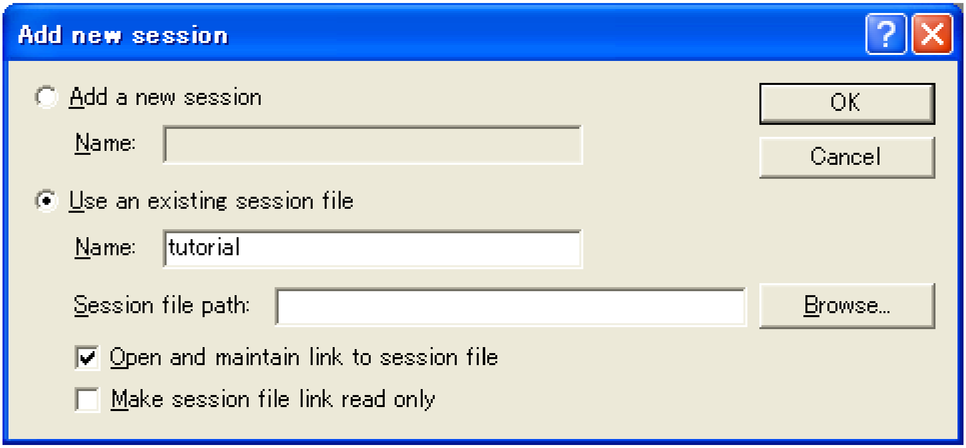


Figure 4.21 [Add new session] Dialog Box

- To import an existing session into a new session file
  1. Select [Debug -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 4.17).
  2. Click the [Add…] button to display the [Add new session] dialog box (figure 4.18).
  3. Check the [Use an existing session file] radio button.
  4. Enter a name for the session.
  5. Enter the name of an existing session file that you would like to import into the existing project or click the [Browse] button to select the file location.

     If the [Open and maintain link to session file] check box is not checked, the imported new session file is generated in the project directory.

     If the [Open and maintain link to session file] check box is checked, a new session file is not generated in the project directory but is linked to the existing session file.

     If the [Make session file link read only] check box is checked, the linked session file is used as read-only.
  6. Click the [OK] button to close the [Debug Sessions] dialog box.

- To remove a session
  1. Select [Debug -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 4.17).
  2. Select the session you would like to remove.
  3. Click the [Remove] button.
     Note that the current session cannot be removed.
  4. Click the [OK] button to close the [Debug Sessions] dialog box.

- To view the session properties
  1. Select [Debug -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 4.17).
  2. Select the session you would like to view the properties for.
  3. Click the [Properties] button to display the [Session Properties] dialog box (figure 4.19).

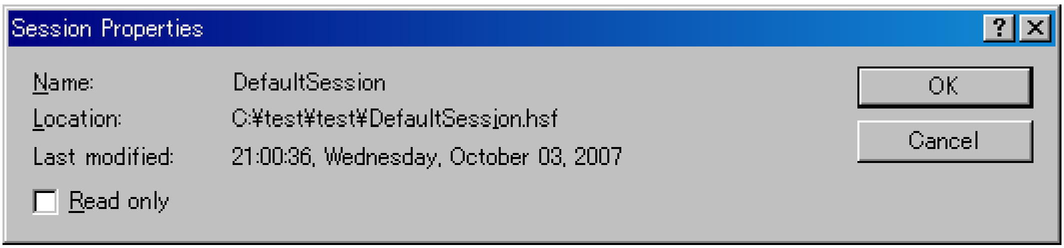


Figure 4.22 [Session Properties] Dialog Box

- To make a session read-only
  1. Select [Debug -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 4.17).
  2. Select the session you would like to make read-only.
  3. Click the [Properties] button to display the [Session Properties] dialog box (figure 4.19).
  4. Check the [Read only] check box to make the link read-only. This is useful if you are sharing debugger-setting files and you do not want data to be modified accidentally.
  5. Click the [OK] button.

- To save a session with a different name
  1. Select [Debug -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 4.17).
  2. Select the session you would like to save.
  3. Click the [Save as…] button to display the [Save Session] dialog box (figure 4.20).

4. Specify the location to save the new file.
5. If you want to export the session file to another location, leave the [Maintain link] check box unchecked. If you would like the High-performance Embedded Workshop to use this location instead of the current session location, check the [Maintain link] check box.
6. Click the [Save] button.

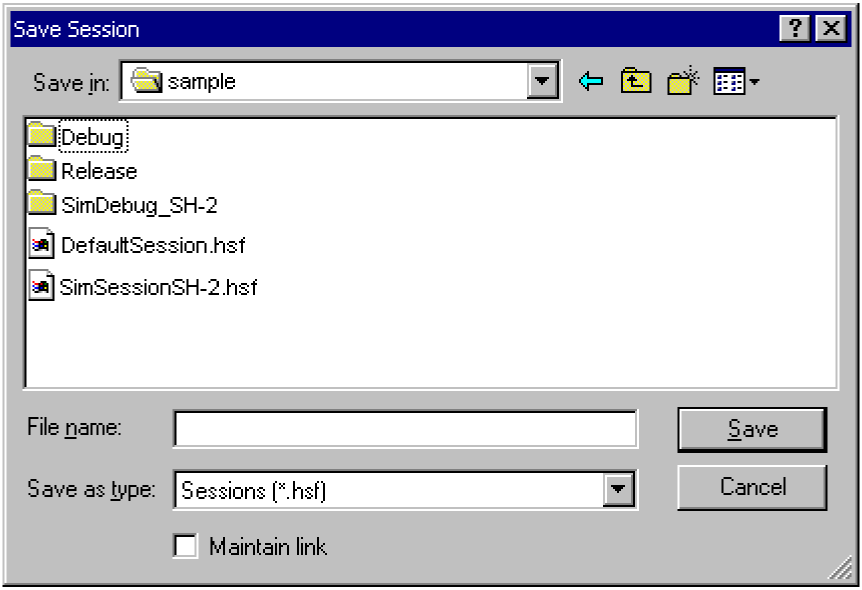


Figure 4.23 [Save Session] Dialog Box

4.4.3 Saving Session Information

- To save a session

  Select [File -> Save Session].

4.5 Connecting the Emulator

Select either of the following two ways to connect the emulator:

(a) Connecting the emulator after the setting at emulator activation

Select [Debug settings] from the [Debug] menu to open the [Debug Settings] dialog box. It is possible to register the download module or the command chain that is automatically executed at activation. For details on the [Debug Settings] dialog box, refer to section 4.3, Setting at Emulator Activation.

After the [Debug Settings] dialog box has been set, when the dialog box is closed, the emulator is connected.

(b) Connecting the emulator without the setting at emulator activation

The emulator can be easily connected by switching the session file that the setting for the emulator use has been registered.

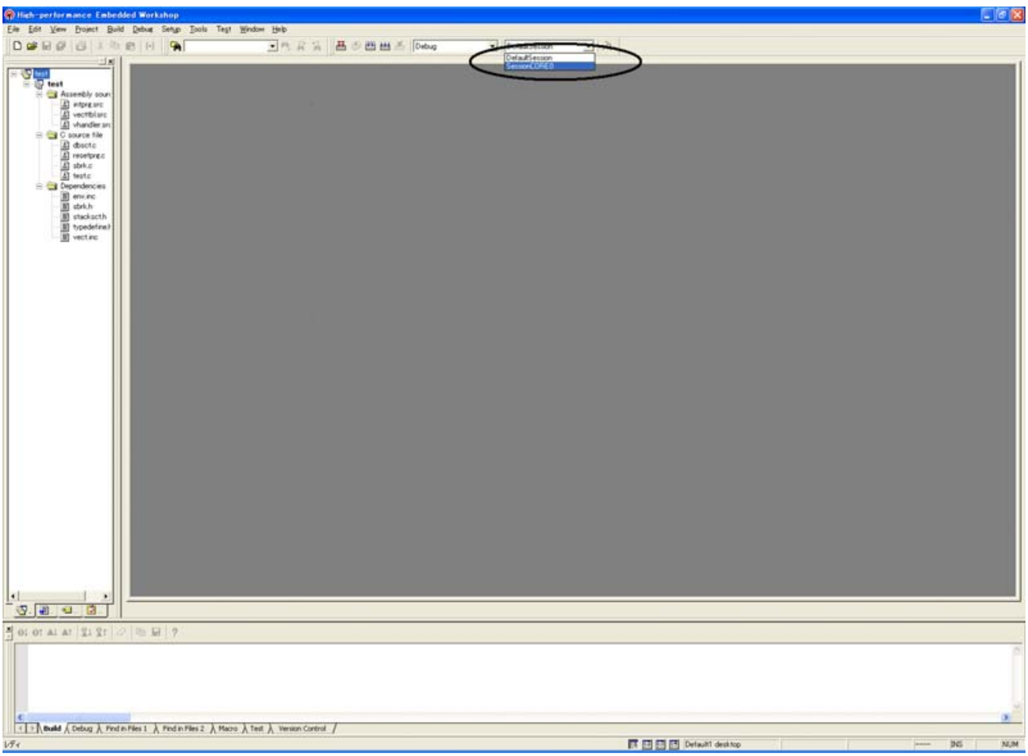


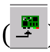
Figure 4.24 Selecting the Session File

In the list box that is circled in figure 4.21, select the session file name including the character string that has been set in the [Target name] text box in figure 4.9, [New Project –8/9– Setting the Debugger Options] dialog box. The setting for using the emulator has been registered in this session file.

After the session file name is selected, the emulator will automatically be connected. For details on the session file, refer to section 4.4, Debug Sessions.

4.6 Reconnecting the Emulator

When the emulator is disconnected, use the following way for reconnection:

Select [Debug -> Connect] or click the [Connect] toolbar button (). The emulator is connected.

Note: 1. The emulator must be selected in the [Target] drop-down list box of the [Debug Settings] dialog box (see figure 4.16, [Debug Settings] Dialog Box ([Target] Page)) that is opened by selecting [Debug settings] from the [Debug] menu.

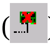
2. If reconnection is to proceed during synchronized debugging, disconnect all of the cores before reconnecting the emulator.

4.7 Ending a Session with the Emulator

When using the toolchain, the emulator can be exited by using the following two methods:

- Canceling the connection of the emulator being activated
- Exiting the High-performance Embedded Workshop

(1) Canceling the connection of the emulator being activated

Select [Disconnect] from the [Debug] menu or click the [Disconnect] toolbar button ().

Note: Do not select the [Disconnect] toolbar button during user program execution or while a dialog box is being displayed for a CPU.

(2) Exiting the High-performance Embedded Workshop

Select [Exit] from the [File] menu.

A message box is displayed. If necessary, click the [Yes] button to save a session. After saving a session, the High-performance Embedded Workshop exits. If not necessary, click the [No] button to exit the High-performance Embedded Workshop.

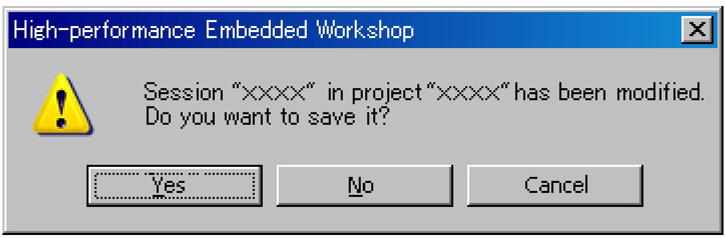


Figure 4.25 [Session has been modified] Message Box

Section 5 Debugging

5.1 Setting up Synchronized Debugging

This section describes how to make the settings for synchronized debugging. For details of functions and operations in synchronized debugging, refer to section 18, Synchronized Debugging Functions in the user's manual for V. 4.09 of the High-performance Embedded Workshop.

5.1.1 Opening the [Synchronized debug] Dialog Box

Open the [Synchronized debugging] dialog box by selecting the [Synchronized debug] item from the [Debugging] menu. Also, when there is a record of synchronized debugging, a synchronized debugging session can be opened from the [Welcome!] dialog box of the High-performance Embedded Workshop.

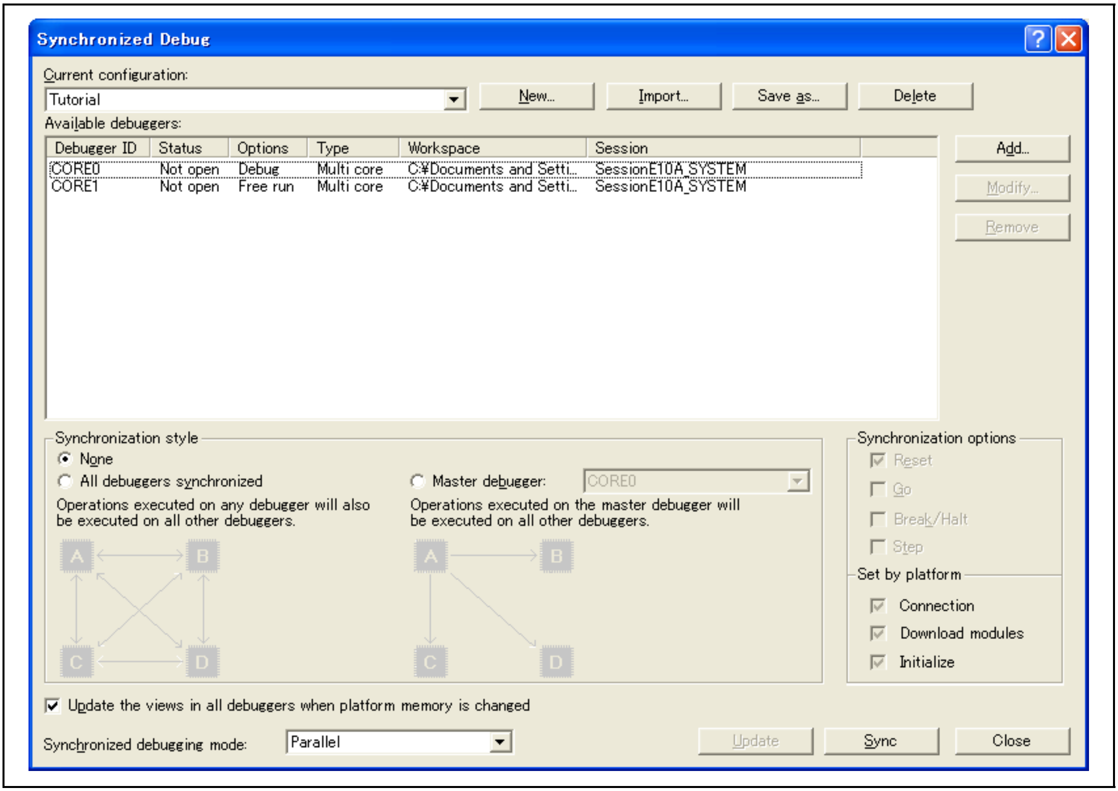


Figure 5.1 [Synchronized debugging] Dialog Box

5.1.2 [Synchronization session] List Box

In the [Synchronization session] list box, set the condition for the synchronization session.

Lists that can be set are shown below.

[Debugger ID]	This setting is for a unique ID that can specify the debugging session. Ensure that the setting is capable of doing so.*1
[Status]	Display the state of the session. Not open: The session is not open in the High-performance Embedded Workshop. The session is opened when the [Synchronize] button is clicked. Not connected: The session is not connected to the emulator. Break: The session is connected to the emulator. The user program is in break state. Running: The session is connected to the emulator. The user program is being executed.
[Options]	When a synchronized debugging session is started, this setting specifies how the session is to be handled. Debug: The session is to be used for debugging. Freerun: The session is not to be used for debugging. The operation of the CPU core is the same as if the emulator were not connected.
[Type]	Display the type of target platform for the session.*2 Single core: Platform with a single core. Multi-core: Platform with multiple cores.
[Workspace]	Display the name of the absolute pass file for workspace file including the session.
[Session]	Display the name of session in the workspace.

Note: 1. Do not mix up session names for different devices.
2. Do not mix up sessions for single-core and multi-core platforms.

5.1.3 [Synchronization style] Group Box

In the [Synchronization style] group box, set the direction of synchronization operations.

Lists that can be set are shown below.

[None]	Synchronization is not performed. All sessions operate individually.
[Synchronization at all debugger]	For checked functions in the [Synchronization options] group box, perform synchronization with all sessions in both directions.
[Master debugger]	For checked functions in the [Synchronization options] group box, take the specified session as the master and perform unidirectional synchronization with all other sessions. Select the [Debugger ID] of the session for use as the master from the drop-down list box.

5.1.4 [Synchronization options] Group Box

In the [Synchronization style] group box, select the operations for synchronization.

The available items are as listed below.

[Reset]	Operations in response to the [Reset CPU] and [Reset Go] functions are synchronized. For synchronization of the response to the [Reset Go] function, the [Go] check box must also should be checked.
[Go]	Operations in response to the [Go] and [Reset Go] functions are synchronized. For synchronization of the response to the [Reset Go] function, the [Reset] check box must also should be checked.
[Break/Halt]	Operations in response to a device break and selection of the [Halt Program] function are synchronized. Selection of synchronized or non-synchronized operation in response to individual types of break is not possible.
[Step]	Operations in response to the various functions for step execution are synchronized. When synchronized stepping is being performed while the other core is executing the user program, operation of the other core at the end of step execution depends on the setting for [Break/Halt].
[Connect]	Operation of the emulator in response to [Connect] is synchronized in all sessions.
[Download Modules]	Operation of the emulator in response to [Download Modules] is synchronized in all sessions.
[Initialize]	Operation of the emulator in response to [Initialize] is synchronized in all sessions.

Note: The functions that can be set in this dialog box vary according to the emulator in use. For details, refer to section 2.2.1, Synchronized Debugging Functions in the additional document, ″Supplementary Information on Using the SHxxxx″.

5.1.5 [Memory update] Options

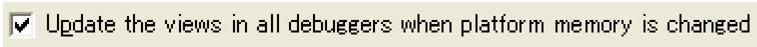


Figure 5.2 [Memory update] option

When this option is checked all High-performance Embedded Workshop views which display memory data (e.g. the memory view, the watch view etc.) in all Debuggers will update whenever the memory is changed in any synchronized Debugger. *
If the option is not set then only the memory views in local Debugger will update when memory is changed. If memory is shared between Debuggers, a manual refresh will need to be performed in the memory related windows of the other Debuggers in order for them to display the correct memory.

Note: If the user program is being executed, reading of memory will lead to a short break.

5.1.6 [Synchronization debugging mode] Drop-Down List Box

In the [Synchronization debugging mode] drop-down list box, set the status for the High-performance Embedded Workshop to be used on the synchronization debugging.

Internal	The specified debuggers will be opened in one instance of the High-performance Embedded Workshop. This mode is only available if all of the debugger sessions are in the same High-performance Embedded Workshop workspace.
Parallel	The debugger sessions will be opened in separate instances of the High-performance Embedded Workshop.

5.2 Setting the Environment for Emulation

This section describes the debugging operations and their related windows and dialog boxes.

5.2.1 Opening the [Configuration] Dialog Box

Selecting [Setup -> Emulator -> System…] or clicking the [Emulator System] toolbar button () opens the [Configuration] dialog box.

5.2.2 [General] Page

On the [General] page, make basic emulator settings for a specific CPU.
These settings can be made per CPU.

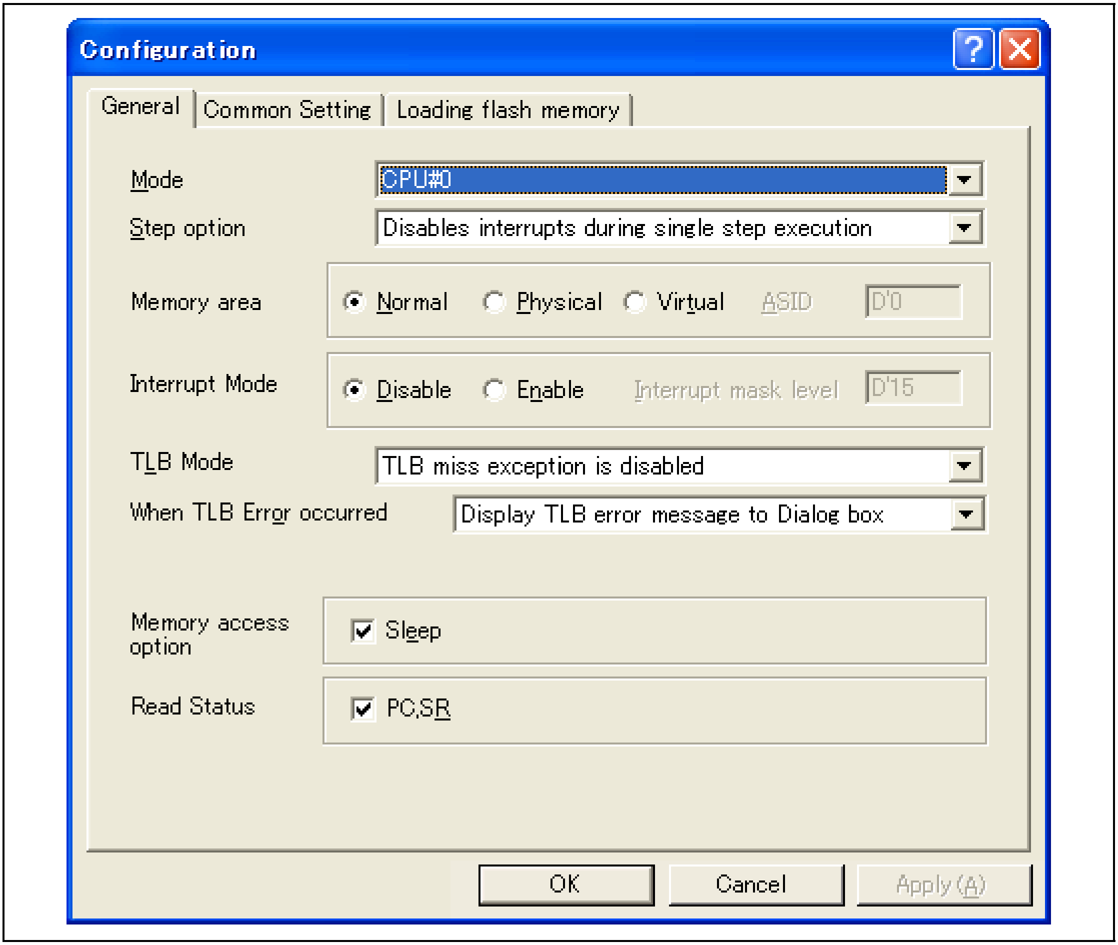


Figure 5.3 [Configuration] Dialog Box ([General] Page)

Items that can be displayed in the sheet are listed below.

[Mode]	Displays the CPU number.
[Step option]	Sets the step interrupt option. Disable interrupts during single step execution: Disables interrupts* during step execution. Enable interrupts during single step execution: Enables interrupts during step execution.
[Reset assert (Auto Connect)]	When the emulator is connected to the host computer and the [CPU reset] function or [Reset after execution] function is used, the reset signal is generated by the emulator.

Note: Includes the interrupts during a break.

CAUTION

Do not to use the [Reset assert (Auto Connect)] option unless the port connector is properly connected as shown in Section 1.5, Recommended Circuit between the H-UDI Port Connector and the MPU of the additional document, ″Supplementary Information on Using the SHxxxx″. Using the [Auto connect] option without the proper connection will damage the user system.

Note: The items that can be set in this dialog box vary according to the emulator in use. For details, refer to the online help.

5.2.3 [Common Setting] Page

On the [Common Setting] page, make basic emulator settings for both CPUs.
These settings are common to the CPUs.

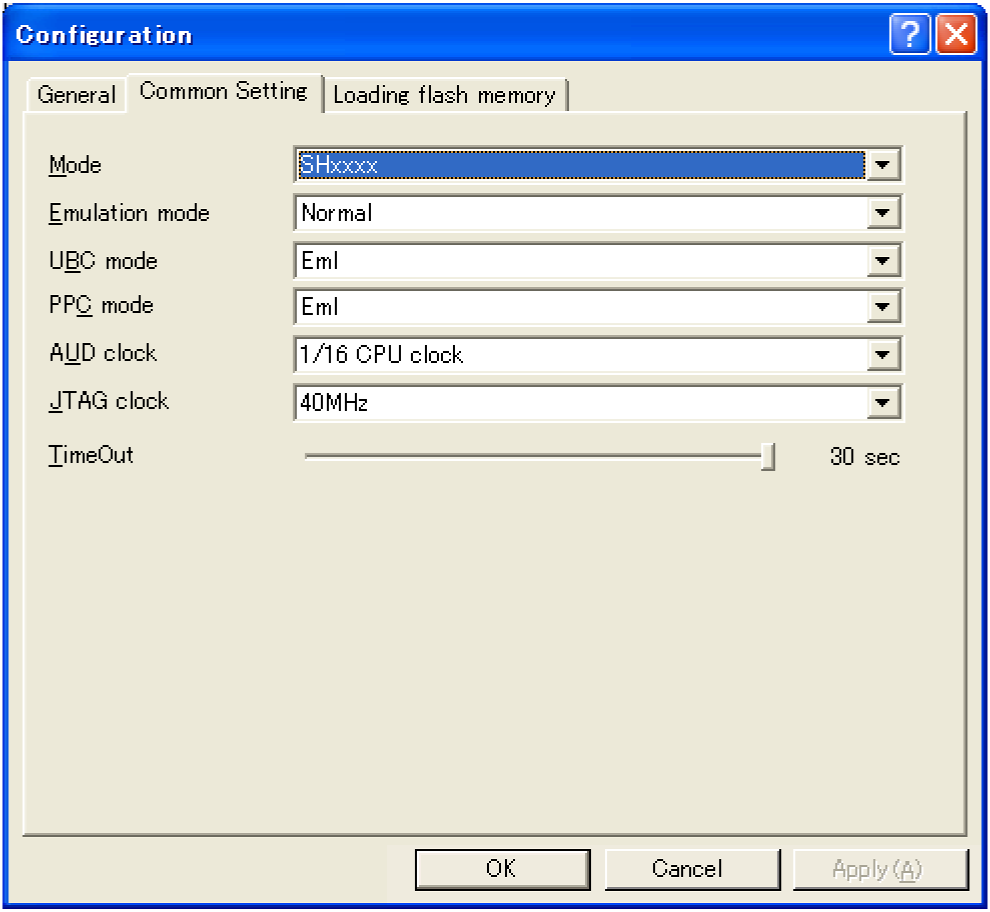


Figure 5.4 [Configuration] Dialog Box ([Common Setting] Page)

Items that can be set on this page are listed below.

[Mode]	Indicates the device name.
[Emulation mode]	Selects the mode of emulation for user programs. Normal: Normal execution of emulation for the user program. No break: The user program is executed with settings of PC breakpoints and hardware breakpoints temporarily disabled.
[UBC mode]	Set the UBC mode. Eml: The UBC is used as an event condition by the emulator. User: The UBC is released to the user.
[PPC mode]	Set the PPC mode. Eml: The PPC is used in the [Performance Analysis] functions of the emulator. User: The PPC is released to the user*.
[AUD clock]	This is the clock signal for the acquisition of AUD trace information. Lower frequencies correspond to a greater incidence of data loss when real-time tracing is in use. Set this item so that the frequency does not exceed the upper limit on AUD clock frequency of the supported device. For details on the upper limits of AUD clock frequency in individual devices, refer to section 2.2.4, Notes on Using the JTAG (H-UDI) Clock (TCK) and AUD Clock (AUDCK) in the additional document, ″Supplementary Information on Using the SHxxxx″.
[JTAG clock]	This is the transfer clock for signals other than AUD trace signals. Lower frequencies correspond to downloading taking longer times. Set this item so that the frequency does not exceed the upper limit on TCK clock frequency of the supported device. For details on the upper limits of TCK clock frequency in individual devices, refer to section 2.2.4, Notes on Using the JTAG (H-UDI) Clock (TCK) and AUD Clock (AUDCK) in the additional document, ″Supplementary Information on Using the SHxxxx″.

[Timeout]	Set the period of waiting before a timeout error is considered to have occurred. The setting is in three-second units over the range from three to thirty seconds.

Note: If ″User″ is selected, certain functions are not available. Since this depends on the product in use, consult the online help system for details.

Note: The effective items and items that can be set in this dialog box differ with the emulator in use. For details, refer to the online help system.

5.2.4 Downloading to the Flash Memory

Sets the emulator operation conditions for downloading the external flash memory. This function is not available depending on the MCU.

For details, refer to section 6.22, Download Function to the Flash Memory Area. These settings can be made per CPU.

Note: Do not use the synchronized downloading function to download separate data for the individual CPUs to the external flash memory.

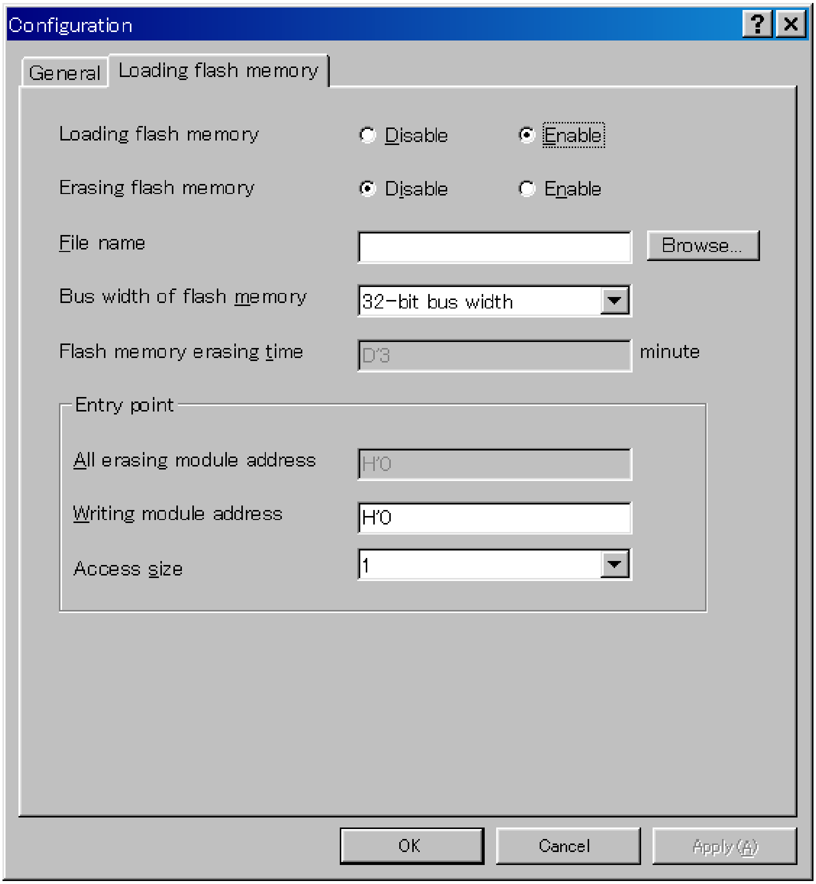


Figure 5.5 [Configuration] Dialog Box ([Loading flash memory] Page)

Items that can be displayed in the sheet are listed below.

[Loading flash memory]	Sets Enable for flash memory downloading. At Enable, when the flash memory is downloaded on the High-performance Embedded Workshop, the write module is always called. Disable: Not download to the flash memory Enable: Download to the flash memory
[Erasing flash memory]	Sets Enable for erasing before the flash memory is programmed. At Enable, the erase module is called before calling the write module. Disable: Not erase the flash memory Enable: Erase the flash memory
[File name]	Sets the write/erase module name. The file that has been set is loaded to the RAM area before loading to the flash memory.
[Bus width of flash memory]	Sets the bus width of the flash memory.
[Flash memory erasing time]	Sets the TIMEOUT value at flash memory erasing. Increase the value if erasing requires much time although the default time is three minutes. The values that can be set are as follows: D'0 (minimum) and D'65535 (maximum). Only positive integers can be input.
[Entry point]	Sets the calling destination address or access size of the write/erase module. (It must be RAM address.) All erasing module address: Inputs the calling destination address of the erase module. Writing module address: Inputs the calling destination address of the write module. Access size: Selects the access size of the RAM area that is used for loading the write/erase module.

5.3 Downloading a Program

This section describes how to download a program and view it as source code or assembly-language mnemonics.

Note: After a break has been detected, the High-performance Embedded Workshop displays the location of the program counter (PC). In most cases, for example if an Elf/Dwarf2-based project is moved from its original path, the source file may not be automatically found. In this case, the High-performance Embedded Workshop will open a source file browser dialog box to allow you to manually locate the file.

5.3.1 Downloading a Program

A load module to be debugged must be downloaded.

To download a program, select the load module from [Debug -> Download] or select [Download] from the popup menu opened by clicking the right-hand mouse button on the load module in [Download modules] of the [Workspace] window.

Notes: 1. Before downloading a program, it must be registered to the High-performance Embedded Workshop as a load module. For registration, refer to section 4.3, Setting at Emulator Activation.

2. When a program is downloaded to the external RAM, the bus controller must be initially set in the area for downloading. Especially, check that the initialization of SDRAM or the setting of the bus width is appropriate for the target system. Download the corresponding debugging information file to execute source-level debugging on the High-performance Embedded Workshop for CPU0 or CPU1.

3. To proceed with source-level debugging for multiple CPUs, use the synchronized downloading function or download debugging information files for the respective CPUs.

5.3.2 Viewing the Source Code

Select your source file and click the [Open] button to make the High-performance Embedded Workshop open the file in the integrated editor. It is also possible to display your source files by double-clicking on them in the [Workspace] window.

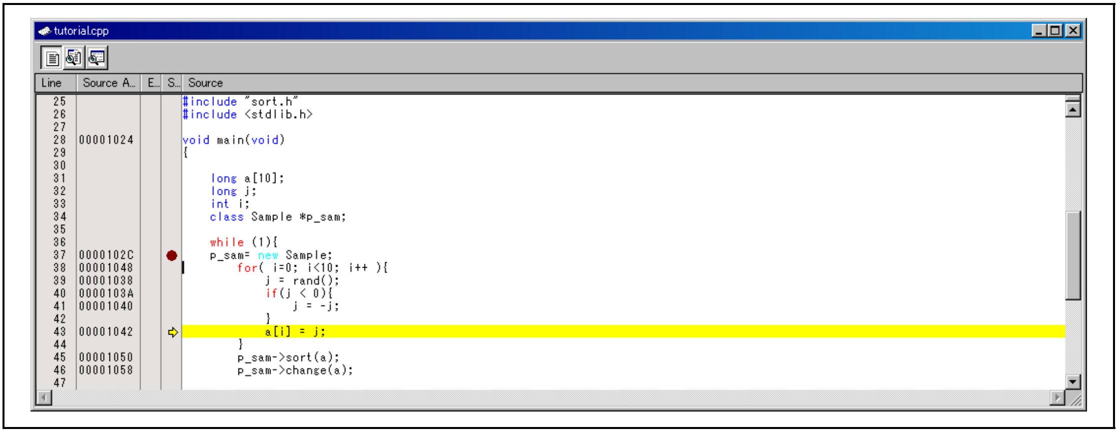


Figure 5.6 [Source] Window

In this window, the following items are shown on the left as line information.

The first column (Source address column): Address information

The second column (Event column): Event information (event condition)

The third column (S/W breakpoint column): PC, bookmark, and breakpoint information

Source address column

When a program is downloaded, an address for the current source file is displayed on the Source address column. These addresses are helpful when setting the PC value or breakpoints.

Event column

The Event column displays the following item:

:An address condition for the event condition is set. The number of address conditions that can be set is the same as that of event condition channels at which the address condition can be set, but it differs depending on the product.

This is also set by using the popup menu.

The bitmap symbol above is shown by double-clicking the Event column. This is also set by using the popup menu.

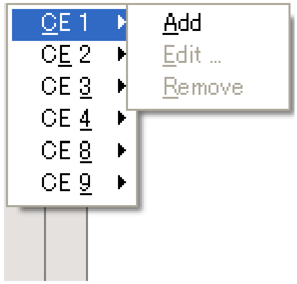


Figure 5.7 Popup Menu

Note: The contents of the Event column are erased when conditions other than the address condition are added to each channel by using the [Edit] menu or in the [Event] window.

S/W breakpoint column

S/W breakpoint column displays the following items:

: A bookmark is set.

: A PC Break is set.

: PC location

➲ To switch off a column in all source files

1. Click the right-hand mouse button on the [Source] window or select the [Edit] menu.
2. Click the [Define Column Format…] menu item.
3. The [Global Editor Column States] dialog box is displayed.
4. A check box indicates whether the column is enabled or not. If it is checked, the column is enabled. If the check box is gray, the column is enabled in some files and disabled in others. Deselect the check box of a column you want to switch off.
5. Click the [OK] button for the new column settings to take effect.

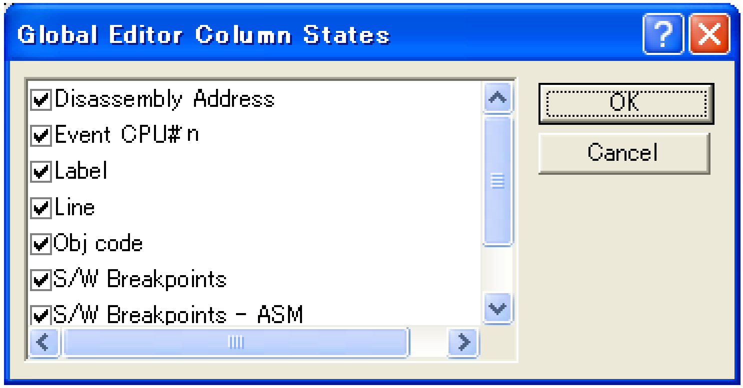


Figure 5.8 [Global Editor Column States] Dialog Box

➲ To switch off a column in one source file

1. Open the source file which contains the column you want to remove and click the [Edit] menu.
2. Click the [Columns] menu item to display a cascaded menu item. The columns are displayed in this popup menu. If a column is enabled, it has a tick mark next to its name. Clicking the entry will toggle whether the column is displayed or not.

5.3.3 Viewing the Assembly-Language Code

Click the [Disassembly] toolbar button at the top of the window when a source file is opened to show the assembly-language code that corresponds to the current source file.

If you do not have a source file, but want to view code in the assembly-language level, either choose [View] -> [Disassembly…] or click the [Disassembly] toolbar button ( ). The [Disassembly] window opens at the current PC location and shows [Address] and [Code] (optional) which show the disassembled mnemonics (with labels when available).

Selecting the [Mixed display] toolbar button ( ) displays both the source and the code. The following shows an example in this case.

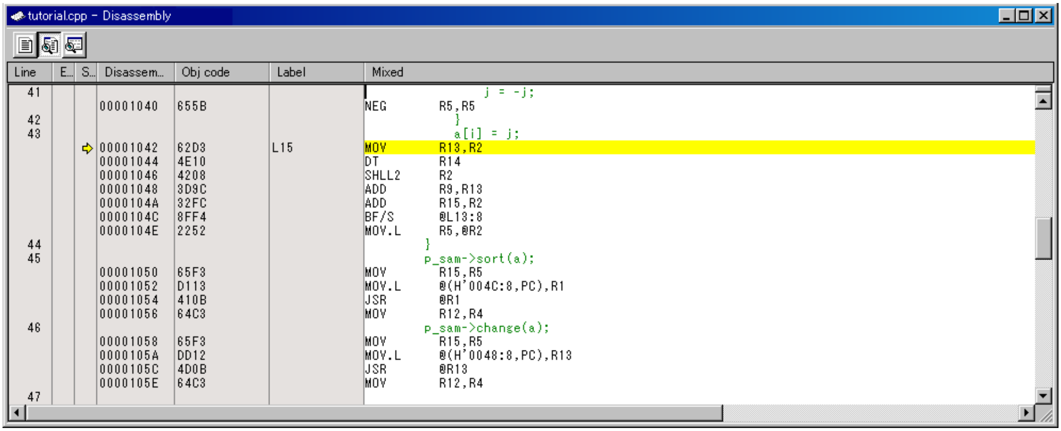


Figure 5.9 [Disassembly] Window

5.3.4 Modifying the Assembly-Language Code

You can modify the assembly-language code by double-clicking on the instruction that you want to change. The [Assembler] dialog box will be opened.

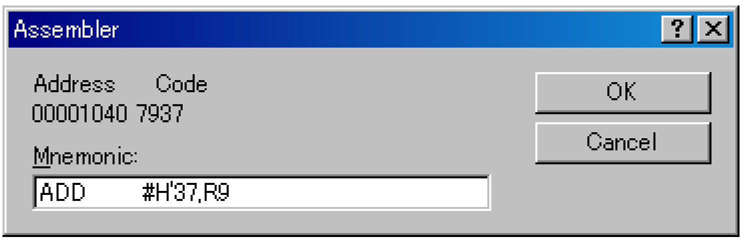


Figure 5.10 [Assembler] Dialog Box

The address, machine code, and disassembled instruction are displayed. Enter the new instruction or edit the current instruction in the [Mnemonic] field. Pressing the [Enter] key will assemble the instruction into memory and move on to the next instruction. Clicking the [OK] button will assemble the instruction into memory and close the dialog box. Clicking the [Cancel] button or pressing the [Esc] key will close the dialog box.

Note: The assembly-language display is disassembled from the machine code on the actual memory. If the memory contents are changed, the dialog box (and the [Disassembly] window) will show the new assembly-language code, but the display content of the [Editor] window will not be changed. This is the same even if the source file contains assembly codes.

5.3.5 Viewing a Specific Address

When you are viewing your program in the [Disassembly] window, you may want to look at another area of your program's code. Rather than scrolling through a lot of code in the program, you can go directly to a specific address. Double-click on the address in the [Disassembly] window or select [Set Address…] from the popup menu, and the dialog box shown in figure 5.8 is displayed.

Figure 5.11 [Set Address] Dialog Box

Enter the address or label name in the edit box and either click on the [OK] button or press the [Enter] key. The [Disassembly] window will be updated to show the code at the new address. When an overloaded function or a class name is entered, the [Select Function] dialog box opens for you to select a function.

5.3.6 Viewing the Current Program Counter Address

Wherever you can enter an address or value into the High-performance Embedded Workshop, you can also enter an expression. If you enter a register name prefixed by the hash character, the contents of that register will be used as the value in the expression. Therefore, if you open the [Set Address] dialog box and enter the expression #pc, the [Editor] or [Disassembly] window will display the current PC address. It also allows the offset of the current PC to be displayed by entering an expression with the PC register plus an offset, e.g., #PC+0x100.

5.4 Displaying Memory Contents in Realtime

Use the [Monitor] window to monitor the memory contents during user program execution. These settings can be made per CPU.

Note: This function is not supported in some devices to be debugged. For details on the specifications of each product, refer to the online help.

5.4.1 Opening the [Monitor] Window

To open the [Monitor] window, select [View -> CPU -> Monitor -> Monitor Setting...] or click the [Monitor] toolbar button () to display the [Monitor Setting] dialog box.

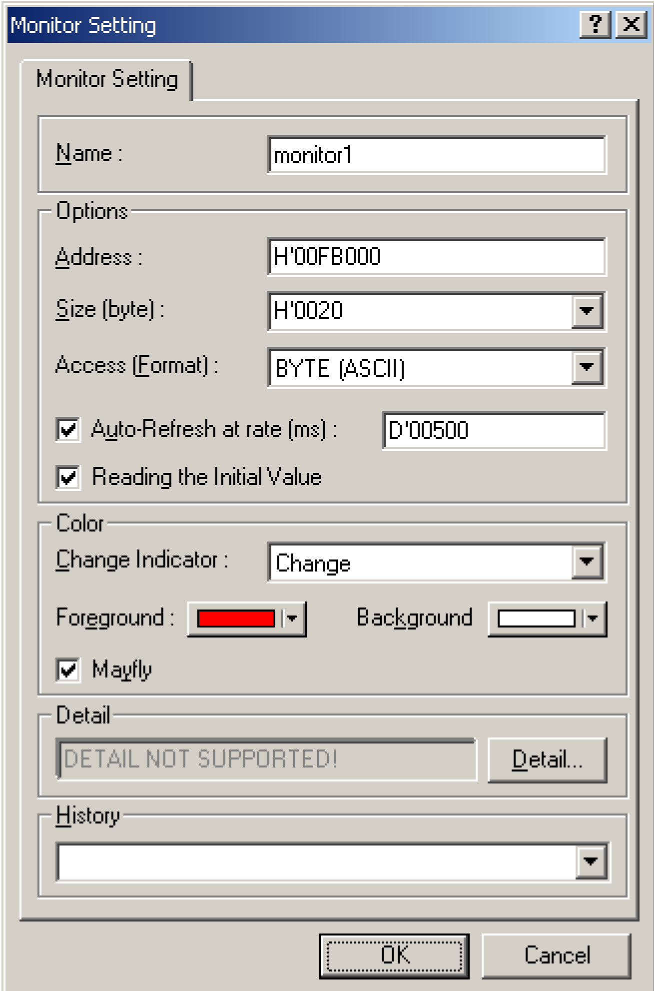


Figure 5.12 [Monitor Setting] Dialog Box

[Name]: Decides the name of the monitor window.

[Options]: Sets monitor conditions.

- [Address]: Sets the start address for monitoring.
- [Size]: Sets the range for monitoring.
- [Access]: Sets the access size to be displayed in the monitor window.

[Auto-Refresh at rate]: Sets the interval for acquisition by monitoring.

[Reading the Initial Value]: Selects reading of the values in the monitored area when the monitor window is opened.

[Color]: Sets the method to update monitoring and the attribute of colors.

[Change Indicator]: Selects how to display the values that have changed during monitoring (available when [Reading the Initial Value] has been selected).

No change: No color change.

Change: Color is changed according to the [Foreground] and [Background] options.

Gray: Those data with values that have not been changed are displayed in gray.

Appear: A value is only displayed after changed.

[Foreground]: Sets the color used for display (available when [Change] has been selected).

[Background]: Sets the background color (available when [Change] has been selected).

[Mayfly]: A check in this box selects restoration of the color of those data which have not been updated in a specified interval to the color selected in the [Background] option. The specified interval is the interval for monitor acquisition (available when [Change], [Gray], or [Appear] has been selected).

[Detail]: Not supported in the emulator.

[History]: Displays the previous settings.

Note: Selection of the foreground or background color may not be available depending on the operating system in use.

After setting, clicking the [OK] button displays the [Monitor] window.

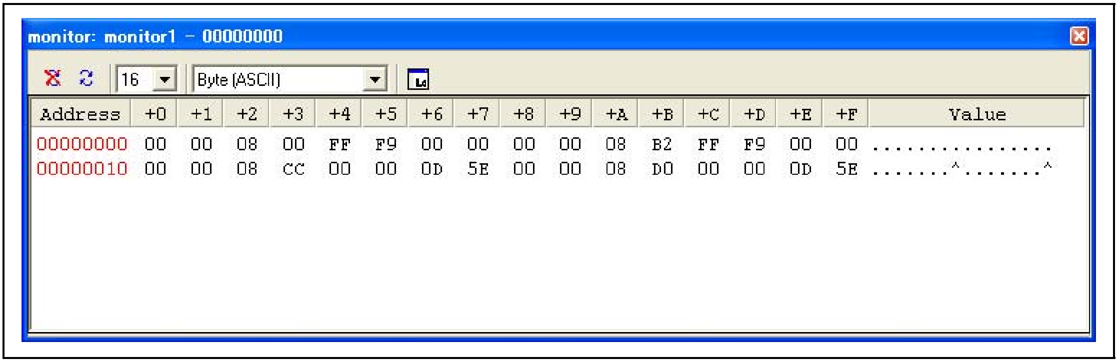


Figure 5.13 [Monitor] Window

During user program execution, the display is updated according to the setting value of the auto-update interval.

Note: Select [Refresh] from the popup menu when data is not displayed correctly after changing the address or content of memory.

5.4.2 Changing the Monitor Settings

Selecting [Monitor Settings...] from the popup menu of the [Monitor] window displays the [Monitor Setting] dialog box, which allows the settings to be changed.

Colors, the size of accesses, and the display format can be easily changed from [Color] or [Access] of the popup menu.

5.4.3 Temporarily Stopping Update of the Monitor

During user program execution, the display of the [Monitor] window is automatically updated according to the auto-update interval. Select [Lock Refresh] from the popup menu of the [Monitor] window to stop the update of display. The characters in the address section are displayed in black, and the update of display is stopped.

Selecting [Lock Refresh] again from the popup menu cancels the stopped state.

5.4.4 Deleting the Monitor Settings

Selecting [Close] from the popup menu of the [Monitor] window to be deleted closes the [Monitor] window and deletes the monitor settings.

5.4.5 Monitoring Variables

Using the [Watch] window refers to the value of any variables.

When the address of the variable registered in the [Watch] window exists within the monitoring range that has been set by the Monitor function, the value of the variable can be updated and displayed.

This function allows checking the content of a variable without affecting the realtime operation.

5.4.6 Hiding the [Monitor] Window

When using the Monitor function to monitor the value of a variable from the [Watch] window, hide the [Monitor] window for the effective use of the screen.

The current monitoring information is listed as the submenu when selecting [Display -> CPU -> Monitor]. The list consists of the [Monitor] window name and the address to start monitoring.

When the left of the list is checked, the [Monitor] window is being displayed.

Selecting items of the [Monitor] window you want to hide from the monitor setting list displays no [Monitor] window and removes the check mark at the left of the list.

To display the [Monitor] window again, select the hidden the [Monitor] window.

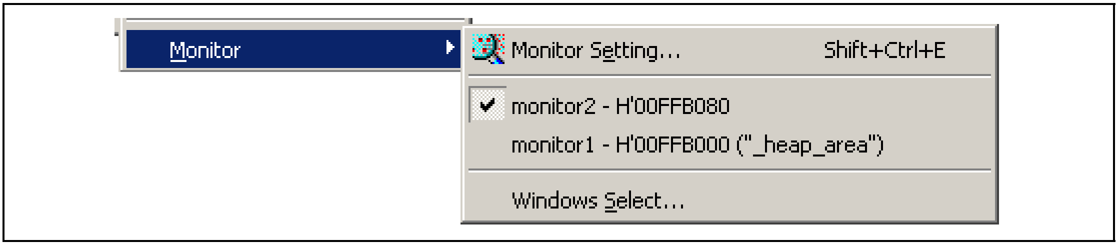


Figure 5.14 Monitor Setting List

5.4.7 Managing the [Monitor] Window

Selecting [Display -> CPU -> Monitor -> Windows Select…] displays the [Windows Select] dialog box. In this window, the current monitoring condition is checked and the new monitoring condition is added, edited, and deleted in succession.

Selecting multiple monitoring conditions enables a temporary stop of update, hiding, and deletion.

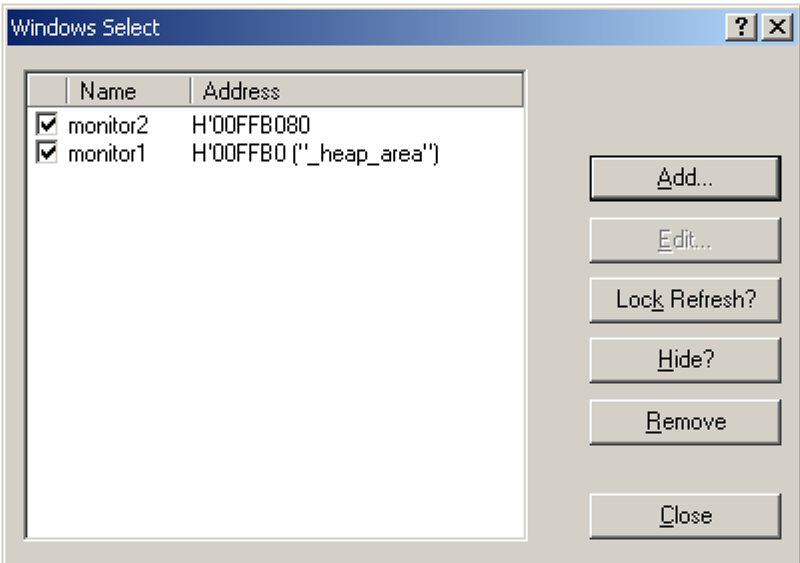


Figure 5.15 [Windows Select] Dialog Box

5.5 Viewing the Current Status

Choose [View -> CPU -> Status] or click the [View Status] toolbar button ( ) to open the [Status] window and see the current status of the debugging platform.

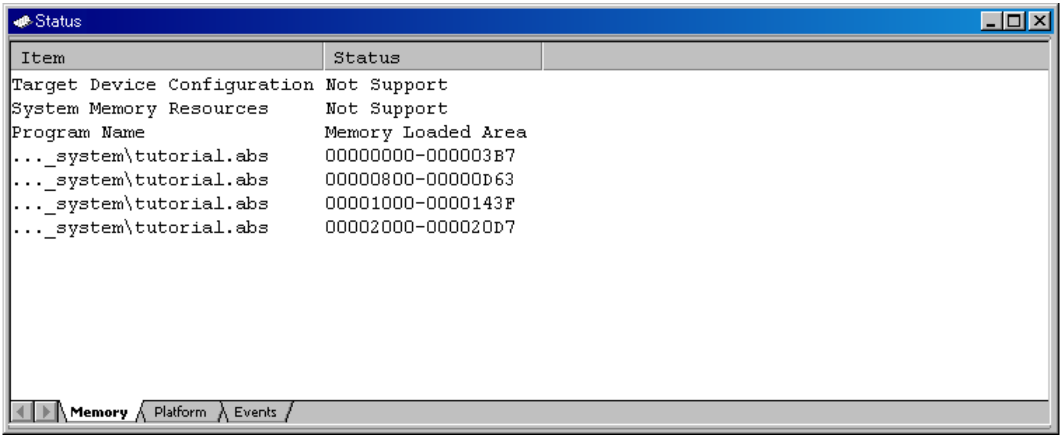


Figure 5.16 [Status] Window

The [Status] window has three sheets:

- [Memory] sheet

  Contains information about the current memory status including the memory-mapping resources and the areas used by the currently loaded object file.

- [Platform] sheet

  Contains information about the status of the emulator, typically including the CPU type and emulation mode, the state of execution, and the statistic information of execution.

- [Events] sheet

  Contains information about the event information, including resource information and breakpoints.

Note: The items that can be set in this dialog box vary according to the emulator in use. For details, refer to the online help.

5.6 Using the Event Points

The emulator has the event point function that performs breaking, tracing, and execution time measurement by specifying more complex conditions along with the PC breakpoints standard for the High-performance Embedded Workshop.

5.6.1 PC Breakpoints

When the instruction at an address specified as a PC breakpoint is fetched, the user program is stopped. Up to 255 points can be set. This setting is available only when synchronized execution, synchronized stepping and synchronized breaks are all enabled. These settings can be made per CPU.

5.6.2 Event Conditions

Event conditions can be used for more complex conditions such as the data condition as well as specification of the single address.

When the condition is satisfied, break conditions are also used as the start/end conditions for performance measurement in addition to halting the user program. When event conditions are used as the start/end conditions for performance measurement, start from setting in the [Performance Analysis] window.

Several event conditions can be combined in a more complex condition. Whether the setting can be made per CPU or is common to both CPUs varies with the device in use.

Notes:
1. When break conditions are used as the start/end conditions for performance measurement, step execution from those conditions is not possible. In addition, when execution is restarted from an address where step operation has been stopped due to satisfaction of a hardware-break address condition for instruction-fetching or a PC breakpoint, further execution would require use of the single-step function, so operation becomes disabled. Restart execution after canceling the address condition for instruction fetching or PC breakpoint.
2. It is not possible to use the break conditions and the start/end conditions for performance measurement at the same time with one channel. If the performance measurement start/end conditions are set, the settings of the break conditions will be disabled.
3. The break conditions that can be set vary according to the emulator in use. For details, refer to the online help.

5.6.3 Opening the [Event] Window

Select [View -> Code -> Eventpoints] or click the [Eventpoints] toolbar button ( ) to open the [Event] window.

The [Event] window has the following two sheets:

[Breakpoint] sheet: Displays the settings made for PC breakpoints. It is also possible to set, modify, and cancel PC breakpoints.

[Event condition] sheet: Displays or sets the settings made for event condition channels.

5.6.4 Setting PC Breakpoints

It is possible to display, modify, and add PC breakpoints on the [Breakpoint] sheet.

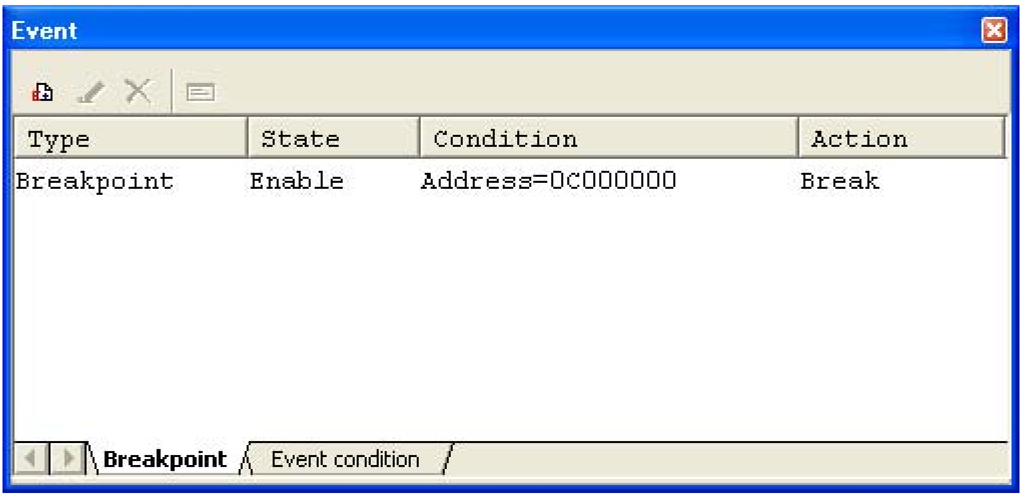


Figure 5.17 [Event] Window ([Breakpoint] Sheet)

This window displays and sets the breakpoints. Items that can be displayed in the sheet are listed below.

[Type] Breakpoint

[State] Whether the breakpoint is enabled or disabled

[Condition] An address that the breakpoint is set

Address = Program counter (Corresponding file name, line, and symbol name)

[Action] Operation of the emulator when a break condition is satisfied

Break: Breaks program execution

Notes: 1. PC breakpoints can only be set when all of synchronized execution, synchronized stepping, and synchronized breaks have been selected.

2. The details of settings differ from product to product. Refer to the online help system for details on the settings for particular products.

When a breakpoint is double-clicked in this window, the [Breakpoint] dialog box is opened and break conditions can be modified.

A popup menu containing the following options is available by right-clicking within the window.

5.6.5 Add

Sets breakpoints. Clicking this item will open the [Breakpoint] dialog box and break conditions can be specified.

5.6.6 Edit

Only enabled when one breakpoint is selected. Select a breakpoint to be edited and click this item. The [Breakpoint] dialog box will open and break conditions can be changed.

5.6.7 Enable

Enables the selected breakpoint(s).

5.6.8 Disable

Disables the selected breakpoint(s). When a breakpoint is disabled, the breakpoint will remain in the list; when specified conditions have been satisfied, a break will not occur.

5.6.9 Delete

Removes the selected breakpoint. To retain the details of the breakpoint but not have it cause a break when its conditions are met, use the Disable option (see section 5.6.8, Disable).

5.6.10 Delete All

Removes all breakpoints.

5.6.11 Go to Source

Only enabled when one breakpoint is selected. Opens the [Source] window at the address of the breakpoint.

5.6.12 [Breakpoint] Dialog Box

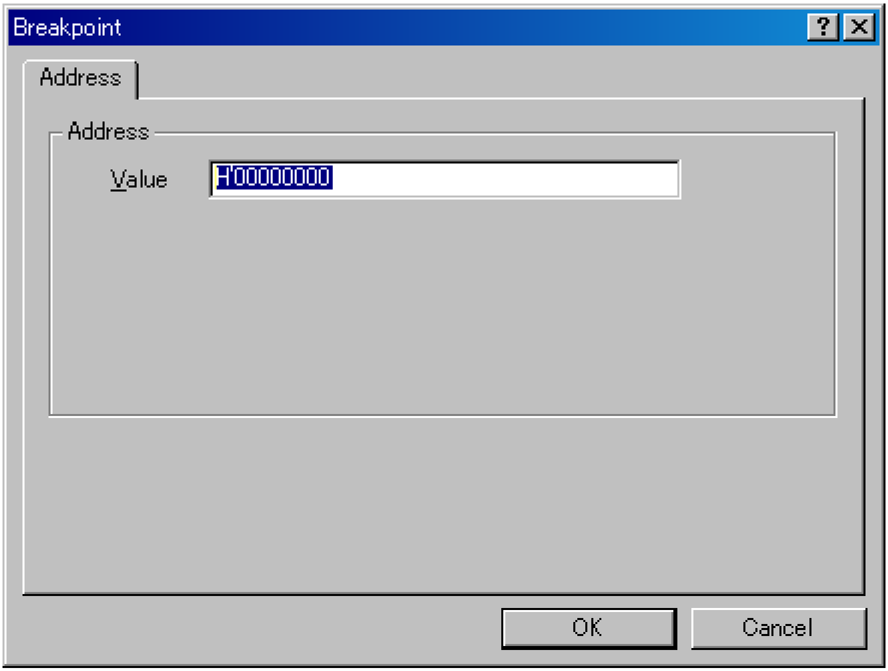


Figure 5.18 [Breakpoint] Dialog Box

This dialog box specifies break conditions.

A breakpoint address to be set is specified in the **[**Value**]** edit box. The PC register can also be specified such as #PC. Up to 255 breakpoints can be specified.

The contents to be set differ depending on the product. For details, refer to the on-line help for each product.

When [Value] is selected, if an overloaded function or class name including a member function is specified in address, the [Select Function] dialog box opens.

Clicking the [OK] button sets the break conditions. Clicking the [Cancel] button closes this dialog box without setting the break conditions.

5.6.13 Setting Event Conditions

On the [Event condition] sheet, the settings for event conditions are displayed, modified, and added.

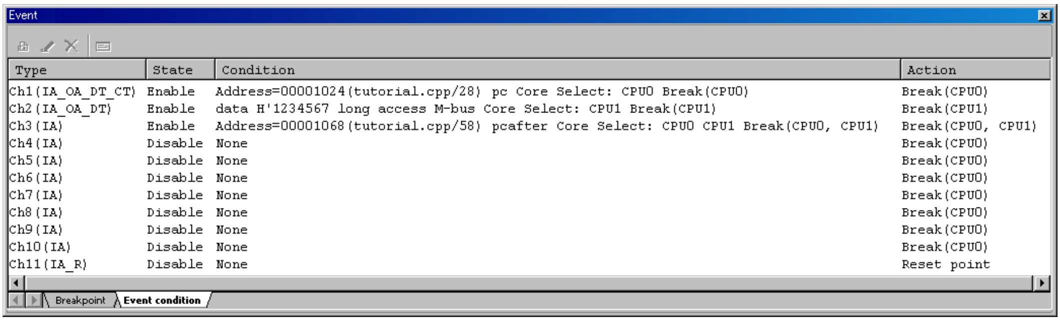


Figure 5.19 [Event] Window ([Event condition] Sheet)

This window displays and sets the event condition. Since the number of channels for detecting conditions and the contents to be set differ depending on the product, refer to the on-line help for each product.

Items that can be displayed in the sheet are listed below.

[Type] Indicates the channel number

[State] Whether the channel is enabled or disabled
Enable: Valid
Disable: Invalid

[Condition] Displays the set conditions. The displayed contents differ depending on the channel.

[Action] Operation of the emulator when a condition is satisfied.
Break: A break in program execution
Trace: Trace information is acquired.
Sequential: The event occurs when the channel conditions are matched in the given sequence.

PAn_Start_Point: Measurement on performance analysis channel n starts.
PAn_End_Point: Measurement on performance analysis channel n ends.

When a channel is double-clicked in this window, the [Event condition] dialog box is opened and the conditions can be modified. For details on the [Event condition] dialog box, refer to the on-line help for each product.

A popup menu containing the following options is available by right-clicking within the window.

5.6.14 Edit...

Only enabled when one channel is selected. Select an event to be edited and click this item. The [Event condition] dialog box will open and conditions can be changed.

5.6.15 Enable

Enables the selected channel(s). A channel that the condition has not been set is not enabled.

5.6.16 Disable

Disables the selected channel(s). When a channel is disabled, the [Action] will not occur even if specified conditions have been satisfied.

5.6.17 Delete

Initializes the condition of the selected channel. To retain the details of the channel but not have it cause the [Action] when its conditions are met, use the Disable option (see section 5.5.16, Disable).

5.6.18 Delete All

Initializes conditions of all channels.

5.6.19 Go to Source

Only enabled when one channel is selected. Opens the [Source] window at address of channel.

If an address value has not been set to the channel, this option cannot be used.

5.6.20 Sequential Conditions

Sets the sequential condition of the channel.

5.6.21 Editing Event Conditions

Handlings for settings other than PC breakpoints and event conditions are common. The following describes examples of such handling.

5.6.22 Modifying Event Conditions

To modify an event condition, select an event condition to be modified, and choose [Edit...] from the popup menu to open the dialog box for the event, which allows the user to modify the event conditions. The [Edit...] menu is only available when one event condition is selected.

5.6.23 Enabling Event Conditions

Select an event condition and choose [Enable] from the popup menu to enable the selected event condition.

5.6.24 Disabling Event Conditions

Select an event condition and choose [Disable] from the popup menu to disable the selected event condition. When an event condition is disabled, the event condition will remain in the list, but an event will not occur when the specified conditions have been satisfied.

5.6.25 Deleting Event Conditions

Select an event condition and choose [Delete] from the popup menu to remove the selected event condition. To retain the event condition but not have it cause an event when its conditions are met, use the [Disable] option (see section 5.6.24, Disabling Event Conditions).

5.6.26 Deleting All Event Conditions

Choose [Delete All] from the popup menu to remove all event conditions.

5.6.27 Viewing the Source Line for Event Conditions

Select an event condition and choose [Go to Source] from the popup menu to open the [Source] or [Disassembly] window at the address of the event condition. The [Go to Source] menu is only available when one event condition that has the corresponding source file is selected.

5.7 Viewing the Trace Information

For the description on the trace function, refer to section 2.2, Trace Functions.

5.7.1 Opening the [Trace] Window

To open the [Trace] window, choose [View -> Code -> Trace] or click the [Trace] toolbar button ( ).

5.7.2 Acquiring Trace Information

The pop up menu of the [Trace window] has a [Settings…] item. If the [Settings…] item is selected, the [Acquisition] dialog box will be displayed. If [I-Trace] is selected as the [Trace Type] in this dialog box, the trace information will be acquired by using the internal trace function.

The acquired trace information is displayed in the [Trace] window.

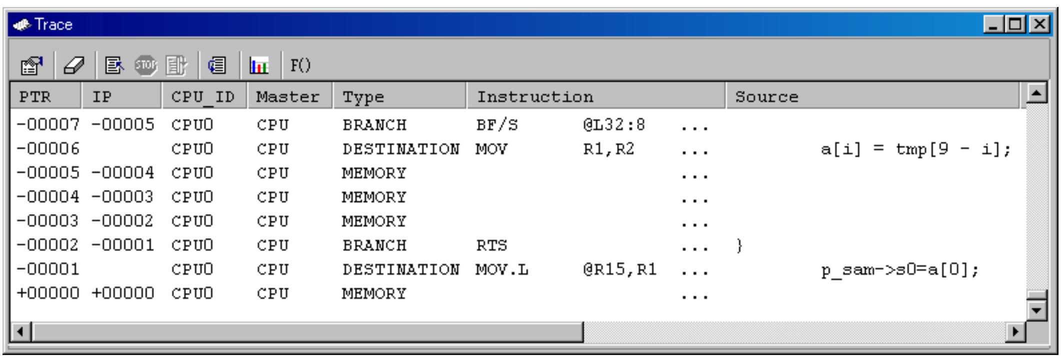


Figure 5.20 [Trace] Window (I-Trace)

This window displays the following trace information items:

[PTR] Pointer to a location in the trace buffer (+0 for the last executed instruction)

[IP] The amount of acquired trace information

[CPU ID] Type of the CPU core:
CPU0: Trace is made for CPU0
CPU1: Trace is made for CPU1

[Master] Master device that generated the event.
CPU: CPU0 was the master.
DMA: The DMAC was the master.

[Type] Type of the trace information
BRANCH: Branch source
DESTINATION: Branch destination
MEMORY: Memory access
PC-RELATIVE: PC-relative access
INSTRUCTION: Instruction fetching from the external space
S_TRACE: Indicates execution of the Trace (x) function
OPERAND PRE-FETCH: Execution of the PREF instruction.

[Branch Type] Type of the branch:
GENERAL: General branch
SUBROUTINE: Subroutine branch
EXCEPTION: Exception branch

[Bus] Display the access type of the cycle:
F-Bus: F bus
M-Bus: M bus
I-Bus: I bus
DMA: Direct memory access

[R/W] Display whether access to data is reading or writing
READ: Read access
WRITE: Write access

[Address] Instruction address

[Data] Display the data value

[Size] Display the size of access:

BYTE: Byte

WORD: Word

LONG: Longword

[Instruction] Instruction mnemonic

[Time stamp] Time stamp, in cycles of Bϕ

[Source] The C/C++ or assembly-language source program

[Label] Label information

Selecting the [Set…] menu in the popup menu of the [Trace] window displays the [Acquisition] dialog box. When [AUD function] is selected in [Trace Type] within the dialog box, the trace information is acquired by using the AUD trace function.

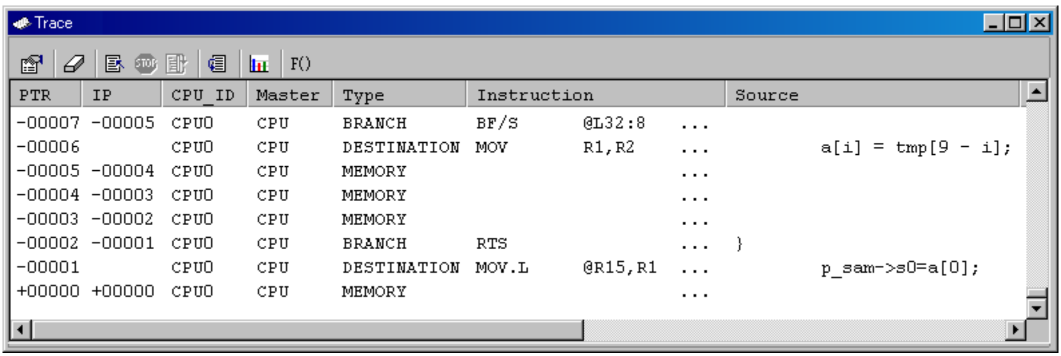


Figure 5.21 [Trace] Window (AUD Trace)

This window displays the following trace information items (some of this information will not be displayed in some products):

[PTR] Pointer to a location in the trace buffer (+0 for the last executed instruction)

[IP] The amount of items of acquired trace information

[CPU_ID] Type of the CPU core

CPU0: Trace is for CPU0

CPU1: Trace is for CPU1

[Master]	Master device that generated the event: CPU: CPU0 was the master
[Type]	Type of the trace information: BRANCH: Branch source DESTINATION: Branch destination MEMORY: Memory access S_TRACE: Executed Trace(x) function LOST: Lost trace information (only in the realtime mode) CPU-WAIT: CPU was waiting for the output of the trace information (only in the non-realtime mode)
[Branch Type]	Type of the branch: GENERAL: General branch SUBROUTINE: Subroutine branch EXCEPTION: Exception branch
[Bus]	Display the access type of the cycle: M-Bus: M bus I-Bus: I bus
[R/W]	Display whether access to data is reading or writing READ: Read access WRITE: Write access
[Address]	Instruction address (AUD trace: If there is no base address in the trace buffer, display the difference only)
[Data]	Display the data value.
[Size]	Display the size of access: BYTE: Byte WORD: Word LONG: Longword
[Instruction]	Instruction mnemonic
[Timestamp]	No timestamp, value is fixed to 0
[Source]	The C/C++ or assembly-language source program
[Label]	Label information

Note: Since the displayed contents differ depending on the product, refer to each product's online help. Some supported chips do not have the AUD tracing function.

The following items may be displayed for some target devices for debugging.
Refer to the additional document ″Supplementary Information on Using the SHxxxx″ and the online help for the specifications of the device.

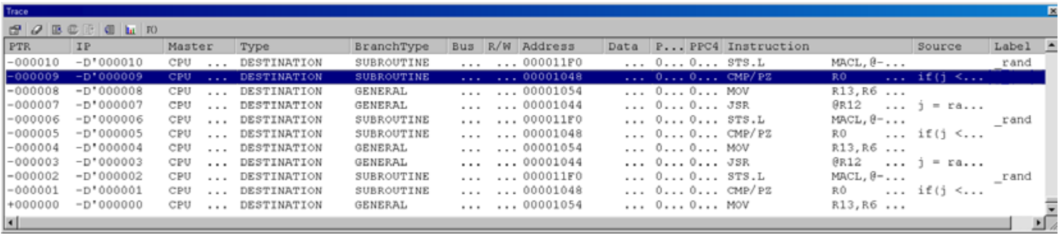


Figure 5.22 [Trace] Window (Type 2)

[PTR] Pointer to lines within the trace buffer (the value for the last instruction to have been executed is + 0

[IP] The number of lines of acquired trace information

[Master] (Bus Master) The number of CPU which accessed the memory, or the type of bus master

[Type] Type of the trace information:
BRANCH: Branch source
DESTINATION: Branch destination
MEMORY: Memory access
S_TRACE: Executed Trace(x) function
LOST: Lost trace information (only in the realtime mode)
CPU-WAIT: CPU was waiting for the output of the trace information
(only in the non-realtime mode)

[Branch Type] Type of branch:
GENERAL: General branch
SUBROUTINE: Subroutine branch
EXCEPTION: Exceptional branch
[Bus] Indicates the bus over which access proceeded
[R/W] Indicates whether access to data was reading or writing
[Address] Address

[Data] Indicates the accessed data
When [Type] is S_TRACE, this is the value of the variable x of the trace(x) function
[PPC] Output of the performance counter
[Instruction] Instruction mnemonic
[Source] C/C ++ or assembler source code
[Label] Label information

It is possible to hide any column not necessary in the [Trace] window. Selecting a column you want to hide from the popup menu displayed by clicking the right-hand mouse button on the header column hides that column. To display the hidden column, select the column from the said popup menu again. Dragging the column with the mouse can change the display order.

5.7.3 Specifying Trace Acquisition Conditions

The capacity of the trace buffer is limited. When the buffer becomes full, the oldest trace information is overwritten. Setting the trace acquisition condition allows acquisition of useful trace information and effective use of the trace buffer.

The trace acquisition condition is set in the [Acquisition] dialog box that is displayed by selecting [Acquisition…] from the popup menu.

Whether information displayed in dialog boxes and the details of settings apply per CPU or are common to the CPUs differs with the device in use.
The dialog box below is for a device where the settings are common to the CPUs.
For specifications of the various devices, refer to the additional document, ″Supplementary Information on Using the SHxxxx″ or the online help.

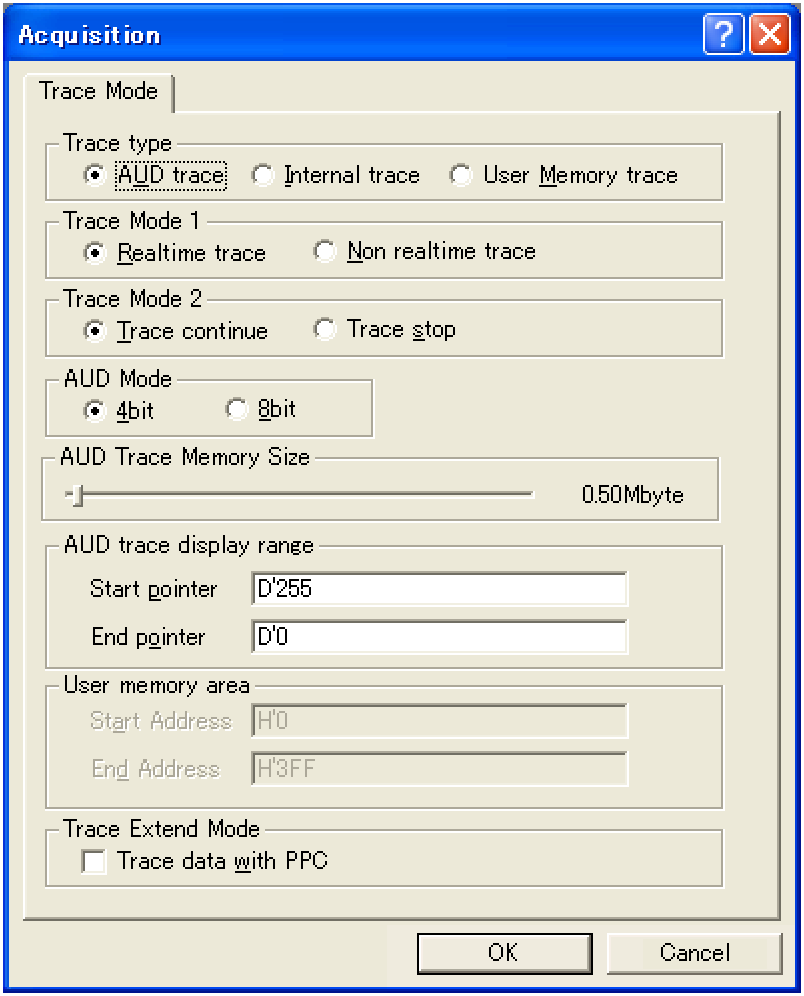


Figure 5.23 [Acquisition] dialog box (on [Trace mode] page)

[Trace type] Selects the type of tracing function

[AUD trace]: Use AUD tracing

[Internal trace]: Use internal tracing

[User memory trace]: Use the function for the output of trace-data memory.

[Trace Mode 1] Decides the kind of operation for consecutive trace information

This can be only be used when [AUD trace] or [User Memory trace] is selected.

[Realtime trace]: Some trace information will not be output.
[Non realtime trace]: The CPU is made to wait for the output of trace data.

[Trace Mode 2]: Decides whether or not operation continues after the trace buffer of the emulator is full. This can only be used when [AUD trace] or [User Memory trace] is selected.

[Trace continue]: Continue to acquire the latest information by overwriting the oldest trace information.

[Trace stop]: Acquisition of trace information stops when the buffer is full.

[AUD Mode]: Depending on the target device for debugging, an 8-bit AUD pin mode may be selectable. Refer to the additional document, ″Supplementary Information on Using the SHxxxx″ and the online help for the specifications of devices. This mode is only valid when [AUD trace] is selected.

[AUD trace Memory Size]: Set the size of the trace buffer memory for the emulator. This mode is only valid when [AUD trace] is selected.

[AUD trace display range]: Set the range for display in the trace window. This mode is only valid when [AUD trace] is selected.
[Start pointer] Traced data are displayed from the value set here.
[End pointer] Traced display are displayed up to the value set here.

[User Memory area]: Set the range for display in the trace window. This mode is only valid when [User Memory trace] is selected.
[Start] Specify the first address of the region of memory where the results of tracing are to be written.
[End address] Specify the last address of the region of memory where the results of tracing are to be written.

[Trace Extend Mode]:
[Trace data is PPC] Output values of the performance counter to the trace window.

The specified contents are set by clicking on the [OK] button. If the [Cancel] button is clicked, the dialog box is closed without the settings being made.

Also, settings other than those in the [Display Type] group box are common to the High-performance Embedded Workshops for CPU0 and CPU1. Settings in the [Display Type] group box are not common to the High-performance Embedded Workshops for CPU0 and CPU1.
Refer to the additional document ″Supplementary Information on Using the SHxxxx″ or the online help for the specifications of devices.

(1) [Trace mode] page

Sets trace acquisition conditions.

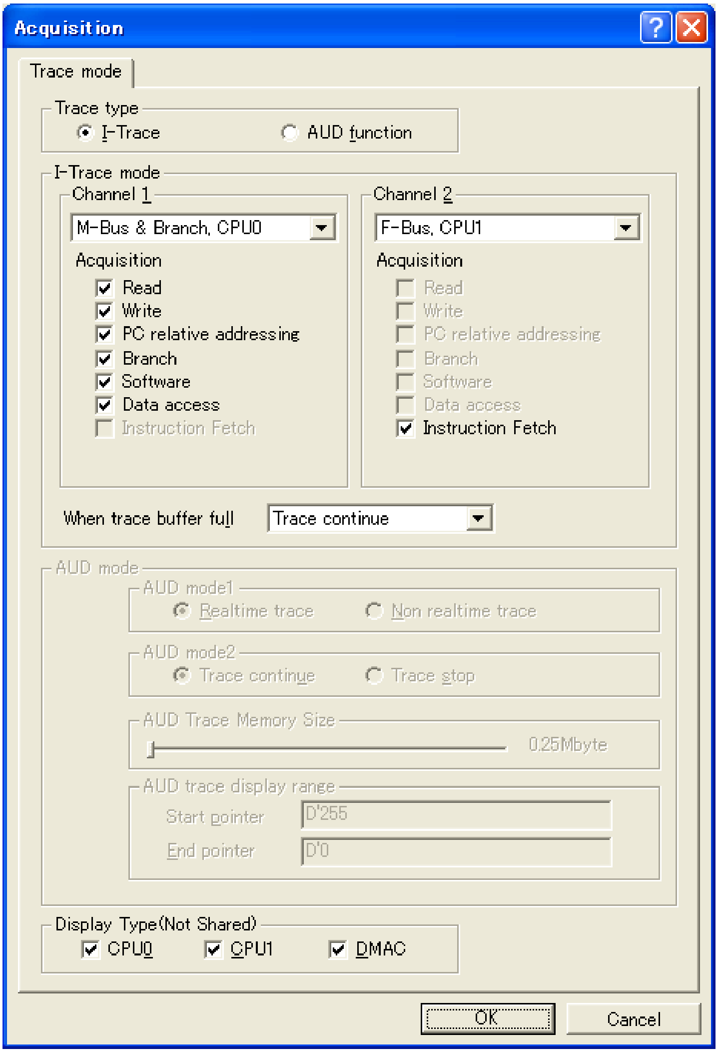


Figure 5.24 [Acquisition] Dialog Box ([Trace mode] Page)

This dialog box specifies the methods and conditions for the acquisition of trace information.

The following items can be set:

[Trace mode]: Set the trace mode conditions

[Trace Type]: Selects the type of trace function.
[I-Trace]: The internal trace function is used.
[AUD function]: The AUD trace function is used.

- [Trace mode] page for internal trace ([I-trace] selected)

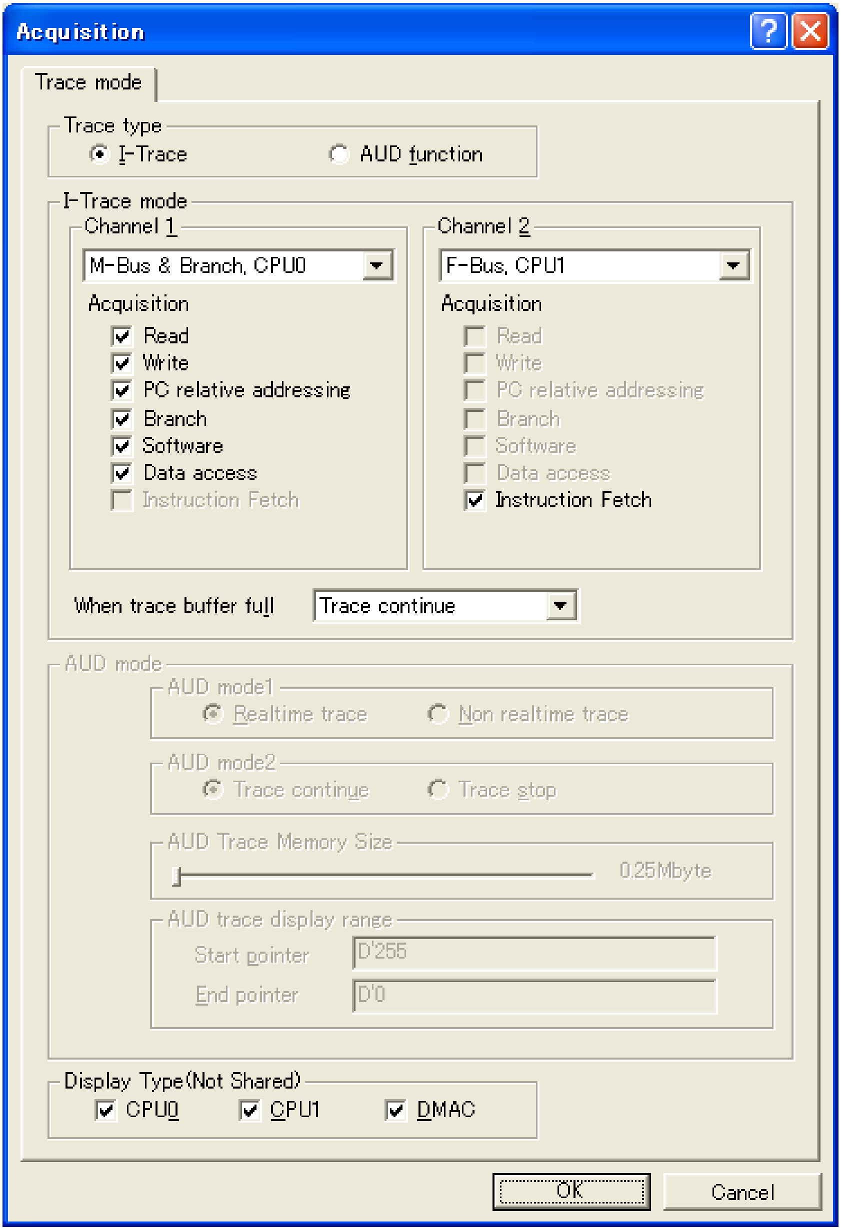


Figure 5.25 [Acquisition] Dialog Box (when I-trace has been selected)

This dialog box is used to specify conditions for the acquisition of trace information.

When [I-Trace] has been selected:

[I-Trace mode]: Set the bus and other conditions for acquisition of the internal trace.

[Channel 1]	: Set the trace acquisition conditions for channel 1. The same conditions cannot be set for channels 1 and 2.	
[Type]	[M-Bus & Branch, CPU0]	: Trace information is acquired with M-bus access and branching by CPU0 as conditions.
	[I-Bus, CPU0]	: Trace information is acquired with I-bus access by CPU0 as condition.
	[F-Bus, CPU0]	: Trace information is acquired with F-bus access by CPU0 as a condition.
	[M-Bus & Branch, CPU1]	: Trace information is acquired with M-bus access and branching by CPU1 as conditions.
	[I-Bus, CPU1]	: Trace information is acquired with I-bus access by CPU1 as a condition.
	[F-Bus, CPU1]	: Trace information is acquired with F-bus access by CPU1 as a condition.
	[DMAC]	: Acquires trace information on access by the DMAC.
[Channel 2]	: Set the trace acquisition conditions for channel 2. The same conditions cannot be set for channels 1 and 2.	
[Type]	[M-Bus & Branch, CPU0]	: Trace information is acquired with M-bus access and branching by CPU0 as conditions.
	[I-Bus, CPU0]	: Trace information is acquired with I-bus access by CPU0 as a condition.
	[F-Bus, CPU0]	: Trace information is acquired with F-bus access by CPU0 as a condition.
	[M-Bus & Branch, CPU1]	: Trace information is acquired with M-bus access and branching by CPU1 as conditions.
	[I-Bus, CPU1]	: Trace information is acquired with I-bus access by CPU1 as a condition.

	[F-Bus, CPU1]	: Trace information is acquired with F-bus access by CPU1 as a condition.
	[None]	: No conditions are set.
[Acquisition]	: Set the conditions for acquisition of the internal trace information.	
	[Read]	: Trace information is acquired on reading.
	[Write]	: Trace information is acquired on writing.
	[PC relative addressing]	: Trace information is acquired with the execution address.
	[Branch]	: Trace information is acquired on the branch.
	[Software]	: Software tracing is a condition.
	[Data access]	: Trace information is acquired on data access
	[Instruction Fetch]	: Trace information is acquired on cycles of instruction fetching.
[When trace buffer full]	: Specify the operation when the trace buffer is completely full.	
	[Trace continue]	: Keep acquiring trace information. The earliest contents of the trace buffer are overwritten.
	[Trace stop]	: Stop trace acquisition.
	[Break (CPU0)]	: Break CPU0.
	[Break (CPU1)]	: Break CPU1.
	[Break (CPU0, CPU1)]	: Break CPU0 and CPU1.

- [Trace mode] page for AUD trace ([AUD function] selected)

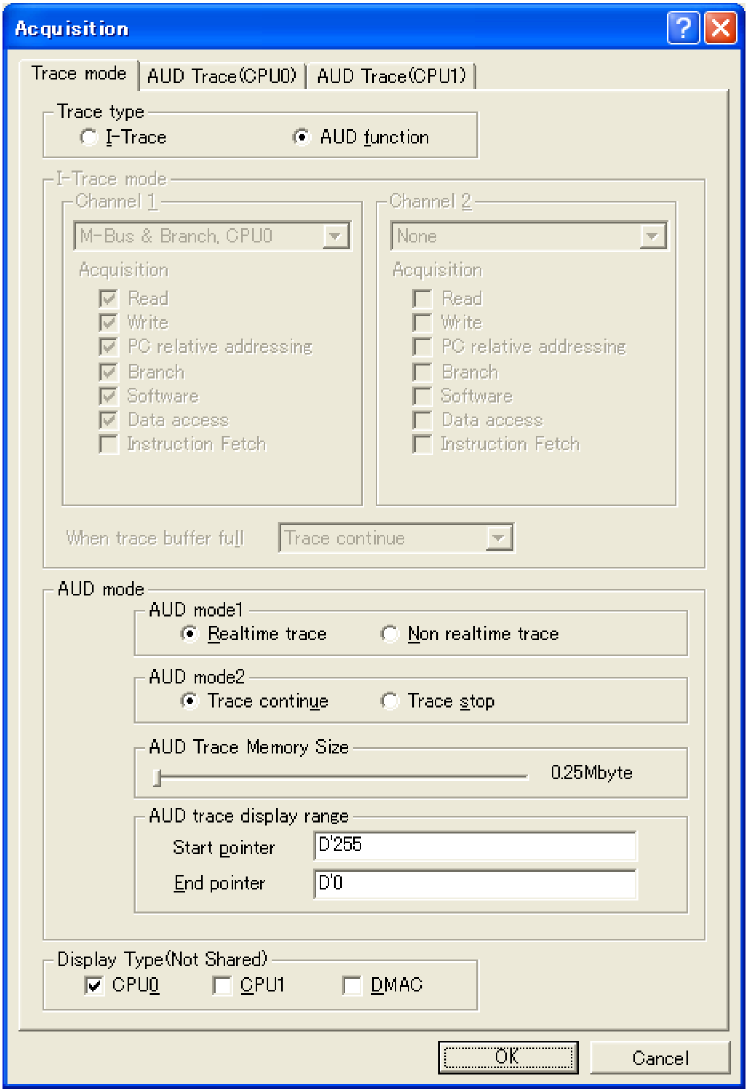


Figure 5.26 [Acquisition] Dialog Box (when AUD function has been selected)

When [AUD function] has been selected

[AUD mode1]	: Selection of realtime or non-realtime trace acquisition	
	[Realtime trace]	: When the trace information is being generated intensely that the output from the AUD pin incapable of keeping up, the CPU temporarily suspends the output of trace information. Therefore, although the user program is run in real time, the acquisition of some trace information might not be possible.
	[Non realtime trace]	: When trace information is being generated so intensely that the output from the AUD pin is incapable of keeping up, CPU operations are temporarily suspended and the output of trace information takes priority. In such cases, the realtime characteristics of the user program are lost.
[AUD mode2]	: Specify the operation when the trace buffer is completely full.	
	[Trace continue]	: Continue trace acquisition. The earliest trace information will be overwritten.
	[Trace stop]	: Once the trace buffer is full, trace information is not acquired.
[AUD trace Memory Size]	: Specify the size of the trace buffer memory of the emulator.	
[AUD trace display range]	: Specify the range for which AUD trace information will be displayed.	
	[Start pointer]	: Set the pointer to the start of the range for AUD tracing.
	[End pointer]	: Set the pointer to the end of the range for AUD tracing.

Common to [I-Trace]/ [AUD function]

[Display Type]	Specify the information to be displayed in the trace window.	
	[CPU0]	: Display the trace information for which the CPU identifier (CPU ID) is CPU0.
	[CPU1]	: Display the trace information for which the CPU identifier (CPU ID) is CPU1.
	[DMAC]	: Display the trace information for which the CPU identifier (CPU ID) is DMA.

(2) [AUD Trace(CPU0)] or [AUD Trace(CPU1)] page

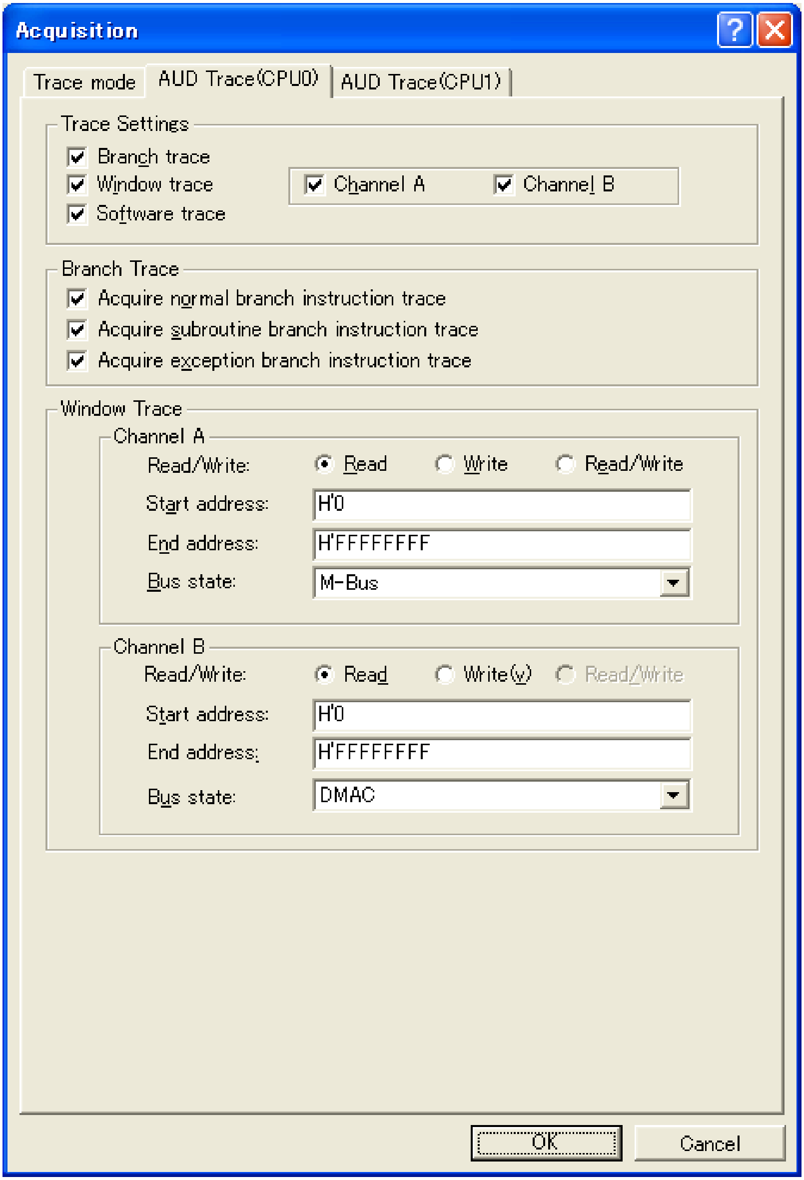


Figure 5.27 [Acquisition] Dialog Box ([AUD Trace (CPU0)] Page)

The settings of either the AUD trace (CPU0) or the AUD trace (CPU1)

Select the AUD trace conditions from the [Trace Settings] when the AUD function is selected.

[Branch trace] : Trace information is acquired by the branch source and the branch destination as conditions.

[Window trace] : Window trace functions. Memory access information is acquired within the specified area.

- [Channel A] : Set whether acquire the window trace information from channel A or not.
- [Channel B] : Set whether acquire the window trace information from channel B or not.

[Software trace] : Software trace functions. Trace is acquired with the software trace instructions.

When checking on the [Branch trace] of the [Trace settings], select the acquisition branch conditions.

[Acquire normal branch instruction trace] : Specify the normal branch as the branch conditions.

[Acquire subroutine instruction trace] : Specify the subroutine branch as the branch conditions.

[Acquire exception branch instruction trace] : Select the exception branch.

When checking on the [Window trace] of the [Trace settings], set the conditions for [Channel A] and [Channel B] of the [Window trace] group box.

[Channel A] : Set the conditions to acquire the AUD trace.

- [Read/Write] : Sets tracing of read or write access or both.
  - [Read] : Trace information is acquired on reading.
  - [Write] : Trace information is acquired on writing.
  - [Read/Write] : Trace information is acquired on reading and writing.
- [Start address] : Sets an address range for the tracing of data access. The start address is set here.
- [End address] : Sets an address range for the tracing of data access. The end address is set here.

[Channel A]	[Bus state]	: Sets a bus to acquire the window trace.	
		[M-Bus]	: Select the M-Bus.
		[DMAC]	: Select the DMA.
[Channel B]	Set the conditions to acquire the AUD trace.		
	[Read/Write]	: Sets tracing of read or write access or both.	
		[Read]	: Trace information is acquired on reading.
		[Write]	: Trace information is acquired on writing.
		[Read/Write]	: Trace information is acquired on reading and writing.
	[Start address]	: Sets an address range for the tracing of data access. The end address is set here.	
	[End address]	: Sets an address range for the tracing of data access. The end address is set here.	
	[Bus state]	: Sets a bus to acquire the window trace.	
		[M-Bus]	: Select the M-Bus.
		[DMAC]	: Select the DMA.

5.7.4 Searching for a Trace Record

Use the [Trace Find] dialog box to search for a trace record. To open this dialog box, choose [Find...] from the popup menu.

These settings are not common to the High-performance Embedded Workshops for the individual CPU. That is, each High-performance Embedded Workshop has its own settings.

The [Trace Find] dialog box has the following options:

Table 5.2 [Trace Find] Dialog Box Pages

Page	Description
[General]	Sets the range for searching.
[Address]	Sets an address condition.
[Data]	Sets a data condition.
[Type]	Selects the type of trace information.
[Bus]	Selects the type of a bus.
[R/W]	Selects the type of access cycles.
[Size]	Selects a unit of access.

Note: Items other than [General] and [Address] vary according to the emulator in use. For details, refer to the online help.

Clicking the [OK] button after setting conditions in those pages stores the settings and starts searching. Clicking the [Cancel] button closes this dialog box without setting conditions.

When a trace record that matches the search conditions is found, the line for the trace record will be highlighted. When no matching trace record is found, a message dialog box will appear.

Only the trace information that satisfies all the conditions set in above pages will be searched.

If a search operation is successful, selecting [Find Next] from the popup menu will move to the next found item.

(1) [General] page

Set the range for searching.

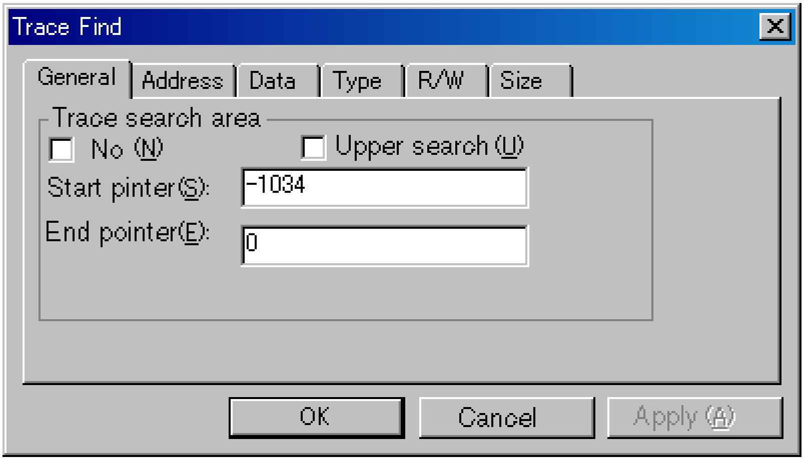


Figure 5.28 [Trace Find] Dialog Box ([General] Page)

[Trace search range]: Sets the range for searching.

[No]: Searches for information that does not match the conditions set in other pages when this box is checked.

[Upper search]: Searches upwards when this box is checked.

[Start pointer]: Enters a PTR value to start a search.

[End pointer]: Enters a PTR value to end a search.

Note: Along with setting the range for searching, PTR values to start and end searching can be set in the [Start PTR] and [End PTR] options, respectively.

(2) [Address] page

Set address condition.

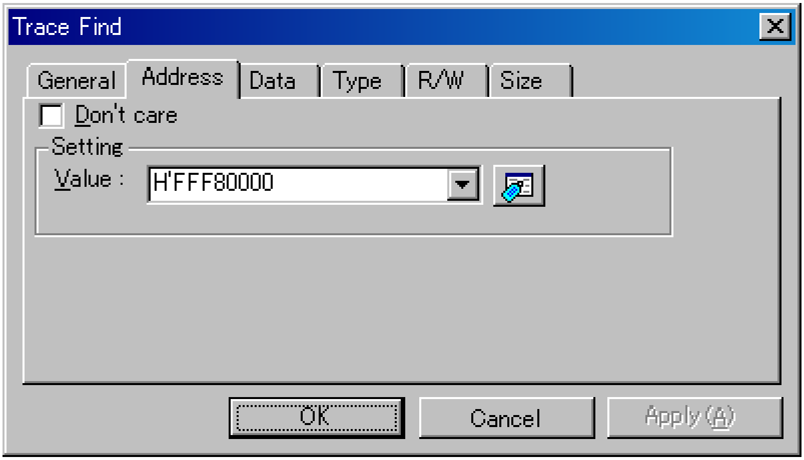


Figure 5.29 [Trace Find] Dialog Box ([Address] Page)

[Don't care]: Detects no address when this box is checked.

[Setting]: Detects the specified address.

[Value]: Enter the address value (not available when [Don’t care] has been checked).

(3) [Data] page

Set a data condition.

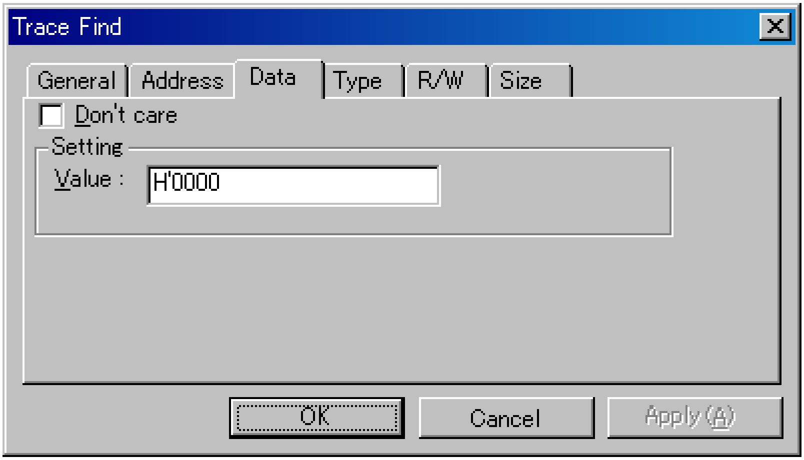


Figure 5.30 [Trace Find] Dialog Box ([Data] Page)

[Don't care]: Detects no data when this box is checked.

[Setting]: Detects the specified data.

[Value]: Enter the data value (not available when [Don’t care] has been checked).

(4) [R/W] page

Select the type of access cycles.

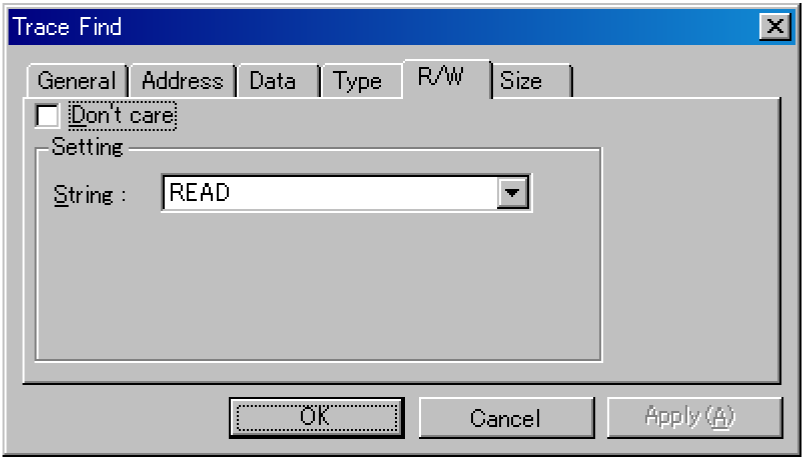


Figure 5.31 [Trace Find] Dialog Box ([R/W] Page)

[Don't care]: Detects no read/write condition when this box is checked.

[Setting]: Detects the specified read/write condition.

[String]: Select a read/write condition (not available when [Don’t care] has been checked).

READ: Read cycle

WRITE: Write cycle

(5) [Type] page

Select the type of Trace information.

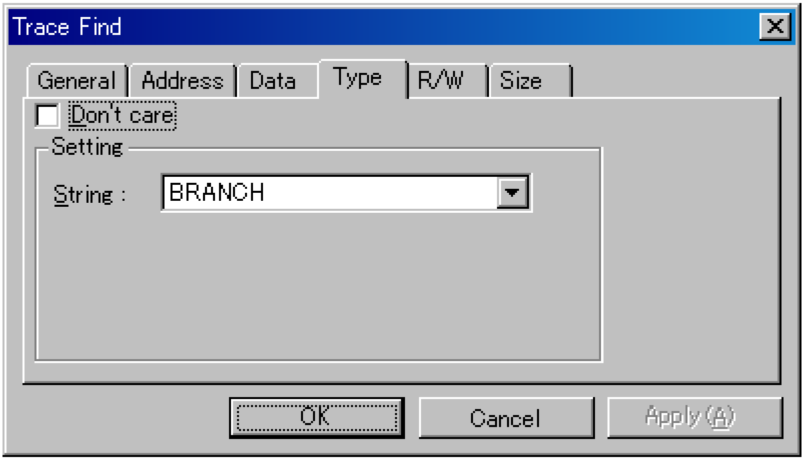


Figure 5.32 [Trace Find] Dialog Box ([Type] Page)

[Don't care]: Detects no type condition when this box is checked.

[Setting]: Detects the specified type condition.

[String]: Select a type condition (not available when [Don’t care] has been checked).

(6) [Size] page

Select a unit of access.

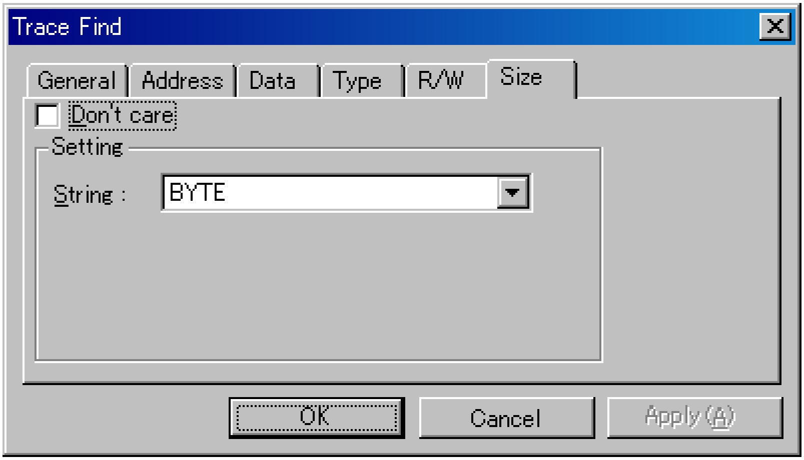


Figure 5.33 [Trace Find] Dialog Box ([Size] Page)

[Don't care]: Detects no size condition when this box is checked.

[Setting]: Detects the specified size condition.

[String]: Select a size condition (not available when [Don’t care] has been checked).

5.7.5 Clearing the Trace Information

When [Clear] is selected from the popup menu, the trace buffer that stores the trace information becomes empty. If several [Trace] windows are open, all [Trace] windows will be cleared as they all access the same buffer.

5.7.6 Saving the Trace Information in a File

Select [Save...] from the popup menu to open the [Save As] file dialog box, which allows the user to save the information displayed in the [Trace] window as a text file. A range can be specified based on the [PTR] number (saving the complete buffer may take several minutes). Note that this file cannot be reloaded into the [Trace] window.

Note: In filtering of trace information, the range to be saved cannot be selected. All the trace information displayed in the [Trace] window after filtering will be saved. Select a filtering range on the [General] page in the [Trace Filter] dialog box if you want to save the selected range. For details on the filtering function, refer to section 5.7.10, Extracting Records from the Acquired Information.

5.7.7 Viewing the [Editor] Window

The [Editor] window corresponding to the selected trace record can be displayed in the following two ways:

- Select a trace record and choose [View Source] from the popup menu.
- Double-click a trace record.

The [Editor] or [Disassembly] window opens and the selected line is marked with a cursor.

5.7.8 Trimming the Source

Choose [Trim Source] from the popup menu to remove the white space from the left side of the source.

When the white space is removed, a check mark is shown to the left of the [Trim Source] menu. To restore the white space, choose [Trim Source] while the check mark is shown.

5.7.9 Temporarily Stopping Trace Acquisition

To temporarily stop trace acquisition during execution of the user program, select [Halt] from the popup menu. This stops trace acquisition and updates the trace display. Use this method to check the trace information without stopping execution of the user program.

5.7.10 Extracting Records from the Acquired Information

Use the filtering function to extract the records you need from the acquired trace information. The filtering function allows the trace information acquired by hardware to be filtered by software. Unlike the settings made in the [Trace Acquisition] dialog box for acquiring trace information by conditions, changing the settings for filtering several times to filter the acquired trace information allows easy extraction of necessary information, which is useful for analysis of data. The content of the trace buffer will not be changed even when the filtering function is used. Acquiring useful information as much as possible by the [Trace Acquisition] settings improves the efficiency in analysis of data because the capacity of the trace buffer is limited.

Use the filtering function in the [Trace Filter] dialog box. To open the [Trace Filter] dialog box, select [Filter…] from the popup menu.Selects a unit of access

These settings are not common to the High-performance Embedded Workshops for CPU0 and CPU1. That is, each High-performance Embedded Workshop has its own settings.

The [Trace Filter] dialog box has the following pages:

Table 5.3 [Trace Filter] Dialog Box Pages

Page	Description
[General]	Selects the range for filtering.
[Address]	Sets an address condition.
[Data]	Sets a data condition.
[Type]	Selects the type of trace information.
[Bus]	Selects the type of a bus.
[R/W]	Selects the type of access cycles.
[Size]	Selects a unit of access.

Note: Items other than [General] and [Address] vary according to the emulator in use. For details, refer to the online help.

Set filtering conditions and then press the [OK] button. This starts filtering according to the conditions. Clicking the [Cancel] button closes the [Trace Filter] dialog box, which holds the settings at the time when the dialog box was opened.

In filtering, only the trace information that satisfies one or more filtering conditions set in the above pages will be displayed in the [Trace] window.

Filtering conditions can be changed several times to analyze data because the content of the trace buffer is not changed by filtering.

(1) [General] page

Set the range for filtering.

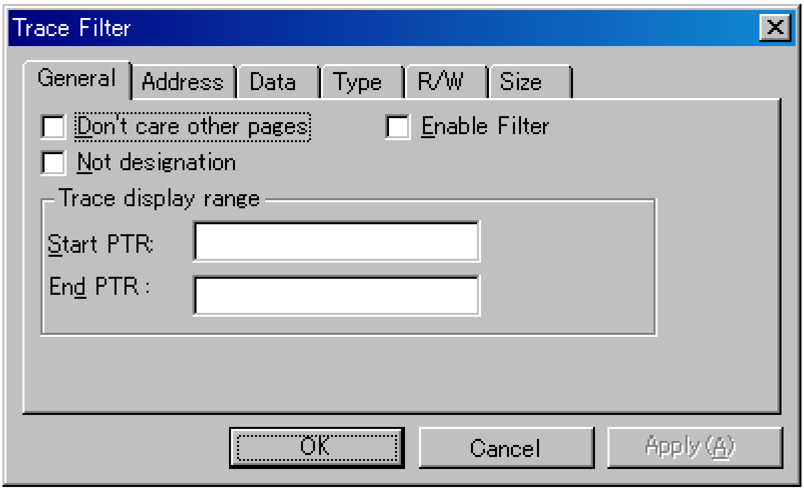


Figure 5.34 [Trace Filter] Dialog Box ([General] Page)

[Don't care other pages]:	Only selects the cycle number when this box is checked. Other options become invalid.
[Enable Filter]:	Enables the filter when this box is checked.
[Not designation]:	Filters information that does not match the conditions set in those pages when this box is checked.
[Trace display range]:	Sets the range for filtering.

[Start PTR]: Enters a PTR value to start filtering.

[End PTR]: Enters a PTR value to end filtering.

Note: Along with setting the range for filtering, PTR values to start and end filtering can be set in the [Start PTR] and [End PTR] options, respectively.

(2) [Address] page

Set an address condition.

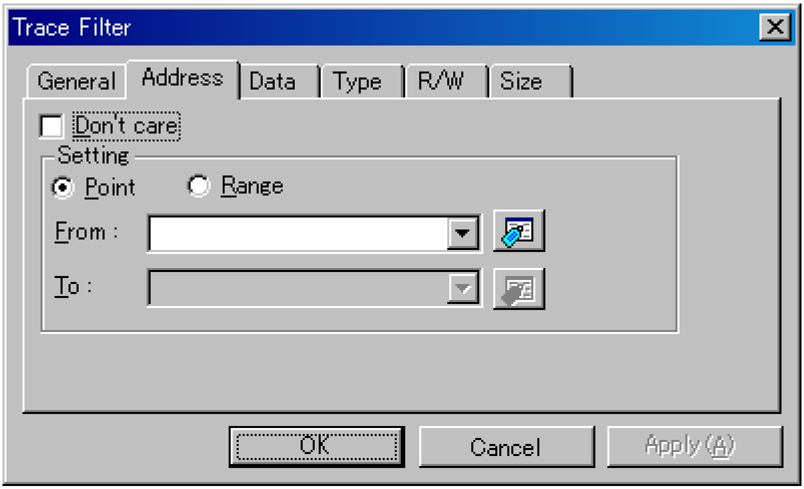


Figure 5.35 [Trace Filter] Dialog Box ([Address] Page)

[Don't care]: Detects no address when this box is checked.

[Setting]: Detects the specified address.

[Point]: Specifies a single address (not available when [Don’t care] has been checked).

[Range]: Specifies an address range (not available when [Don’t care] has been checked).

[From]: Enter a single address or the start of the address range (not available when [Don’t care] has been checked).

[To]: Enter a single address or the end of the address range (only available when [Range] has been selected).

Note: Along with setting the address range, the start and end of the address range can be set in the [From] and [To] options, respectively.

(3) [Data] page

Set a data condition.

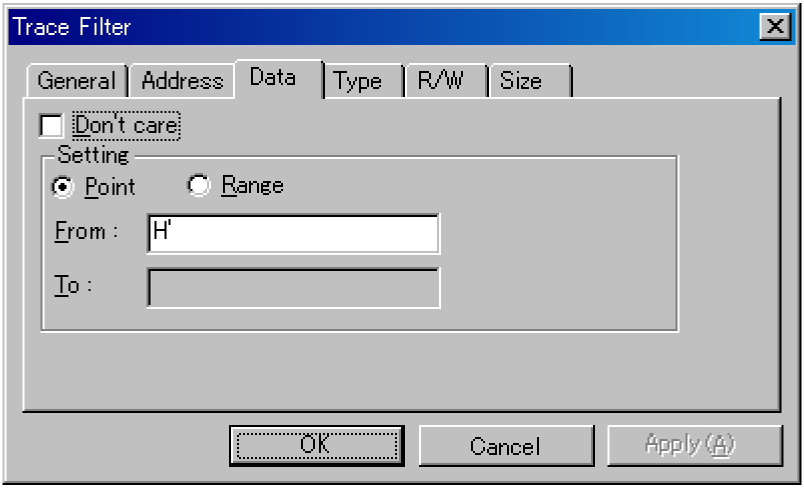


Figure 5.36 [Trace Filter] Dialog Box ([Data] Page)

[Don't care]: Detects no data when this box is checked.

[Setting]: Detects the specified data.

[Point]: Specifies single data (not available when [Don’t care] has been checked).

[Range]: Specifies a data range (not available when [Don’t care] has been checked).

[From]: Enter single data or the minimum value of the data range (not available when [Don’t care] has been checked).

[To]: Enter the maximum value of the data range (only available when [Range] has been selected).

Note: Along with setting the data range, the minimum and maximum values can be set in the [From] and [To] options, respectively.

(4) [R/W] page

Select the type of access cycles.

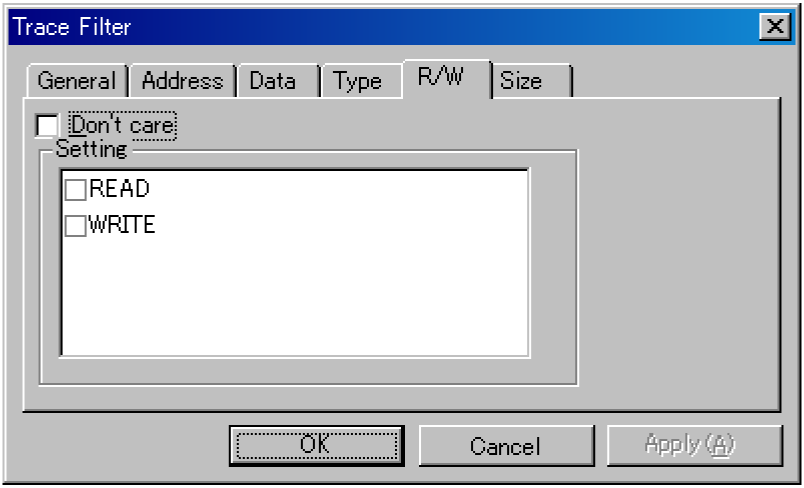


Figure 5.37 [Trace Filter] Dialog Box ([R/W] Page)

[Don't care]: Detects no read/write condition when this box is checked.

[Setting]: Detects the specified read/write condition.

READ: Detects read cycles when this box is checked (not available when [Don’t care] has been checked).

WRITE: Detects write cycles when this box is checked (not available when [Don’t care] has been checked).

(5) [Type] page

Select the type of Trace information. The selection is not available when a time stamp is acquired.

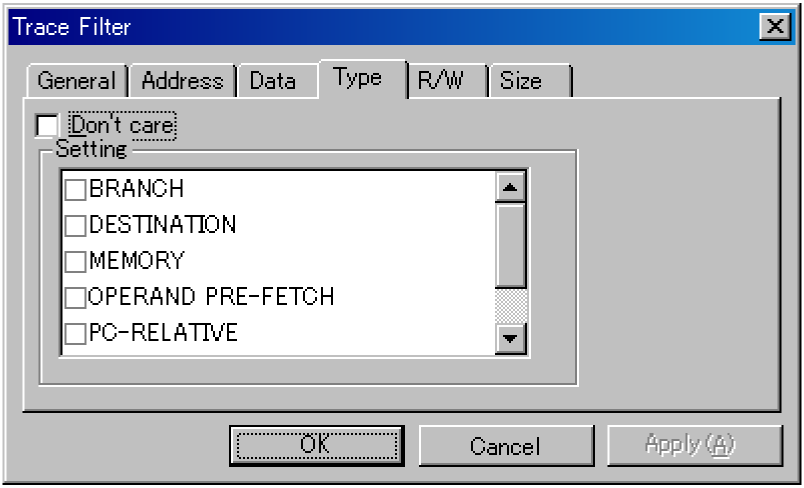


Figure 5.38 [Trace Filter] Dialog Box ([Type] Page)

[Don't care]: Detects no type condition when this box is checked.

[Setting]: Detects the specified type condition (not available when [Don’t care] has been checked).

(6) [Size] page

Select a unit of the access.

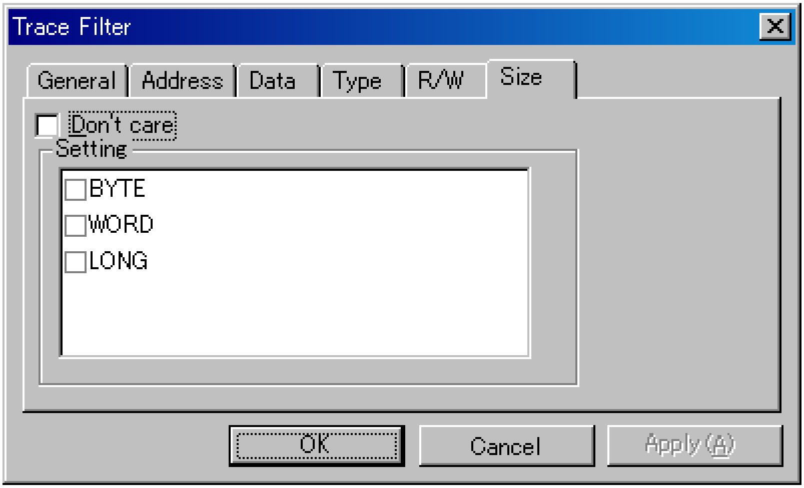


Figure 5.39 [Trace Filter] Dialog Box ([Size] Page)

[Don't care]: Detects no size condition when this box is checked.

[Setting]: Detects the specified size condition (not available when [Don’t care] has been checked).

5.7.11 Analyzing Statistical Information

Choose [Statistic…] from the popup menu to open the [Statistic] dialog box and analyze statistical information under the specified conditions.

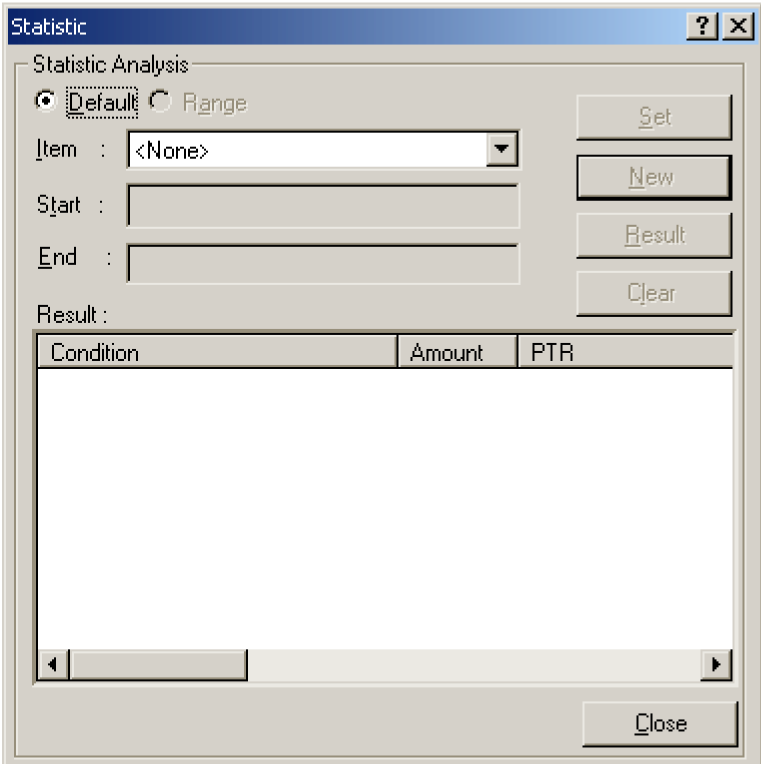


Figure 5.40 [Statistic] Dialog Box

[Statistic Analysis]: Setting required for analysis of statistical information.

[Default]: Sets a single input value or character string.

[Range]: Sets the input value or character string as a range.

[Item]: Sets the item for analysis.

[Start]: Sets the input value or character string. To set a range, the start value must be specified here.

[End]:	Specify the end value if a range has been set (only available when [Range] has been selected).
[Set]:	Adds a new condition to the current one.
[New]:	Creates a new condition.
[Result] button:	Obtains the result of statistical information analysis.
[Clear]:	Initializes the settings.
[Result] list box:	Displays all conditions and results of statistical information analysis.
[Close]:	Closes this dialog box. All the results displayed in the [Result] list will be cleared.

This dialog box allows the user to analyze statistical information concerning the trace information. Set the target of analysis in [Item] and the input value or character string by [Start] and [End]. Click the [Result] button after setting a condition by pressing the [New] or [Add] button to analyze the statistical information and display its result in the [Result] list.

Note: In this emulator, only [PTR] can be set as a range. Each of other items must be specified as a character string. In analysis of statistical information, character strings are compared with those displayed in the [Trace] window. Only those that completely match are counted. Note, however, that this test is not case sensitive. The number of blanks will not be cared either.

5.7.12 Extracting Function Calls from the Acquired Trace Information

To extract function calls from the acquired trace information, select [Function Call…] from the popup menu. The [Function Call Display] dialog box will be displayed.

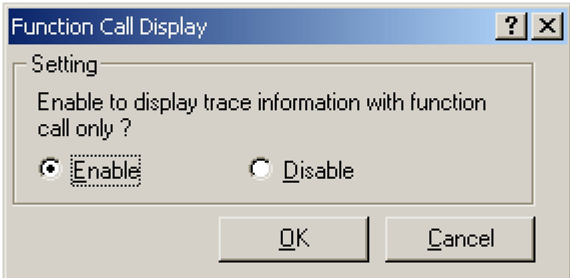


Figure 5.41 [Function Call Display] Dialog Box

[Setting]: Selects whether or not to extract function calls.

[Enable]: Extracts function calls.

[Disable]: Does not extract function calls.

When [Enable] is selected, only the cycles that include function calls are extracted for display from the acquired trace information. The content of the trace buffer is not changed by extraction of function calls. Using this function for the trace information that includes function calls allows the user to know the order of function calls.

5.8 Analyzing Performance

Use the performance analysis function to measure execution performance. The performance analysis function does not affect the realtime operation because it measures execution performance in the specified range by using the on-chip circuit for performance measurement.

These settings are not common to the High-performance Embedded Workshops for CPU0 and CPU1. That is, each High-performance Embedded Workshop has its own settings.

Note: The measurement conditions and the number of channels differ depending on the product.

5.8.1 Opening the [Performance Analysis] Window

Choose [View -> Performance -> Performance Analysis] or click the [PA] toolbar button ( ) to open the [Select Performance Analysis Type] dialog box.

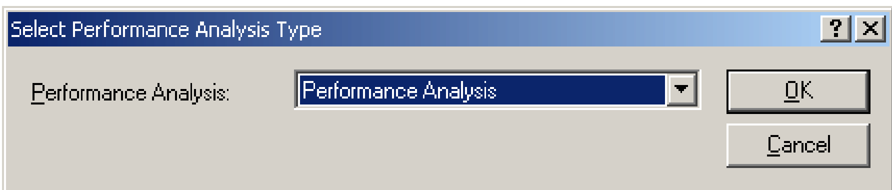


Figure 5.42 [Select Performance Analysis Type] Window

Click the [OK] button to open the [Performance Analysis] window.

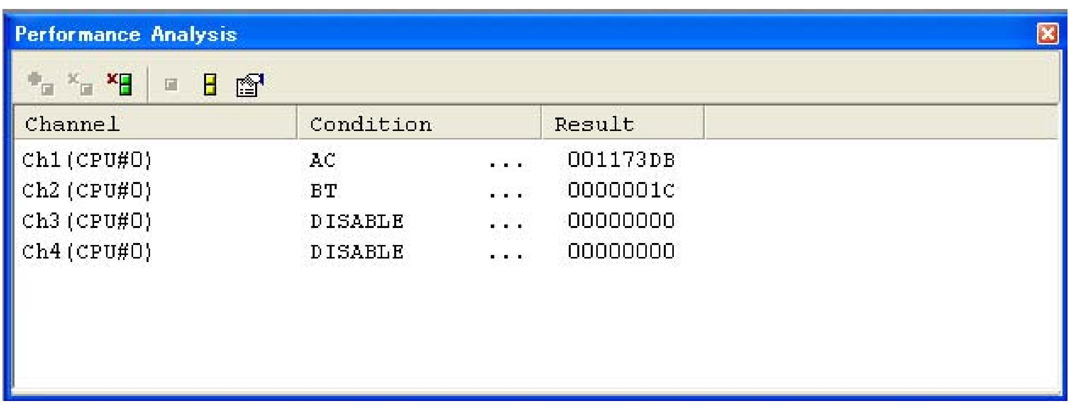


Figure 5.43 [Performance Analysis] Window

It is possible to hide any column not necessary in the [Performance Analysis] window. Selecting a column you want to hide from the popup menu displayed by clicking the right-hand mouse button on the header column hides that column. To display the hidden column, select the column from the said popup menu again.

5.8.2 Setting Conditions for Measurement

Conditions for measurement can be displayed and changed in the [Performance Analysis] window. Select a point where a condition is to be set, and then select [Set…] from the popup menu to display the [Performance Analysis Properties] dialog box.

5.8.3 Starting Performance Data Acquisition

Executing the user program clears the result of previous measurement and automatically starts measuring execution performance according to the conditions that have been set. Stopping the user program displays the result of measurement in the [Performance Analysis] window.

5.8.4 Deleting a Measurement Condition

Select [Reset] from the popup menu with a measurement condition selected to delete the condition.

5.8.5 Deleting All Measurement Conditions

Choose [Reset All] from the popup menu to delete all the conditions that have been set.

Section 6 Tutorial [SH-2A]

6.1 Introduction

A tutorial program is provided to introduce the main functions of the emulator. Operation in [Parallel] mode is described with the aid of this program.

Explanations where something else is not stated apply to operations of the High-performance Embedded Workshop for CPU1. The tutorial program is written in C++ and runs through the High-performance Embedded Workshops for CPU0 and CPU1 to sort ten random data items into ascending or descending order. The tutorial program performs the following actions:

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

The file `tutorial.cpp` contains source code for the tutorial program. The file `Tutorial.abs` is a compiled load module in the Elf/Dwarf2 format.

Notes: 1. Operation of `Tutorial.abs` is big endian. For little-endian operation, `Tutorial.abs` must be recompiled. After recompilation, the addresses may differ from those given in this section.

2. This section describes general usage examples for the emulator. For the specifications of particular products, refer to the additional document, ″Supplementary Information on Using the SHxxxx″, or the online help.

3. The operation address of `Tutorial.abs` attached to each product differs depending on the product. Replace the address used in this section with upper 16 bits of the actually loaded address.
Example: Although the PC address is H’0000006c in the manual, enter H’0C00xxxx when the loaded address of `Tutorial.abs` is H’0C00006c (upper bit H’0000 is changed to H’0C00).

4. The displayed addresses and data may differ from those given in this section depending on the MCU to be used.

6.2 Running the High-performance Embedded Workshop

Selects [Renesas] → [High-performance Embedded Workshop] → [High-performance Embedded Workshop from the [Program] item in the [Start] menu.

6.3 Setting up Synchronized Debugging

1. The [Welcome!] dialog box is displayed.

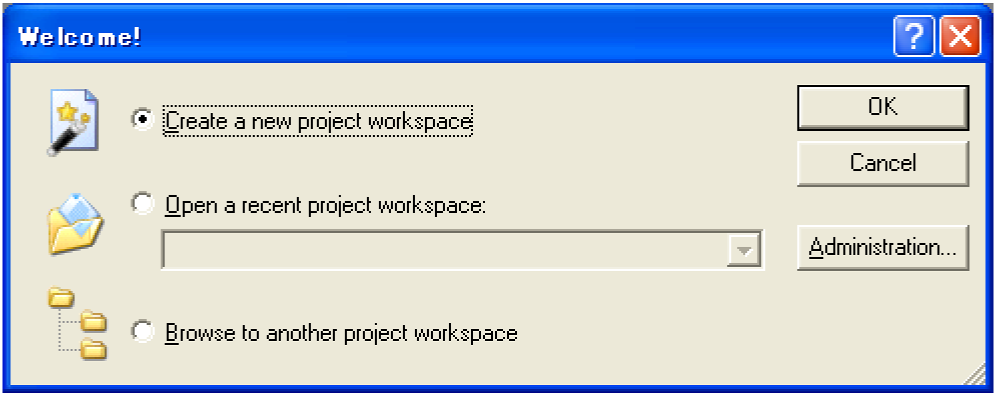


Figure 6.1 [Welcome!] Dialog Box

Click the [Cancel] button here.

2. Select the [Synchronized debug] from the [Debug] menu to open the dialog box shown below.

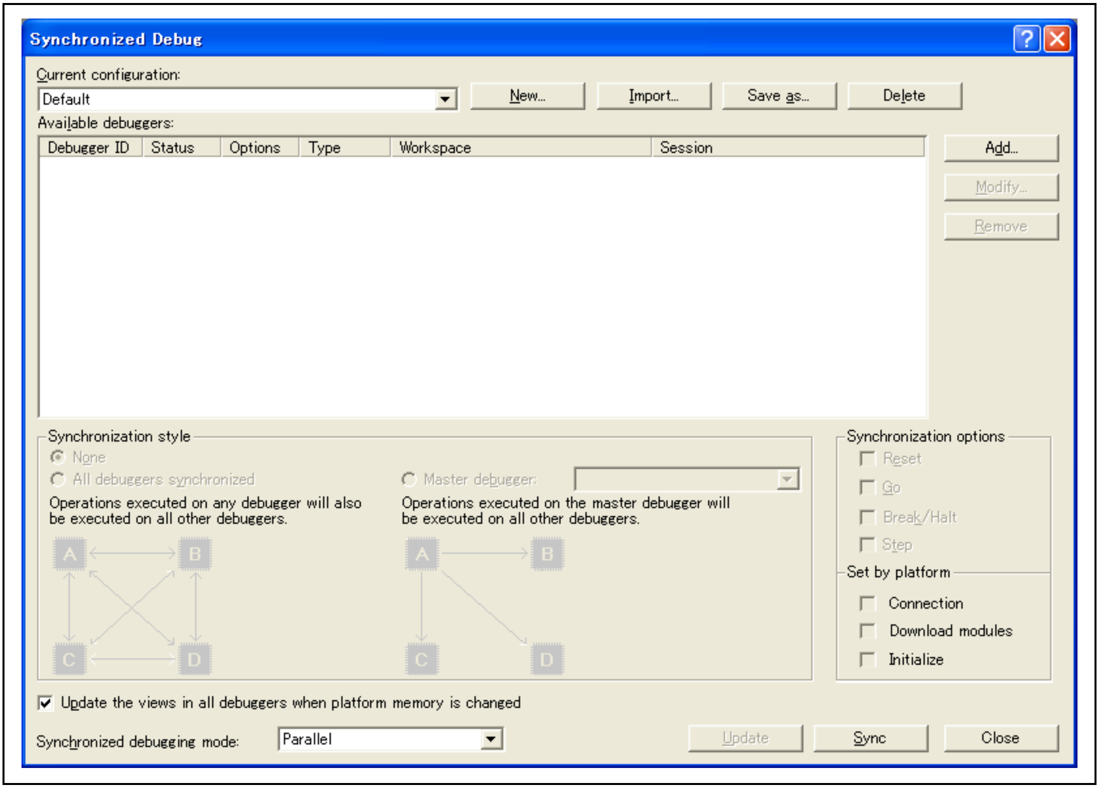


Figure 6.2 [Synchronized Debug] Dialog Box

3. Click on [New…] and enter the [Setting name for the Synchronized Debug]; for this tutorial, we have used SH2A-DUAL_Tutorial.
4. Click on the [Add] button to open the [Add debugger] dialog box.
   Click on the [Browse] button, find
   <Drive where the OS has been installed>:
   \WorkSpace\Tutorial\E10A-USBM\SH2A-DUAL\SH2A-DUAL\Tutorial\xxxx\xxxx.hws
   (**** represents the target device group), and read the file in.
   Click on the [OK] button, select [CPU0] from the [Project] drop-down list box, and close the dialog box.

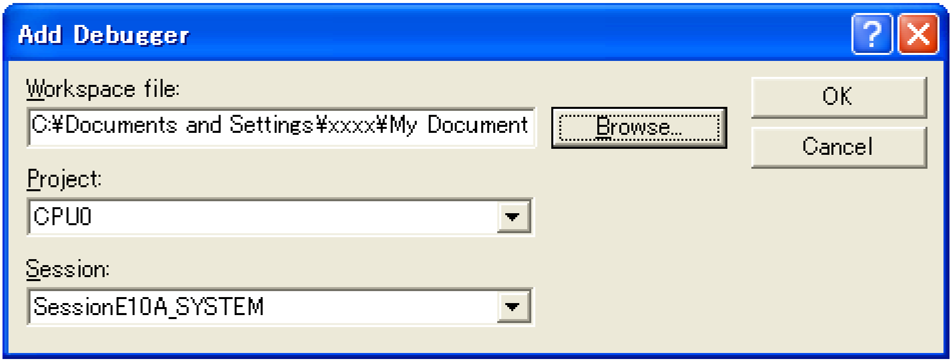


Figure 6.3 [Add Debugger] Dialog Box

Note: In case of failure to read the workspace, open the workspace and then read it, in accord with the procedure described in section 3.9, System Check.

5. Click on the [Add] button again to open the [Add debugger] dialog box.
   Check whether the workspace which was read previously to the [Workspace file] is displayed. If this is not the case, read the workspace again by following the procedure 4, above.
   Selects [CPU1] from the [Project] drop-down list box and then click on the [OK] button.
6. Set up the synchronized debugging state.
   Select [All debuggers synchronized] under [Synchronization style] and check all of the following check boxes under [Synchronization options]: [Go], [Break/Halt], and [Step].
   Select “Parallel” from the [Synchronization debugging mode] drop-down list.

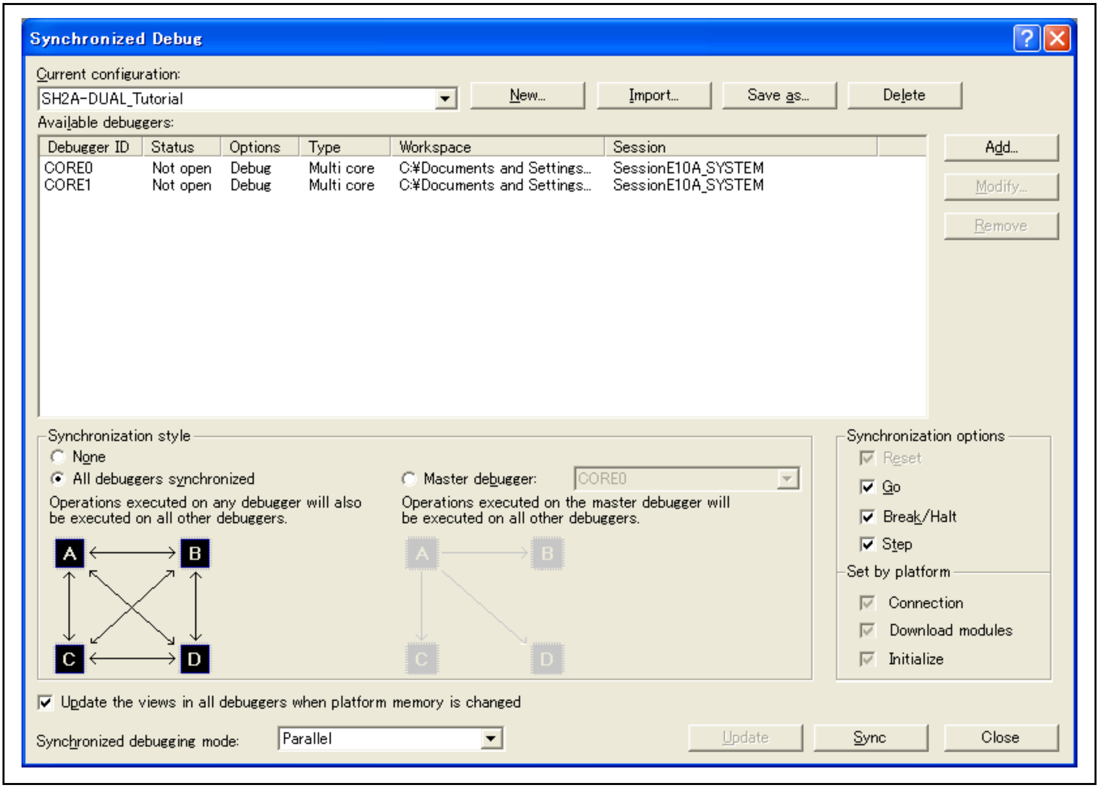


Figure 6.4 [Synchronized Debug] Dialog Box

Click on the [Synchronized Debug] button and start up the High-performance Embedded Workshop according to the procedure in section, 3.9 System Check.

6.4 Setting up the Emulator

The clocks which are used for data communications must be set up on the emulator before the program is downloaded.

- AUD clock

  A clock used in acquiring AUD traces.

  If its frequency is set too low, complete data may not be acquired during realtime tracing.

  Set the frequency not to exceed the upper limit for the MPU's AUD clock.

  The AUD clock is only needed for using emulators that have an AUD trace function.

- JTAG (H-UDI) clock (TCK)

  This is the clock for transfer in cases other than an AUD trace.

  If its frequency is set too low, downloading will be slow.

  Set a frequency that does not exceed the upper limit of the guaranteed TCK range for the supported device.

  For details on the upper limits on the frequency of the AUD Clock (AUDCK) and TCK for all products, refer to section 2.2.3, Notes on the Trace Function (4) AUD Trace, and section 2.2.4, Notes on Using the JTAG (H-UDI) Clock (TCK), in the additional document, ″Supplementary Information on Using the SHxxxx″.

The following is a description of the procedure used to set the clocks.

6.5 Setting the [Configuration] Dialog Box

- Select [Emulator] then [Systems…] from the [Setup] menu in the High-performance Embedded Workshop for CPU1 to set a communication clock. The [Configuration] dialog box is displayed.

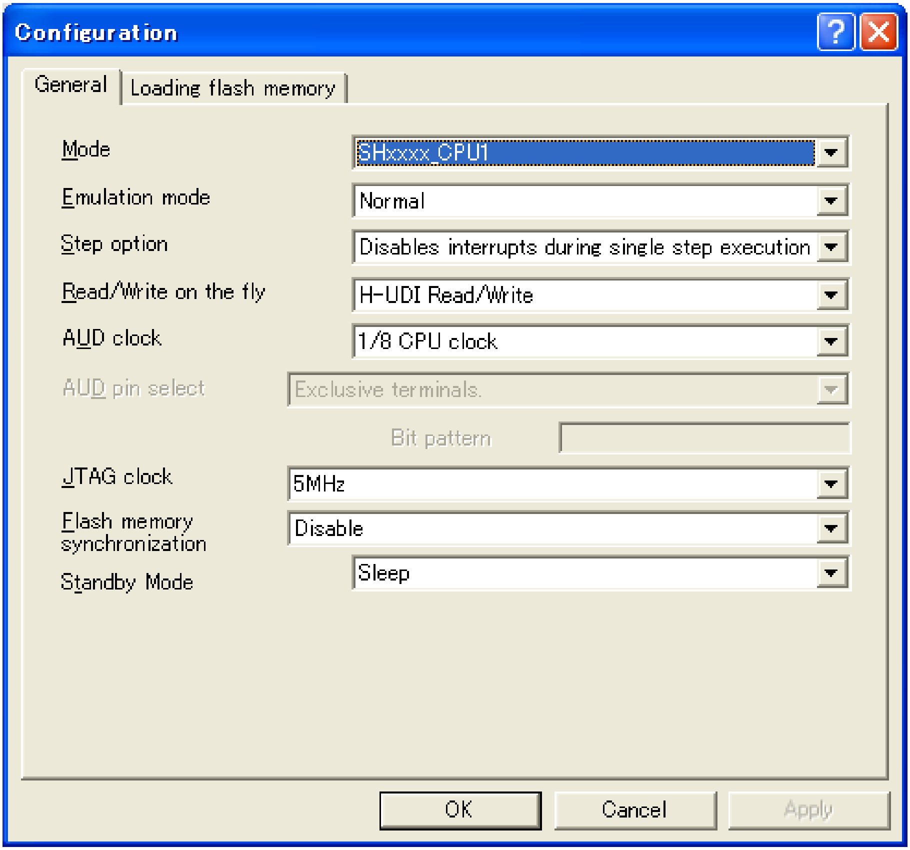


Figure 6.5 [Configuration] Dialog Box

- Set appropriate values in the [AUD clock] and [JTAG clock] combo boxes. The clock also operates with the default value.

Note: The items that can be set in this dialog box differ according to the product. For details on the settings for each product, refer to the online help.

- Click the [OK] button to set a configuration.

6.6 Checking the Operation of the Target Memory for Downloading

Check that the destination memory area for downloading is operating correctly.

When the destination memory is SDRAM or DRAM, a register in the bus controller of the CPU must be set before downloading. Set the bus controller correctly in the [IO] window according to the memory type to be used.

When the required settings, such as the settings for the bus controller, have been completed, display and edit the contents of the destination memory in the [Memory] window in the High-performance Embedded Workshop for CPU1 to check that the memory is operating correctly.

Note: The above way of checking the operation of memory may be inadequate. It is recommended that a program for checking the memory be created.

- Select [Memory…] from the [CPU] submenu of the [View] menu and enter ***H'00000000, H'00000000,*** and ***H'FFFFFFFF*** in the [Display address], [Scroll Start Address], and [Scroll End Address] edit boxes, respectively.

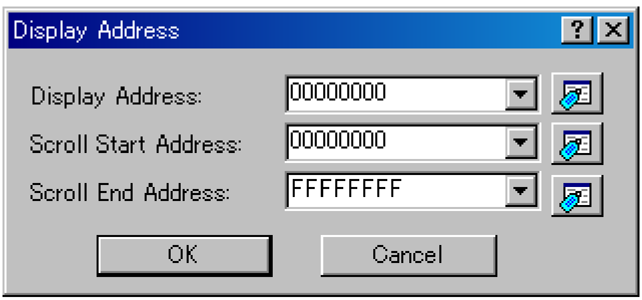


Figure 6.6 [Display Address] Dialog Box

- Click the [OK] button. The [Memory] window is displayed and shows the specified memory area.

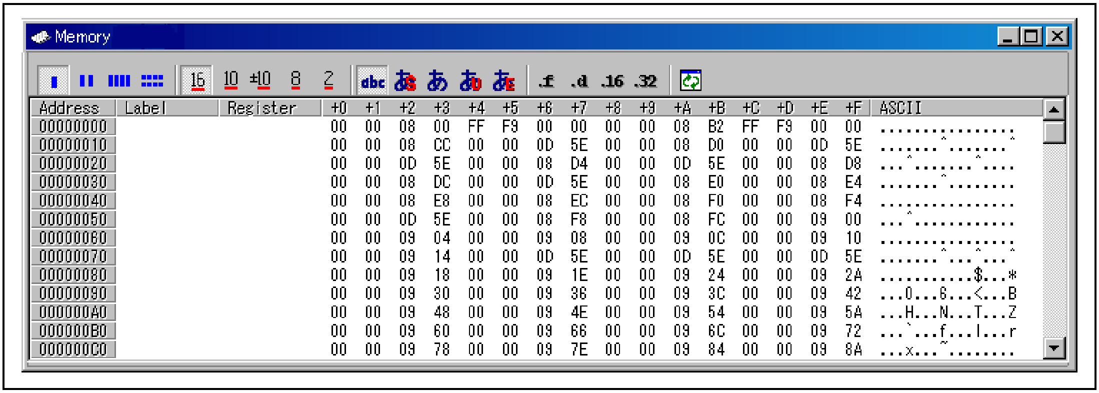


Figure 6.7 [Memory] Window

- Placing the mouse cursor on a point in the display of data in the [Memory] window and double-clicking allows the values at that point to be changed. Data can also be directly edited around the current position of the text cursor.

6.7 Downloading the Tutorial Program

6.7.1 Downloading the Tutorial Program

Download the object program to be debugged.

To proceed with source-level debugging with the High-performance Embedded Workshop for CPU0 or the High-performance Embedded Workshop for CPU1, download the debugging information file for the corresponding CPU.

- Select [Download module] from [Tutorial.abs] under [Download modules].

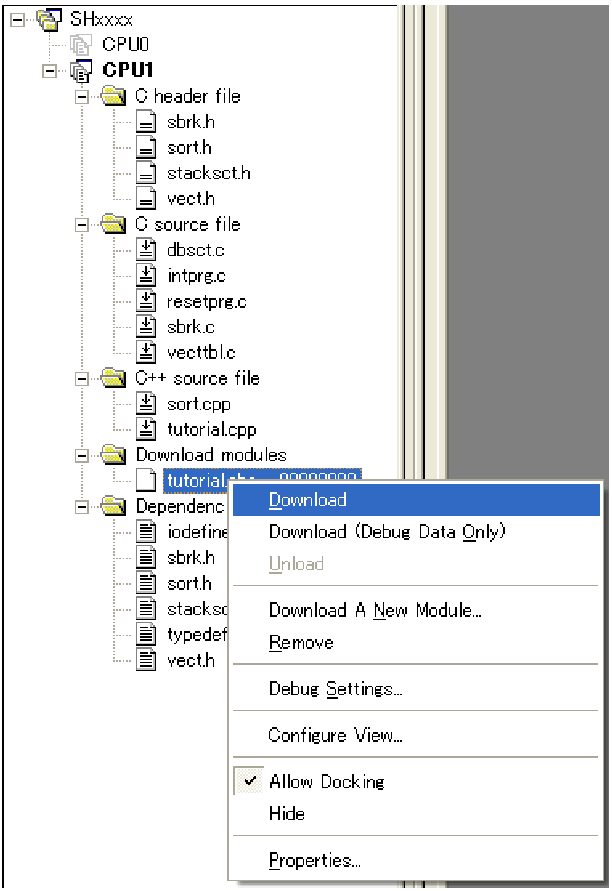


Figure 6.8 Downloading the Tutorial Program

6.7.2 Displaying the Source Program

The High-performance Embedded Workshop allows the user to debug a user program at the source level.

- Double-click [tutorial.cpp] under [C++ source file] in the High-performance Embedded Workshop for CPU1.

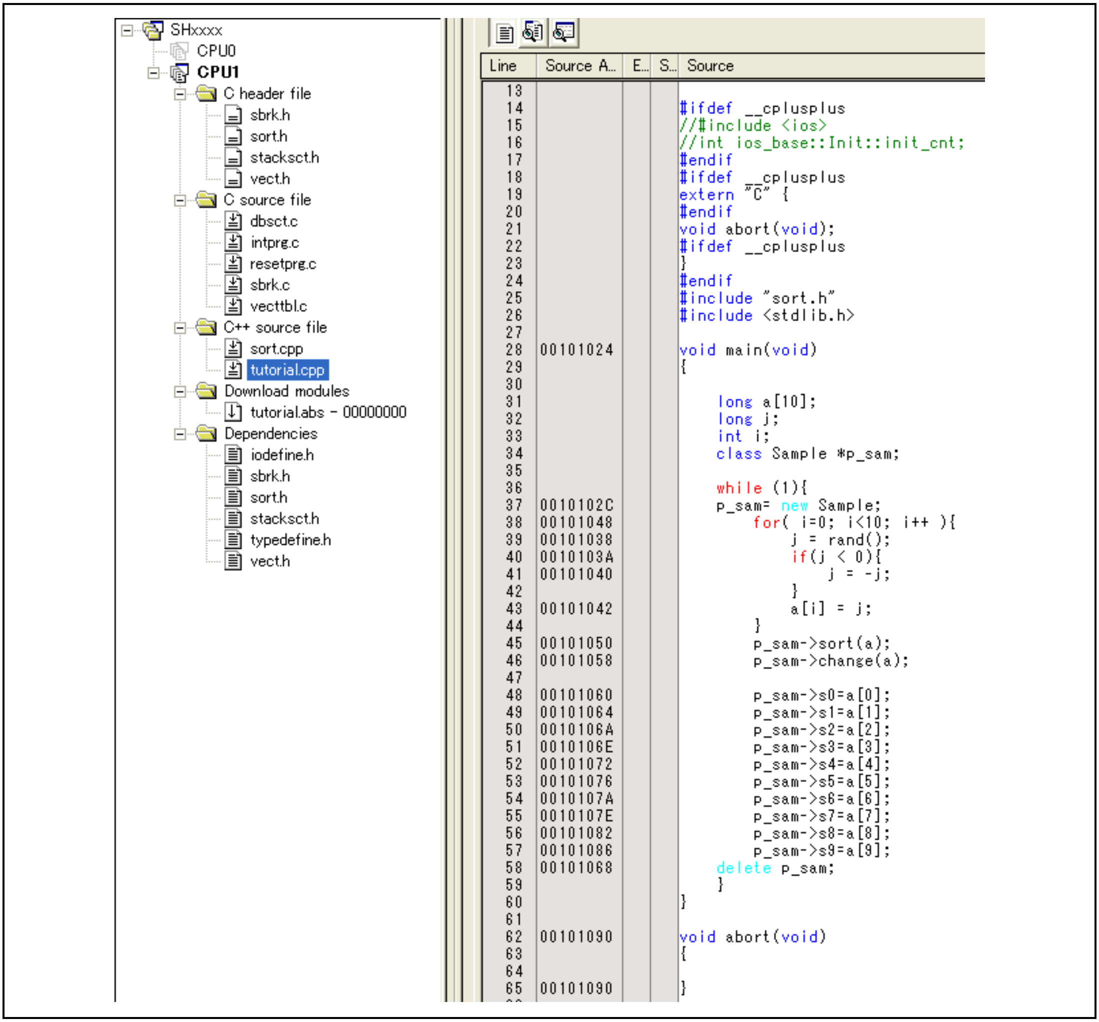


Figure 6.9 [Editor] Window (Displaying the Source Program)

Select a font and size that are legible from the [Format…] option in the [Setup] menu if necessary.

Initially the [Editor] window shows the start of the user program, but the user can use the scroll bar to scroll through the user program and look at the other statements.

6.8 Setting a PC Breakpoint

A PC breakpoint is a simple debugging function.

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

- Select by double-clicking the [S/W breakpoint] column on the line containing the `sort` function call in the High-performance Embedded Workshop for CPU1.

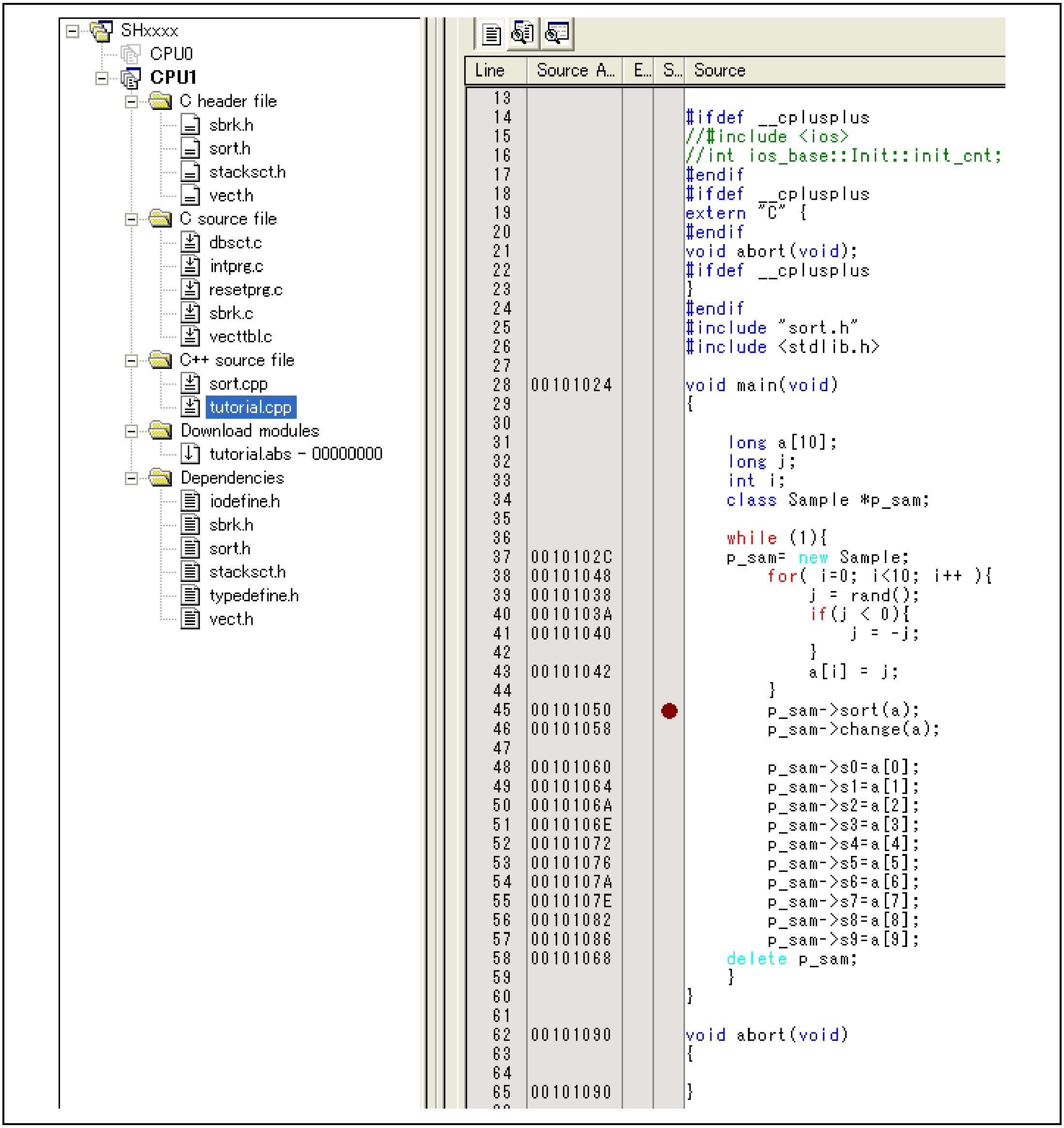


Figure 6.10 [Editor] Window (Setting a PC Breakpoint)

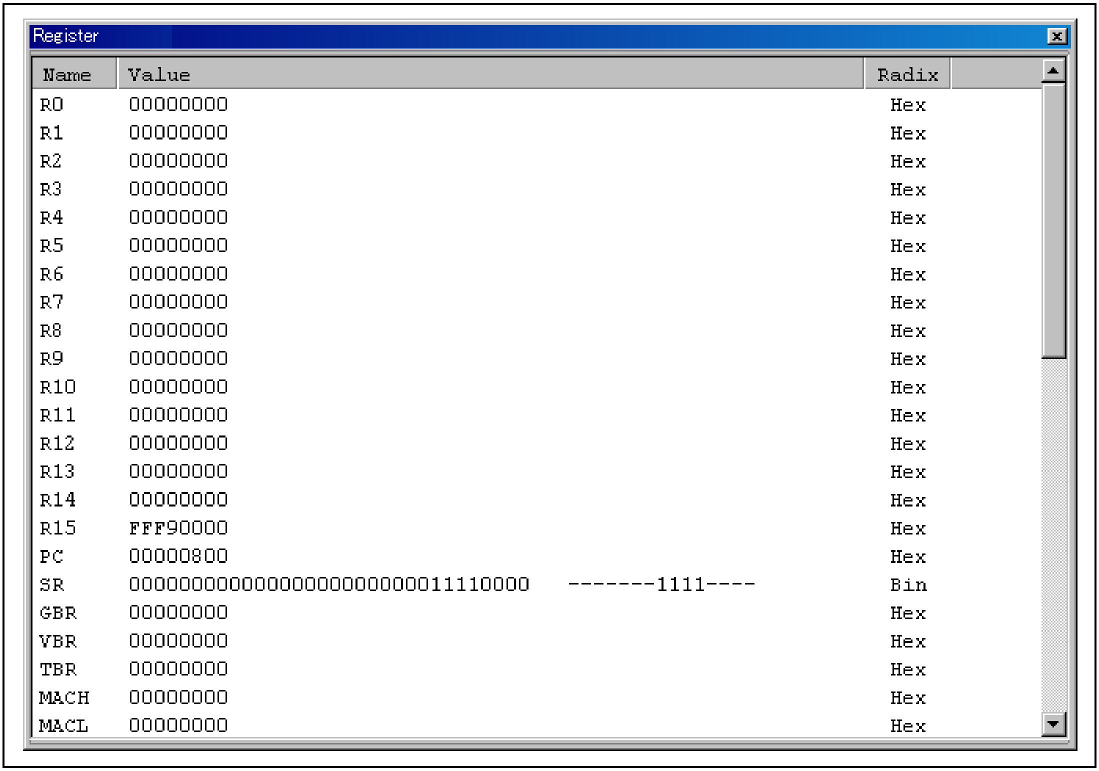
The symbol • will appear on the line containing the `sort` function. This shows that a PC breakpoint has been set.

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

6.9 Setting Registers

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

- Select [Registers] from the [CPU] submenu of the [View] menu in the High-performance Embedded Workshop for CPU0 and CPU1. The [Register] window is displayed.


Register

Name	Value	Radix
R0	00000000	Hex
R1	00000000	Hex
R2	00000000	Hex
R3	00000000	Hex
R4	00000000	Hex
R5	00000000	Hex
R6	00000000	Hex
R7	00000000	Hex
R8	00000000	Hex
R9	00000000	Hex
R10	00000000	Hex
R11	00000000	Hex
R12	00000000	Hex
R13	00000000	Hex
R14	00000000	Hex
R15	FFF90000	Hex
PC	00000800	Hex
SR	00000000000000000000000011110000 --------1111----	Bin
GBR	00000000	Hex
VBR	00000000	Hex
TBR	00000000	Hex
MACH	00000000	Hex
MACL	00000000	Hex

Figure 6.11 [Register] Window

- To change the value of the program counter (PC), double-click the value area in the [Register] window with the mouse. The following dialog box is then displayed, and the value can be changed. Set the program counter to H'00000800 in this tutorial program, and click the [OK] button.

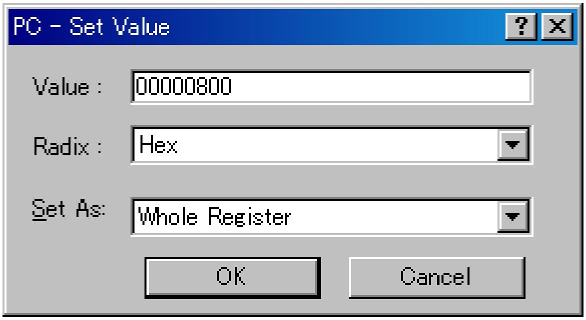


Figure 6.12 [Register] Dialog Box (PC)

- Change the value of the stack pointer (SP) in the same way. Set H'FFF90000 for the value of the stack pointer in this tutorial program.
  When using the MCU with flash memory, specify the end address of the internal RAM for the stack pointer (SP). The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.

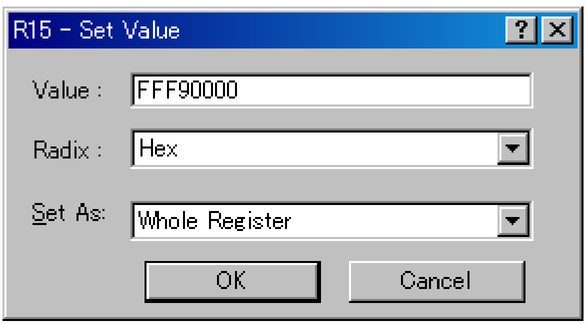


Figure 6.13 [Register] Dialog Box (R15)

6.10 Executing the Program

Execute the program as described in the following:

- Since the [Go] check box under [Synchronization options] in the [Synchronized debug] dialog box is selected, execute the program by selecting [Go] from the [Debug] menu of the High-performance Embedded Workshop for either CPU0 or CPU1, and then selecting [Go] from the [Debug] menu or selecting the [Go] button on the toolbar of the High-performance Embedded Workshop for CPU1.

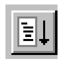


Figure 6.14 [Go] Button

Once program execution has started, `'**RUNNING'` will be displayed on the status bar. The program will be executed up to any breakpoint that has been set in the High-performance Embedded Workshop for CPU1. Since synchronized breaking is enabled ([All debuggers synchronized], [Break/Program Halt] check box under [Synchronization options] selected), the High-performance Embedded Workshops for CPU0 and CPU1 will break at the same time.

An arrow will be displayed in the [S/W breakpoint] column to indicate the position where the program was suspended, and the message `[BREAKPOINT]` in the status bar.

Notes: 1. When the source file is displayed after a break, a path of the source file may be inquired. The location of the source file is as follows:
Display of the source file after a break may necessitate an enquiry regarding the path of the source file. The location of the source file is as follows:

<Drive where the OS has been installed>:

\WorkSpace\Tutorial\E10A-USBM\SH2A-DUAL\SH2A-DUAL\Tutorial\CPU1\source.

2. If program execution is failed, select [Reset CPU] from the [Debug] menu, reset the device, and restart the procedure from figure 6.8.

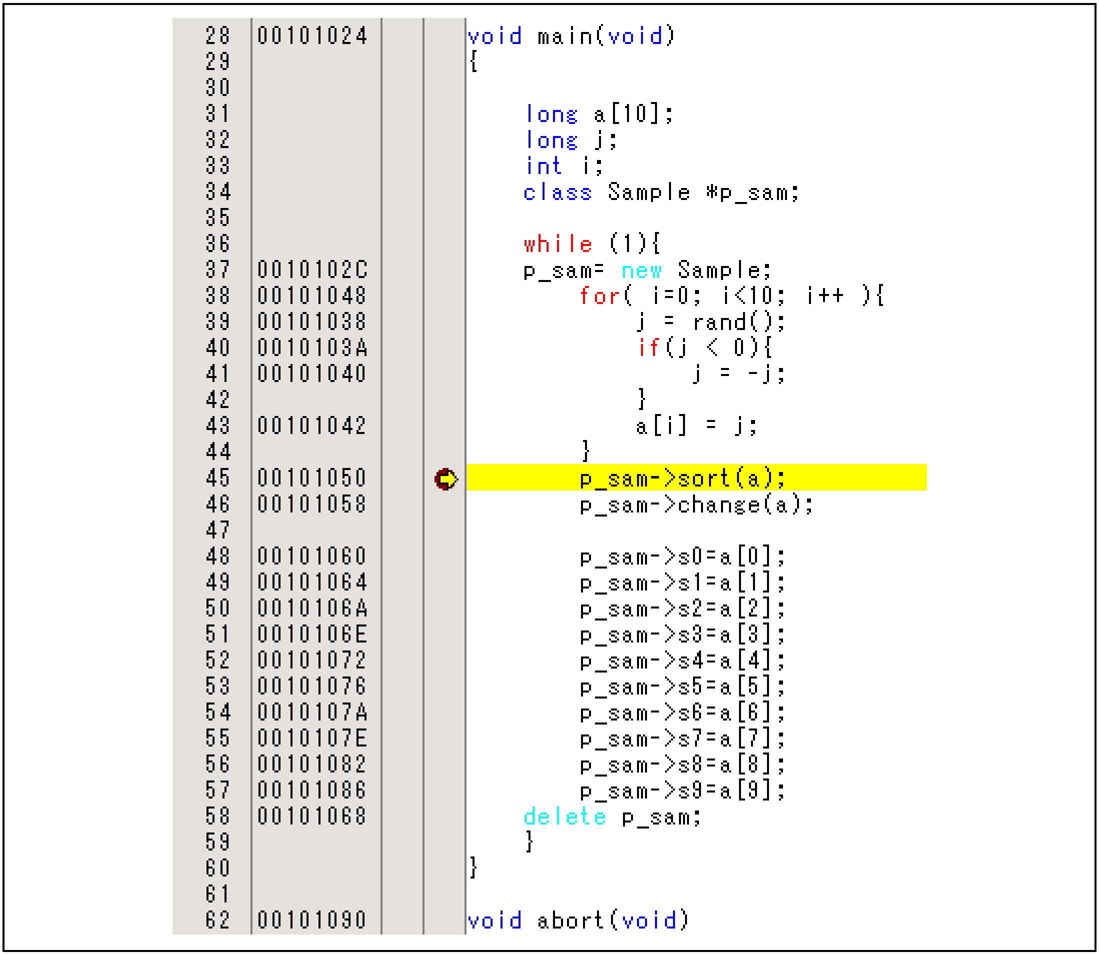


Figure 6.15 [Editor] Window (Break State)

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

- Select [Status] from the [CPU] submenu of the [View] menu. After the [Status] window is displayed, open the [Platform] sheet, and check the `Status` of `Cause of last break`.

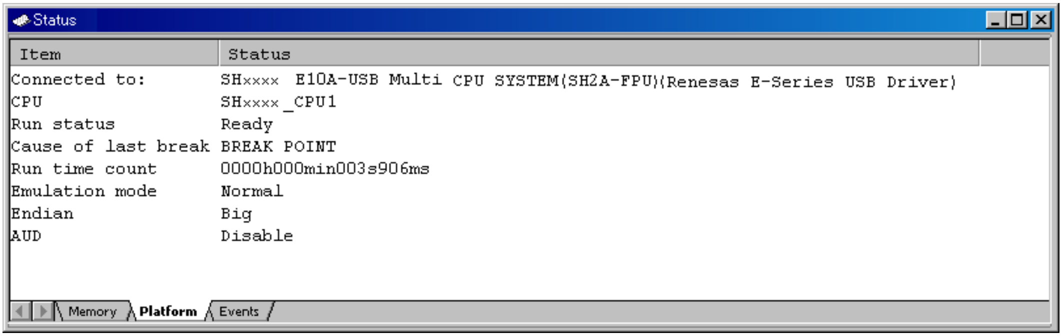


Figure 6.16 [Status] Window

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

6.11 Reviewing Breakpoints

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

- Select [Eventpoints] from the [Code] submenu of the [View] menu of the High-performance Embedded Workshop for CPU1. The [Event] window is displayed. Select the [Breakpoint] sheet.

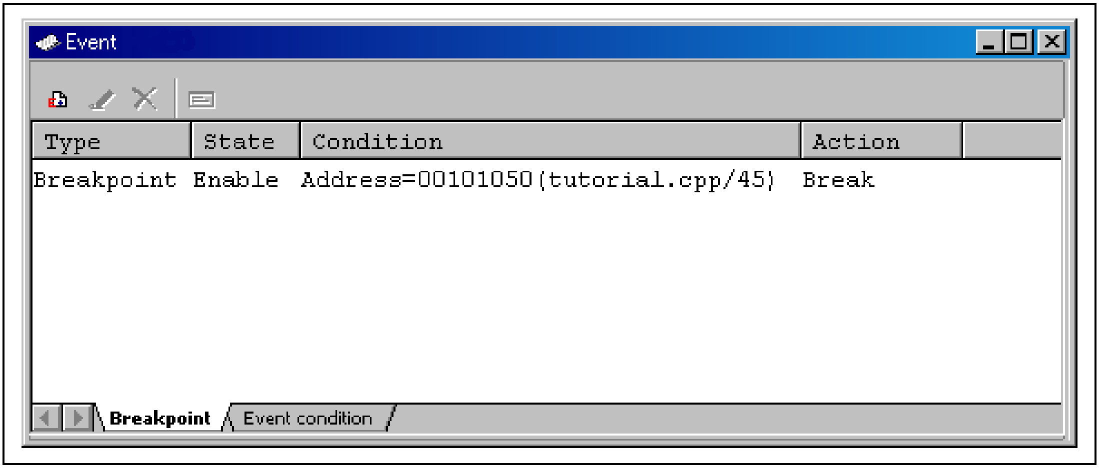


Figure 6.17 [Event] Window

The popup menu, opened by clicking the [Event] window with the right-hand mouse button, allows the user to set or change breakpoints, define new breakpoints, and delete, enable, or disable breakpoints.

6.12 Referring to Symbols

The [Label] window can be used to display the information on symbols in modules.

Select [Label] from the [Symbol] submenu of the [View] menu of the High-performance Embedded Workshop for CPU1. The [Label] window is displayed so that the user can refer to the addresses of symbols in modules.

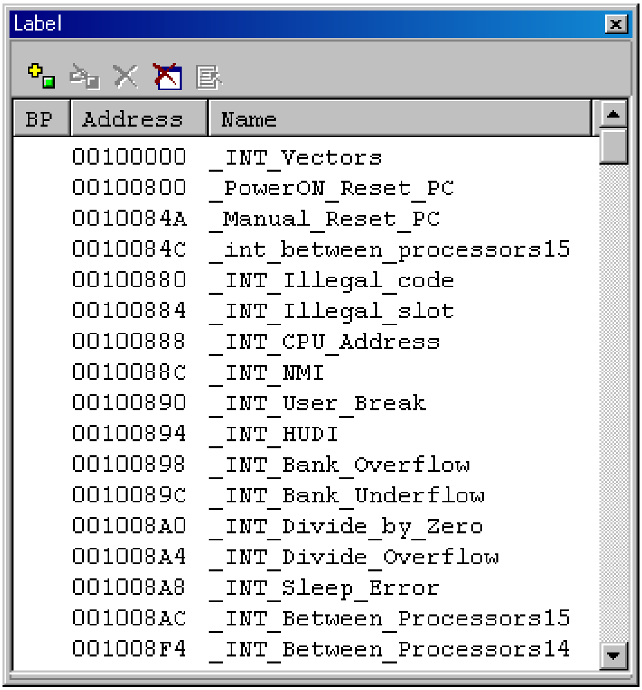


Figure 6.18 [Label] Window

6.13 Viewing Memory

When the label name is specified, the user can view the memory contents that the label has been registered in the [Memory] window. For example, to view the memory contents corresponding to `_main` in word size:

- Select [Memory …] from the [CPU] submenu of the [View] menu in the High-performance Embedded Workshop for CPU1, enter ***`_main`*** in the [Display Address] edit box, ***`00000000`*** in the [Scroll Start Address] edit box, and ***`FFFFFFFF`*** in the [Scroll End Address] edit box.

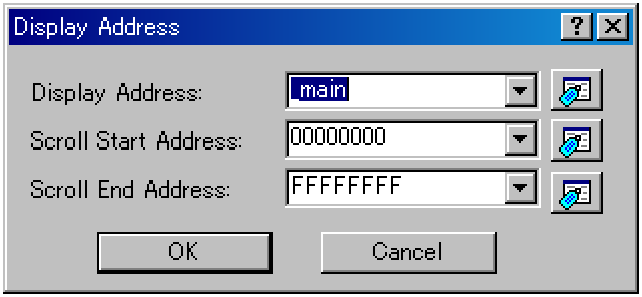


Figure 6.19 [Display Address] Dialog Box

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

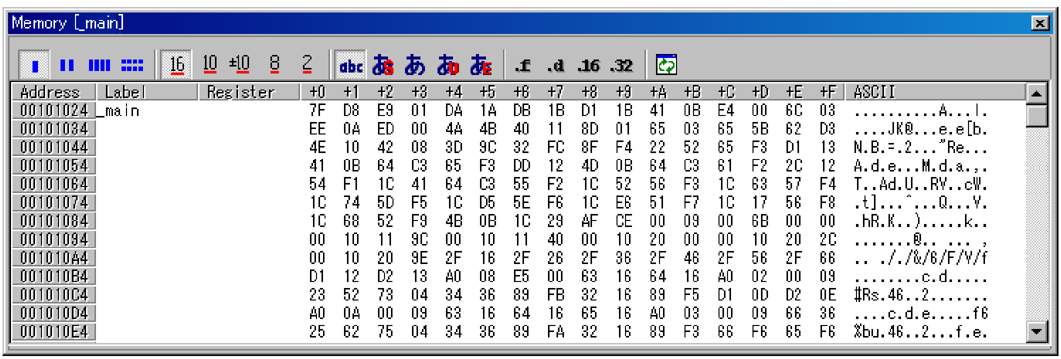


Figure 6.20 [Memory] Window

6.14 Watching Variables

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

- Place the cursor in the column to the left of where array a is displayed in the [Editor] window of the High-performance Embedded Workshop for CPU1.
- Click the right-hand mouse button and select [Instant Watch...].

The following dialog box will be displayed.

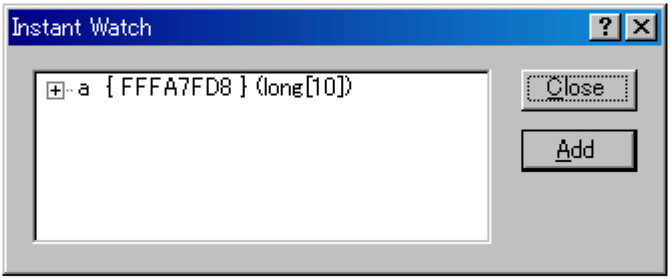


Figure 6.21 [Instant Watch] Dialog Box

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

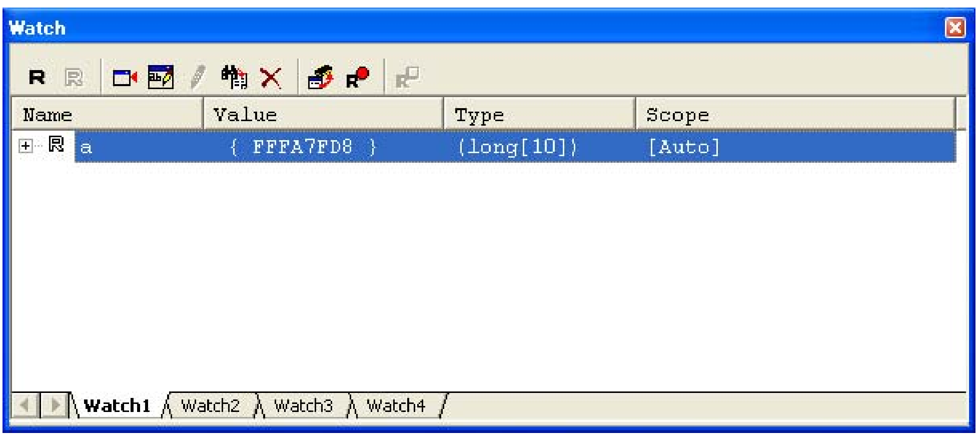


Figure 6.22 [Watch] Window (Displaying the Array)

The user can also add variables to the [Watch] window by specifying those name.

- Click the [Watch] window with the right-hand mouse button and select [Add Watch…] from the popup menu.

The following dialog box will be displayed. Enter variable p_sam.

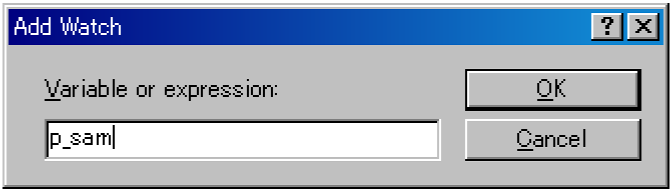


Figure 6.23 [Add Watch] Dialog Box

- Click the [OK] button.

The [Watch] window will now also show the instance p_sam.

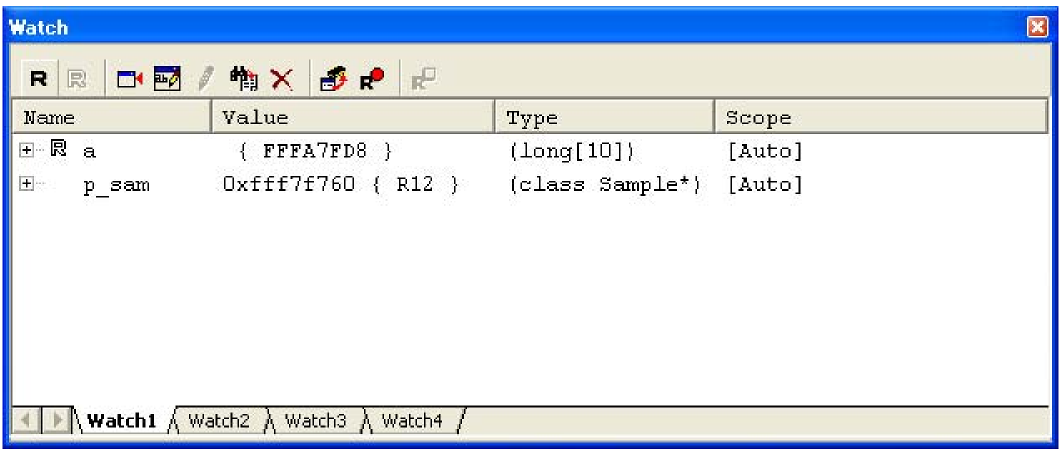


Figure 6.24 [Watch] Window (Displaying the Variables)

The user can click mark '+' at the left side of array a in the [Watch] window to watch all the elements.

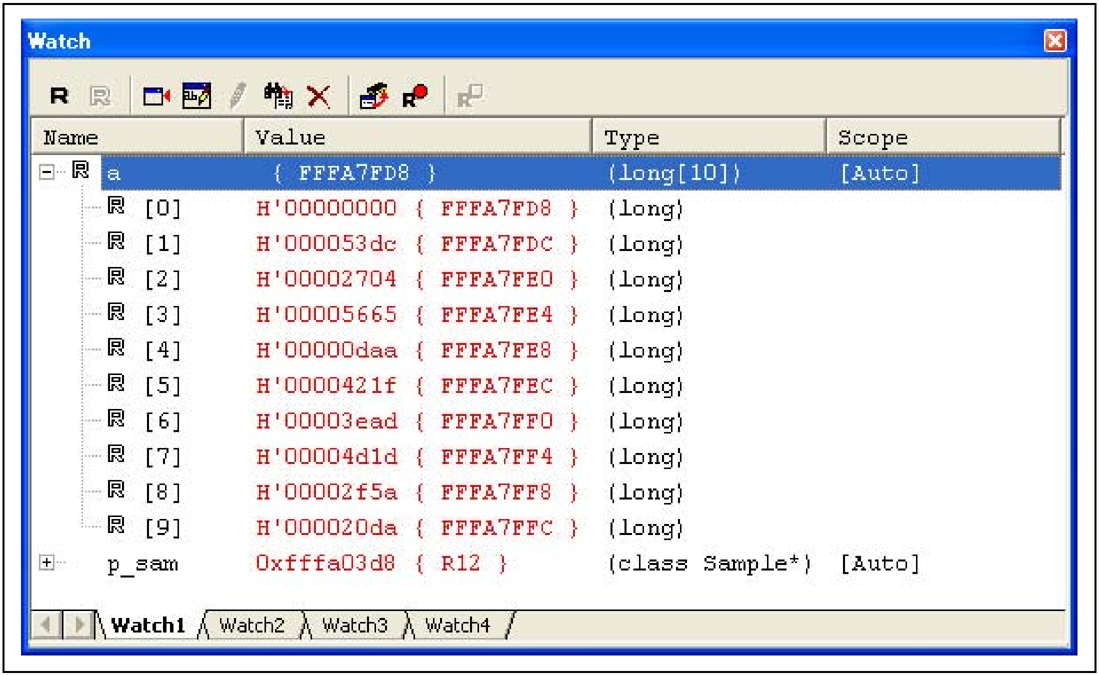


Figure 6.25 [Watch] Window (Displaying Array Elements)

6.15 Displaying Local Variables

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

- Select [Locals] from the [Symbol] submenu of the [View] menu of the High-performance Embedded Workshop for CPU1. The [Locals] window is displayed.

The [Locals] window shows the local variables in the function currently pointed to by the program counter, along with their values. Note, however, that the [Locals] window is initially empty because local variables are yet to be declared.

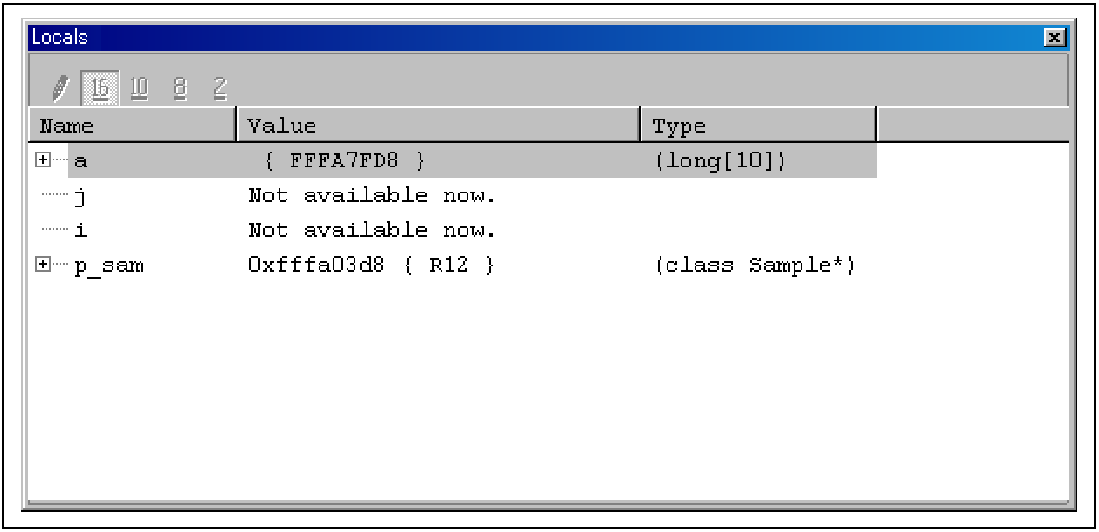


Figure 6.26 [Locals] Window

- Click mark '+' at the left side of array `a` in the [Locals] window to display the elements.
- Refer to the elements of array `a` before and after the execution of the `sort` function, and confirm that random data is sorted in descending order.

6.16 Stepping Through a Program

The High-performance Embedded Workshop provides a range of step menu commands that allow efficient program debugging.

Table 6.1 Step Option

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

6.16.1 Executing [Step In] Command

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

- To step through the `sort` function, select [Step In] from the [Debug] menu in the High-performance Embedded Workshop for CPU1, or click the [Step In] button on the toolbar. Since the synchronized stepping is enabled ([All debuggers synchronized], [Step] check box under [Synchronization options]), this operation will lead to synchronized stepping in.

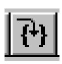


Figure 6.27 [Step In] Button

Figure 6.28 [Editor] Window (Step In)

- The highlighted line moves to the first statement of the `sort` function in the [Editor] window in the High-performance Embedded Workshop for CPU1.

6.16.2 Executing [Step Out] Command

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

To step out of the `sort` function, select [Step Out] from the [Debug] menu in the High-performance Embedded Workshop for CPU1, or click the [Step Out] button on the toolbar.

Note: It takes time to execute this function. When the calling source is clarified, use [Go To Cursor].

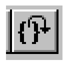


Figure 6.29 [Step Out] Button

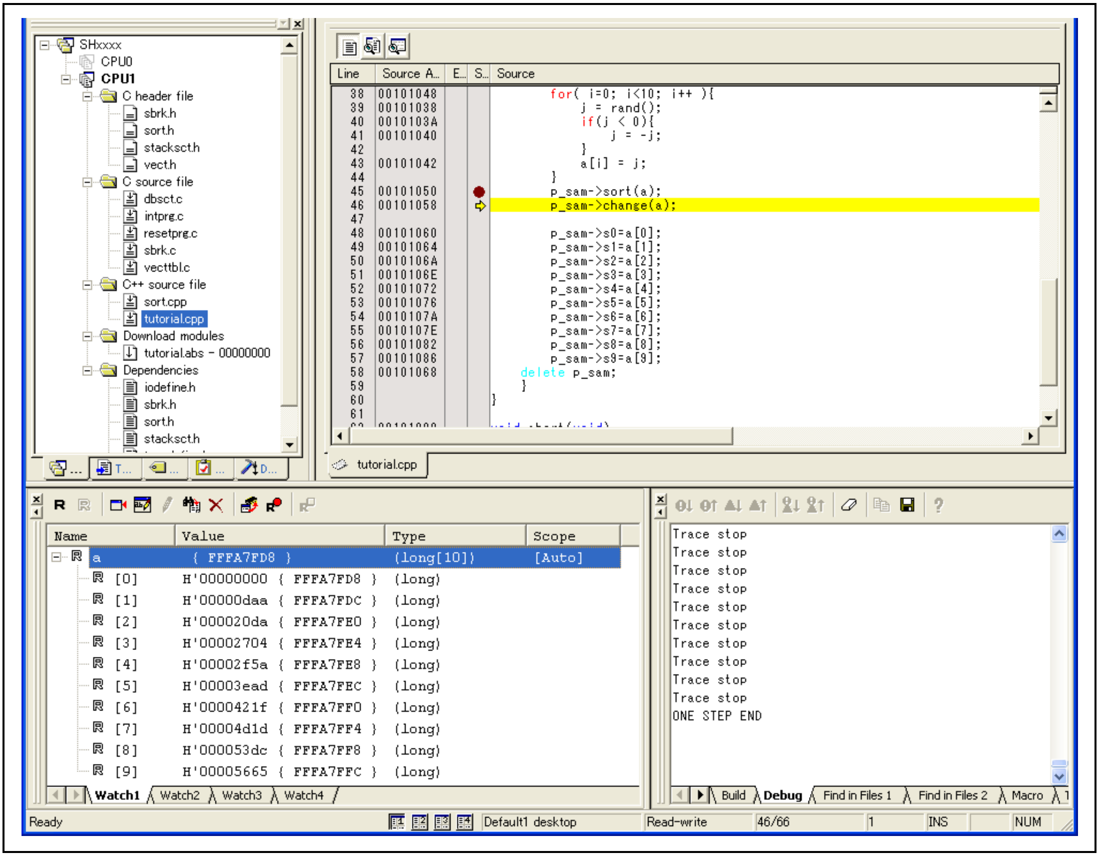


Figure 6.30 [High-performance Embedded Workshop] Window (Step Out)

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

According to the position in the source code at the start of synchronization of the session for CPU0, stepping out on the CPU0 side may not be completed. In such cases, complete stepping out by selecting the [STOP] button on the toolbar.

6.16.3 Executing [Step Over] Command

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

- Move to the `change` function following the procedures described in section 6.16.2, Executing [Step Out] Command.
- To step through all statements in the `change` function at a single step, select [Step Over] from the [Debug] menu of the High-performance Embedded Workshop for CPU1, or click the [Step Over] button on the toolbar.

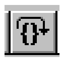


Figure 6.31 [Step Over] Button

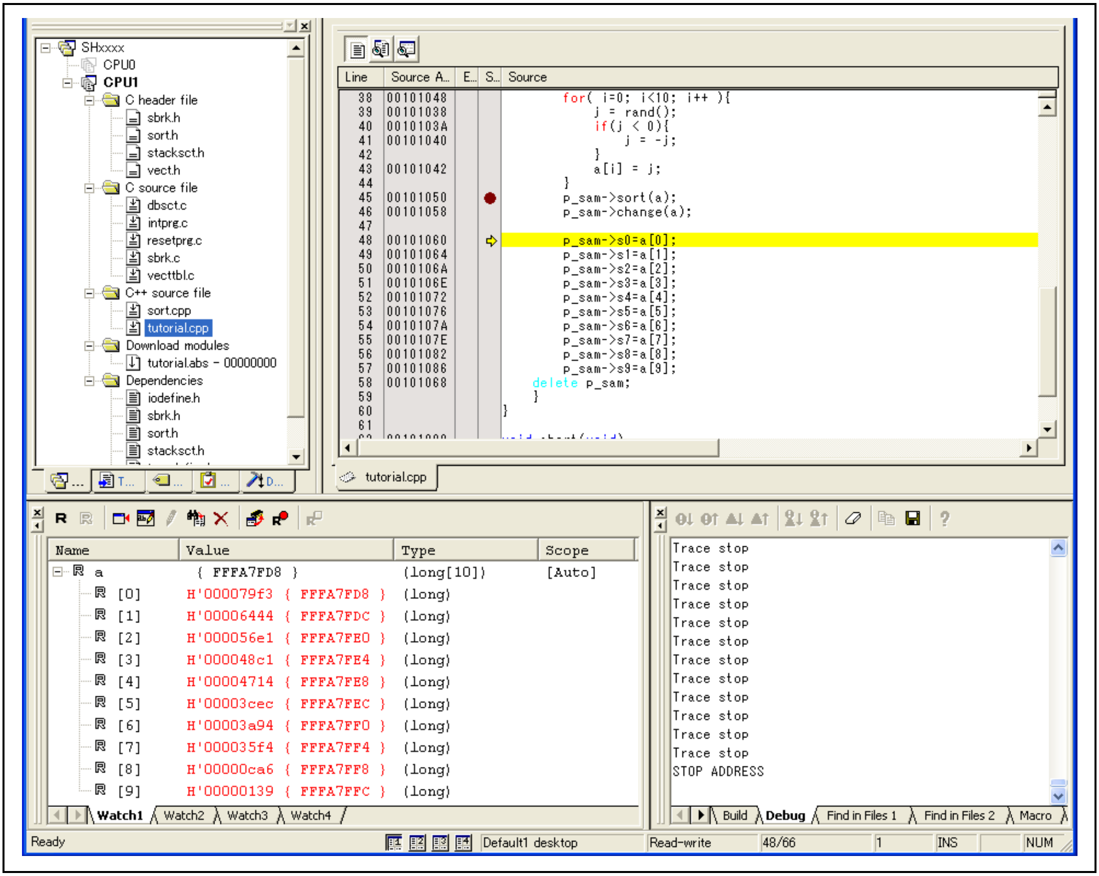


Figure 6.32 [High-performance Embedded Workshop] Window (Step Over)

6.17 Forced Breaking of Program Executions

The High-performance Embedded Workshop can force a break in the execution of a program.

- Cancel all breaks.
- To execute the remaining sections of the `main` function, select [Go] from the [Debug] menu in the High-performance Embedded Workshop for CPU1, or the [Go] button on the toolbar.

Figure 6.33 [Go] Button

- The program goes into an endless loop. To force a break in execution, select [Halt] from the [Debug] menu of the High-performance Embedded Workshop for CPU1, or click on the [STOP] button on the toolbar. Since synchronized stepping is enabled ([All debuggers synchronized], [Step] check box under [Synchronization options]), the High-performance Embedded Workshops for CPU0 and CPU1 will break at the same time.

Figure 6.34 [STOP] Button

6.18 Break Function

The emulator has PC and hardware break functions. With the High-performance Embedded Workshop, a PC breakpoint can be set using the [Breakpoint] sheet of the [Event] window, and a hardware break condition can be set using the [Event condition] sheet.

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

6.18.1 PC Break Function

The emulator can set up to 255 PC breakpoints. Other methods for setting a PC breakpoint than in section 6.8, Setting a PC Breakpoint, are described below.

- Select [Eventpoints] from the [Code] submenu of the [View] menu in the High-performance Embedded Workshop for CPU1. The [Event] window is displayed.
- Select the [Breakpoint] sheet.

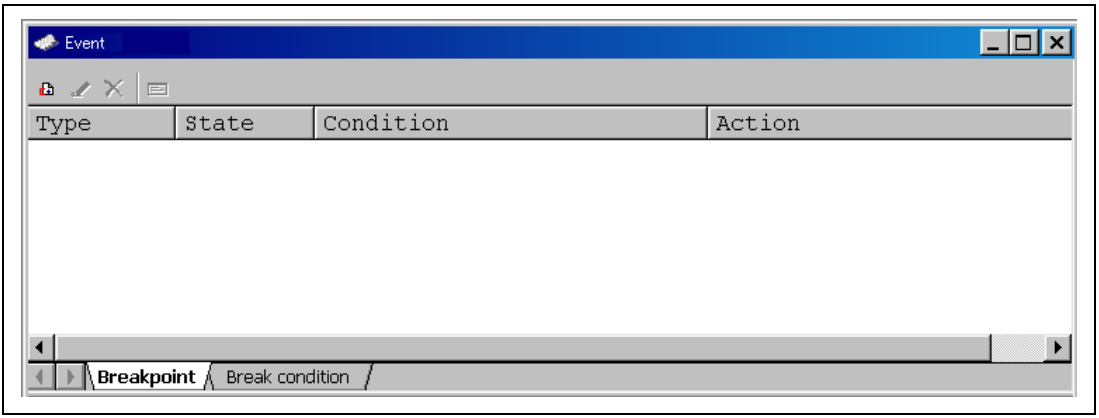


Figure 6.35 [Event] Window (Before PC Breakpoint Setting)

- Click the [Event] window with the right-hand mouse button and select [Add…] from the popup menu.
- Enter ***H'00101060*** in the [Address] edit box.

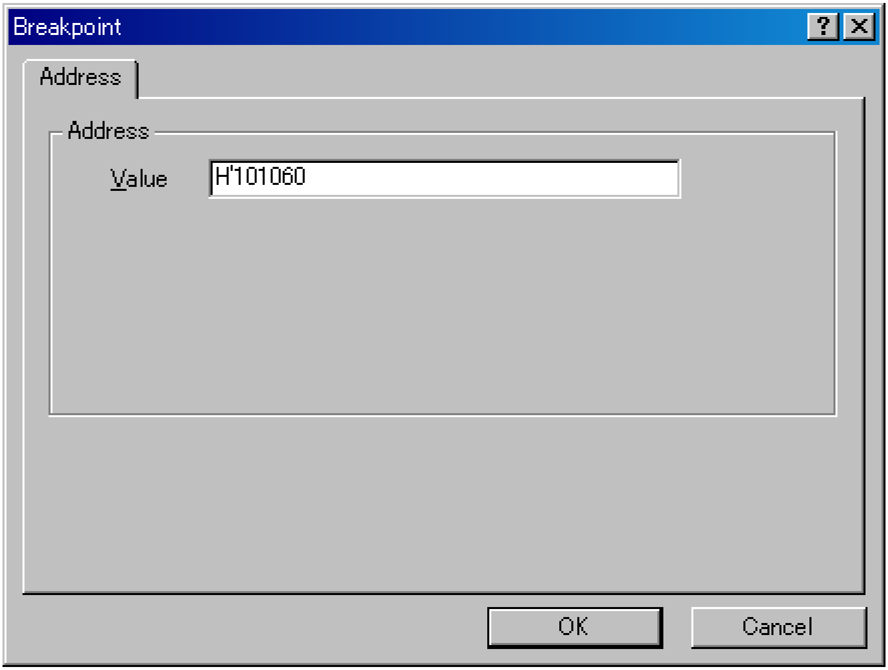


Figure 6.36 [Breakpoint] Dialog Box

Note: This dialog box differs according to the product. For the items of each product, refer to the online help.

- Click the [OK] button.

The PC breakpoint that has been set is displayed in the [Event] window.

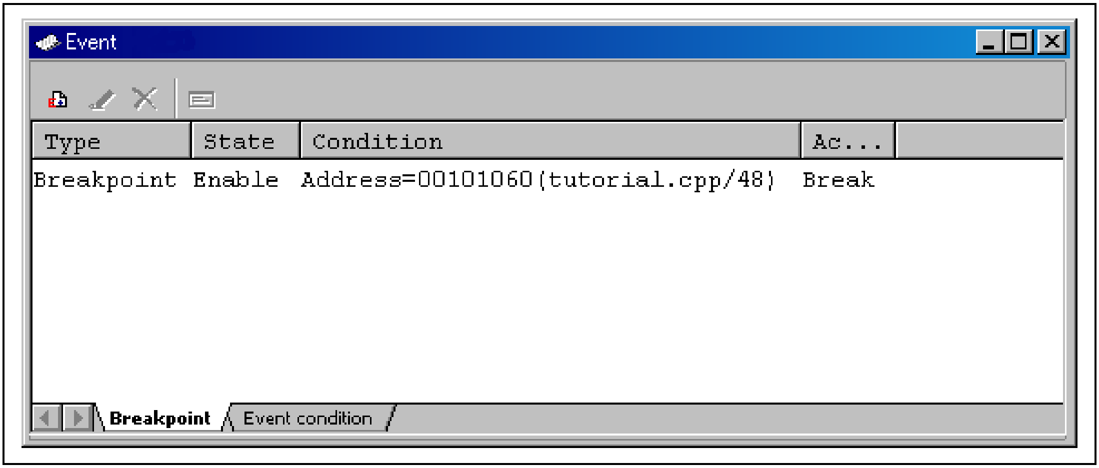


Figure 6.37 [Event] Window (PC Breakpoint Setting)

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

To stop the tutorial program at the PC breakpoint, the following procedure must be executed:

- Set the program counter and stack pointer values (PC = H'00000800 and R15 = H'FFF90000) that were set in section 6.8, Setting Registers, in the [Register] window of the High-performance Embedded Workshops for CPU0 and CPU1. Click the [Go] button in the High-performance Embedded Workshops for either CPU0 or CPU1.

- If program execution is failed, reset the device and execute again the procedures above.

The program runs, and stops at the set PC breakpoint.

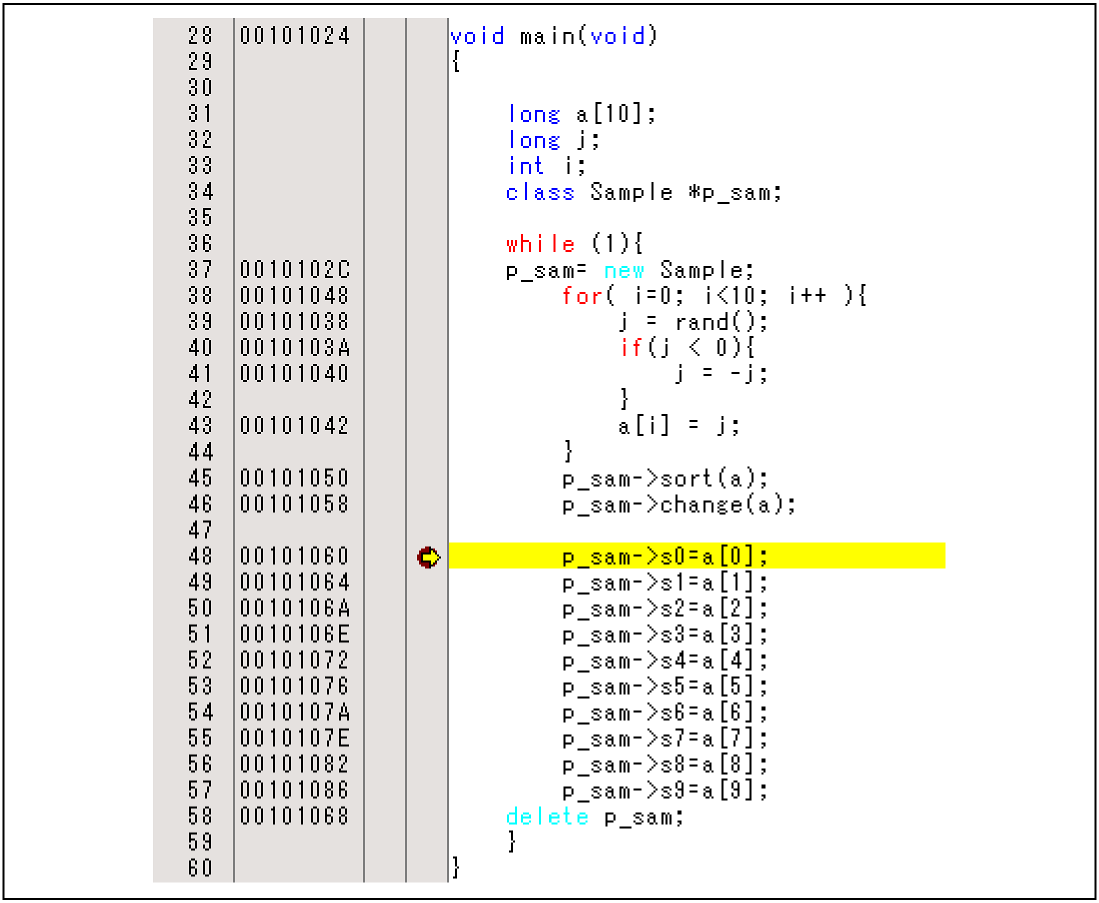


Figure 6.38 [Editor] Window at Execution Stop (PC Break)

The [Status] window displays the following contents.

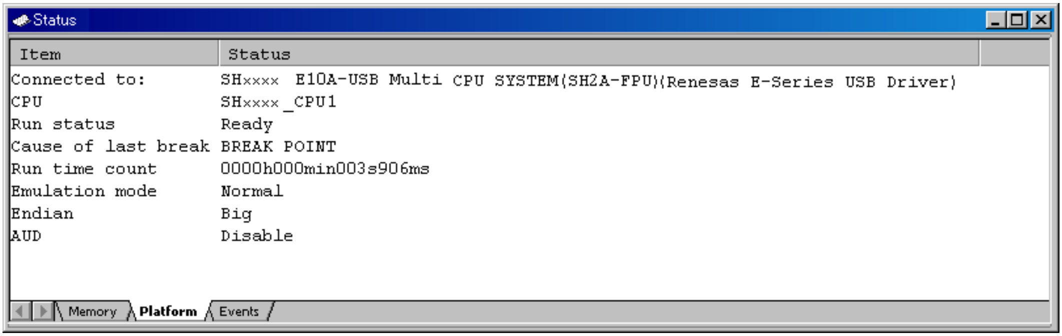


Figure 6.39 Displayed Contents of the [Status] Window (PC Break)

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

6.19 Hardware Break Function

A method is given below in which the address bus condition is set under Ch1 (IA_OA_DT_CT) as hardware break conditions.

- Select [Eventpoints] from the [Code] submenu of the [View] menu of the High-performance Embedded Workshop for CPU1. The [Event] window is displayed.
- The PC breakpoint that has been previously set is deleted. Click the [Event] window with the right-hand mouse button and select [Delete All] from the popup menu to cancel all PC breakpoints that have been set.
- To set a Ch1 (IA_OA_DT_CT), click the [Event condition] tab.

Up to eleven breakpoints can be set independently for the hardware break condition, in the Event conditions 1 to 11. In this example, set the hardware break condition for Ch1 (IA_OA_DT_CT).

Note: The number of hardware break conditions differs according to the product. For the number that can be specified for each product, refer to the online help.

- Select a line of Ch1 (IA_OA_DT_CT) in the [Event] window. When highlighted, double-click this line.

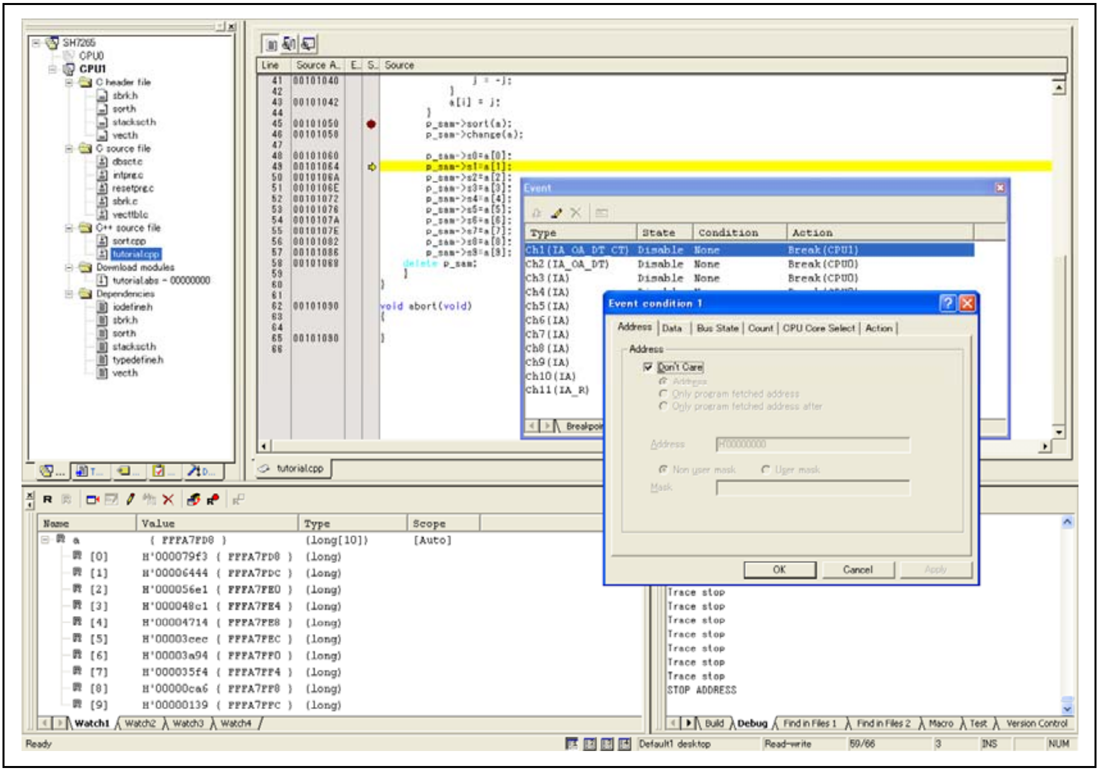


Figure 6.40 [High-performance Embedded Workshop] Window ([Ch1 (IA_OA_DT_CT])

- The [Event condition1] dialog box is displayed.
- Clear the [Don't care] check box in the [Address] page.
- Select the [Only program fetched address] radio button and enter ***H'00101050*** as the value in the [Address] edit box.

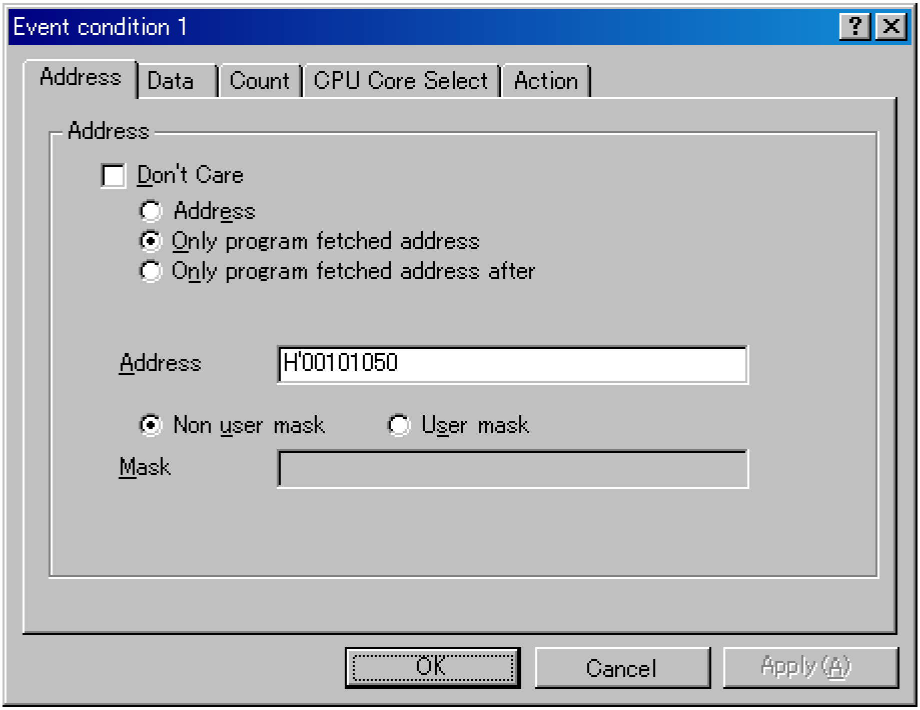


Figure 6.41 [Address] Page ([Event condition1] Dialog Box)

Note: The items that can be set in this dialog box differ according to the product. For details on the settings for each product, refer to the online help.

- Click the [OK] button.
- The first point display in the State line changes from `Disable` to `Enable`.
- The first point display in the Condition line changes from `None` to `Address = H'00101050 (tutorial.cpp/45) pc Core Select: CPU1 Break (CPU1).` Break (CPU1) is displayed for the first point in the Action line.
- Set the program counter and stack pointer values (PC = H'00000800 and R15 = H'FFF90000) that were set in section 6.8, Setting Registers, in the [Register] window of the High-performance Embedded Workshops for either CPU0 or CPU1. Click on the [Go] buttons of both High-performance Embedded Workshops.
  The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.
- If program execution is failed, reset the device and execute again the procedures above.

The program runs and then stops at the condition specified under Ch1 (IA_OA_DT_CT).

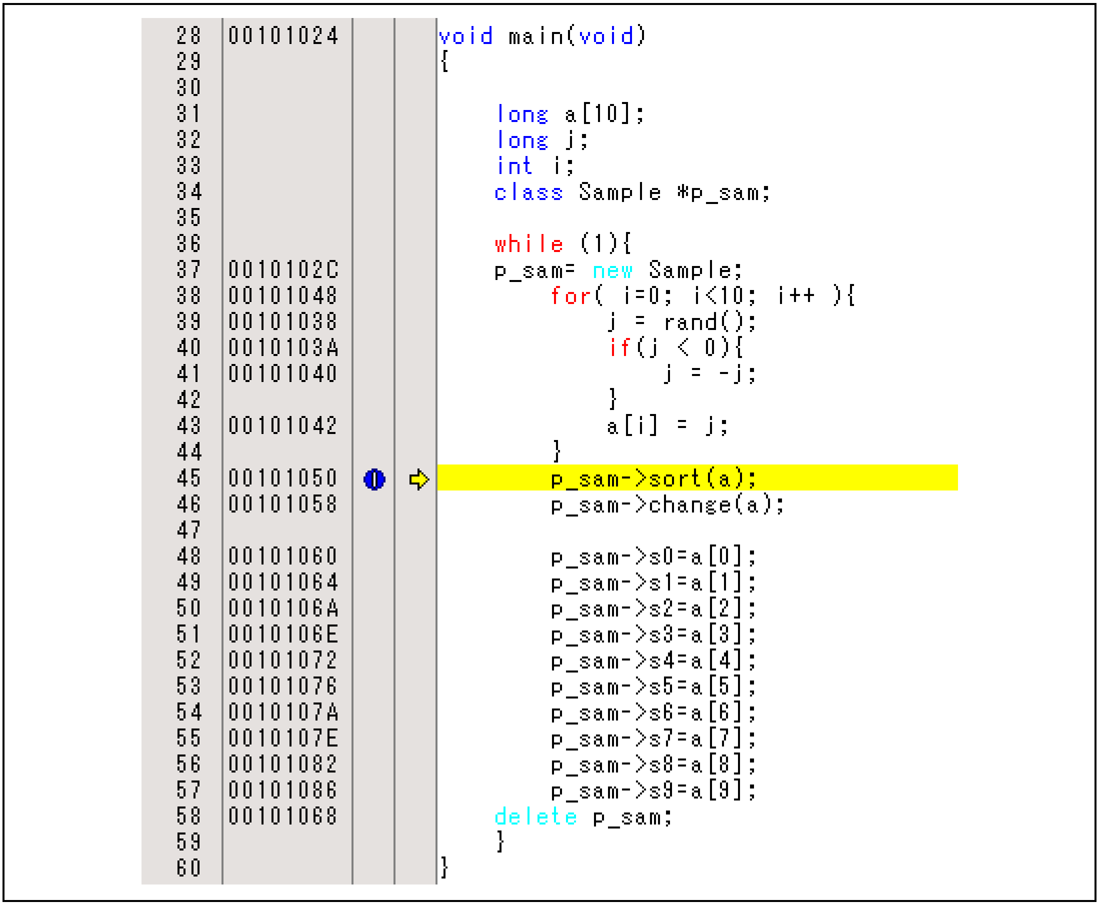


Figure 6.42 [Editor] Window at Execution Stop ([Ch1 (IA_OA_DT_CT)])

The [Status] window displays the following contents.

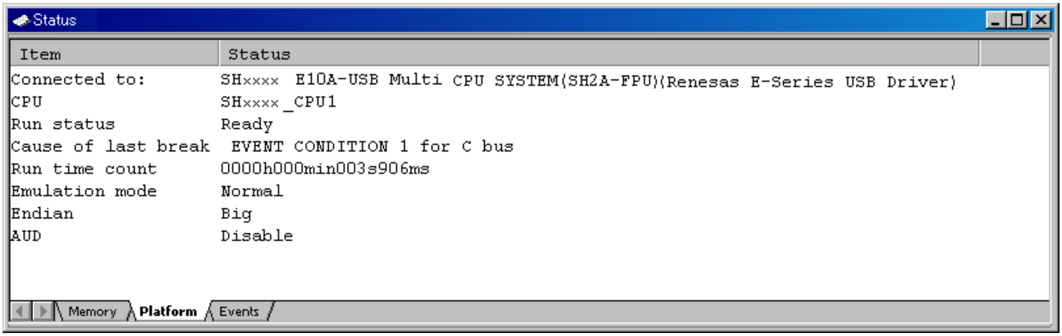


Figure 6.43 Displayed Contents of the [Status] Window ([Ch1 (IA_OA_DT_CT])

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

6.19.1 Setting the Sequential Break Condition

The emulator has sequential break functions.

Set hardware break conditions as follows:

Ch1 (IA_OA_DT_CT): A break condition is satisfied immediately after address H'00101050 is accessed.

Ch2 (IA_OA_DT): A break condition is satisfied immediately after address H'00101042 is accessed.

Follow the setting method described in the previous section.

To set these breakpoints as sequential:

- Select [Combination action (Sequential or PtoP)] from the popup menu by clicking the [Event] window with the right-hand mouse button. The [Combination action (Sequential or PtoP)] dialog box will open.

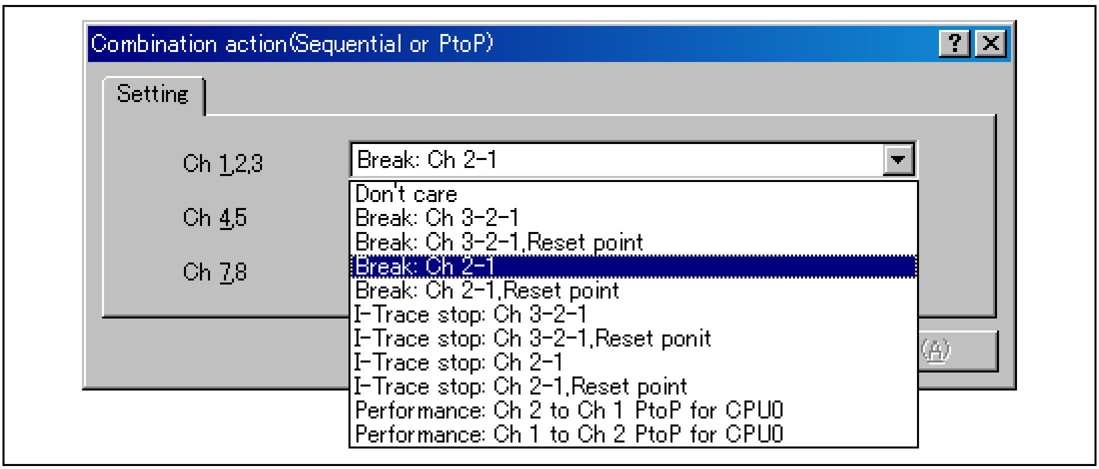


Figure 6.44 [Combination action (Sequential or PtoP)] Dialog Box

Note: The items that can be displayed in this dialog box differ according to the product. For the items that can be displayed, refer to the online help.

- Select [Break: Ch2-1] and click on the [OK] button.

When the setting is completed, the [Event] window will be as shown in figure 6.41.

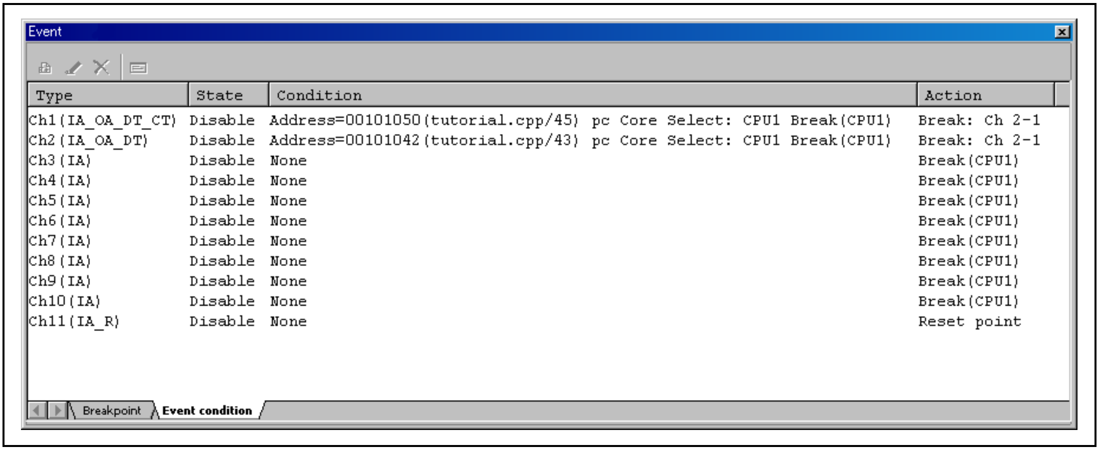


Figure 6.45 [Event condition] Page

Soon after set the [Combination action (Sequential or PtoP)], Ch1 (IA_OA_DT_CT) or Ch2 (IA_OA_DT) will be invalid. Select Ch1 (IA_OA_DT_CT) or Ch2 (IA_OA_DT) individually, and click the [Event] window with the right-hand mouse button and select [Enable].

Note: The items that can be displayed in this dialog box differ according to the product. For the items that can be displayed, refer to the online help.

- Set the program counter and stack pointer values (PC = H'00000800 and R15 = H'FFF90000) that were set in section 6.8, Setting Registers, in the [Register] windows of the High-performance Embedded Workshops for CPU0 and CPU1. Click on the [Go] buttons of both High-performance Embedded Workshops.

- If program execution is failed, reset the device and execute again the procedures above.

The program runs and then stops at the condition specified as Event Condition 1.

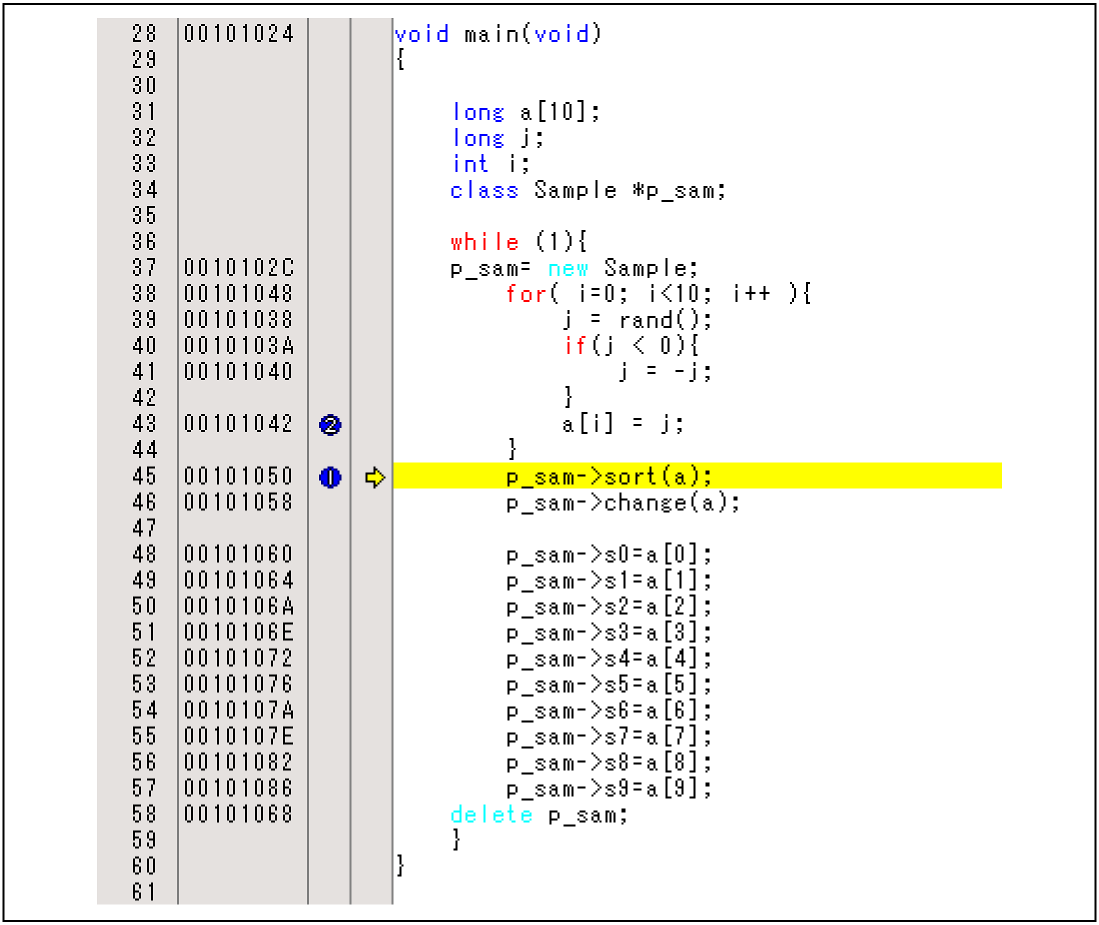


Figure 6.46 [Editor] Window at Execution Stop (Sequential Break)

The [Status] window displays the following contents.

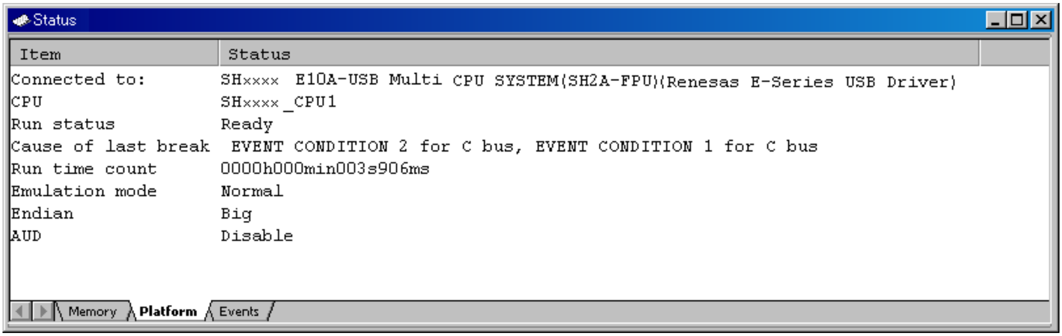


Figure 6.47 Displayed Contents of the [Status] Window (Sequential Break)

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

- The sequential break conditions that have been previously set are deleted. Click the [Event] window with the right-hand mouse button and select [Delete All] from the popup menu to cancel all hardware break conditions that have been set.
- Select [Combination action (Sequential or PtoP)] from the popup menu by clicking the [Event] window with the right-hand mouse button. The [Combination action (Sequential or PtoP)] dialog box will open (figure 6.40).
- Select the [Don’t care] from the [Ch 1,2, 3] drop-down list and click the [OK] button.

6.20 Trace Functions

The emulator has two branch-instruction trace functions.

- Internal Trace Function

  Refer to section 5.7, Viewing the Trace Information, to see the setting and displayed contents.

- AUD Trace Function

  This is the large-capacity trace function that is enabled when the AUD pin is connected to the emulator. When a set of the branch source and branch destination instructions is one branch, the maximum number of events acquired by a trace is 262,144.

  Refer to section 5.7, Viewing the Trace Information, to see the setting and displayed contents.

6.20.1 Displaying the [Trace] Window

Select [Trace] from the [Code] submenu of the [View] menu of the High-performance Embedded Workshop for CPU1. The result of the acquired trace is displayed.

6.20.2 Internal Trace Function

The branch source and branch destination information for the latest several branch instructions are displayed.

In the internal trace function, the type of the branch instruction can be specified and acquired.

The type is specified as follows:

- Select [Trace] from the [View] menu.
- Click the [Trace] window with the right-hand mouse button and select [Set…] from the popup menu to display the [Acquisition] dialog box.

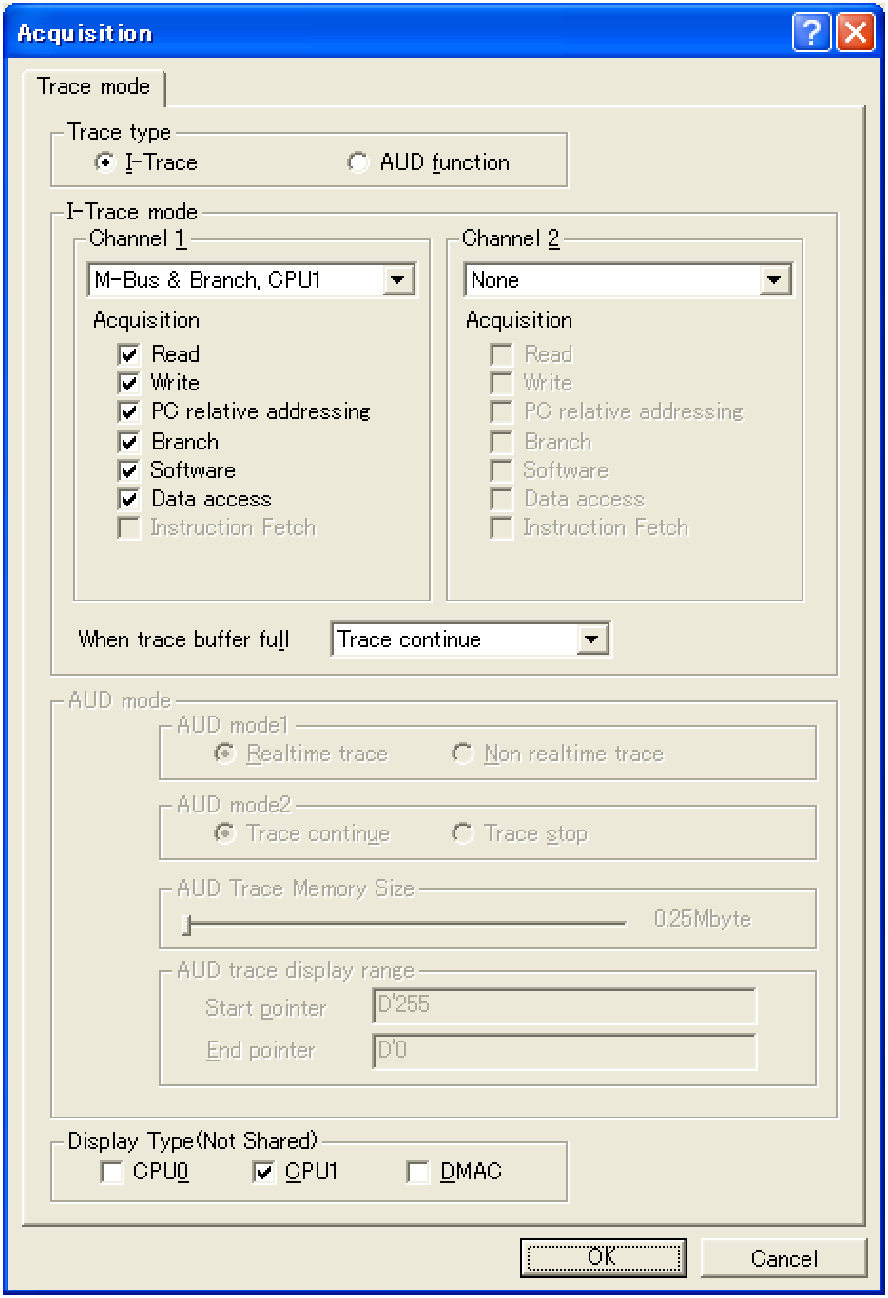
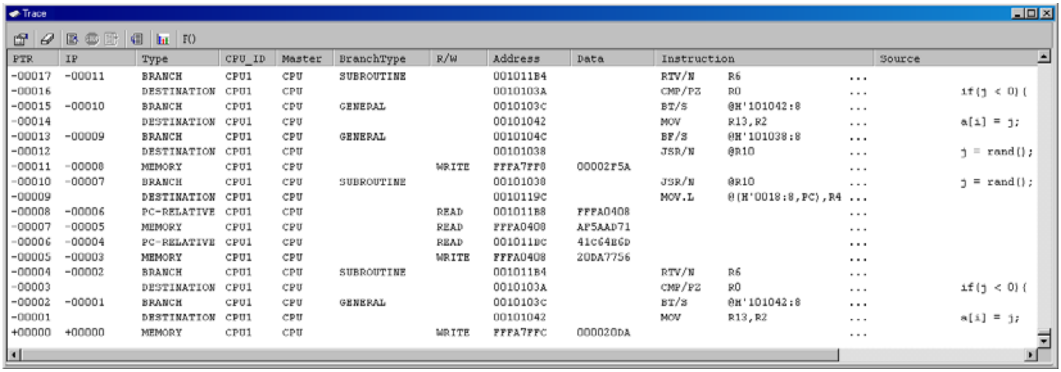


Figure 6.48 [Acquisition] Dialog Box

Note: The items that can be set in this dialog box differ according to the product. For details on the settings for each product, refer to the online help.

Run the program as shown in the example of section 6.19, Hardware Break Function. The trace results are displayed in the [Trace] window after the program execution is completed.



PTR	IP	Type	CPU_ID	Master	BranchType	R/W	Address	Data	Instruction	Source
-00017	-00011	BRANCH	CPU1	CPU	SUBROUTINE		001011B4		RTV/N R6 ...	
-00016		DESTINATION	CPU1	CPU			0010103A		CMP/PZ R0 ...	if(j < 0){
-00015	-00010	BRANCH	CPU1	CPU	GENERAL		0010103C		BT/S @H'101042:8 ...	
-00014		DESTINATION	CPU1	CPU			00101042		MOV R13,R2 ...	a[i] = j;
-00013	-00009	BRANCH	CPU1	CPU	GENERAL		0010104C		BF/S @H'101038:8 ...	
-00012		DESTINATION	CPU1	CPU			00101038		JSR/N @R10 ...	j = rand();
-00011	-00008	MEMORY	CPU1	CPU		WRITE	FFFA7FF8	00002F5A	...	
-00010	-00007	BRANCH	CPU1	CPU	SUBROUTINE		00101038		JSR/N @R10 ...	j = rand();
-00009		DESTINATION	CPU1	CPU			0010119C		MOV.L @(H'0018:8,PC),R4 ...	
-00008	-00006	PC-RELATIVE	CPU1	CPU		READ	001011B8	FFFA0408	...	
-00007	-00005	MEMORY	CPU1	CPU		READ	FFFA0408	AF5AAD71	...	
-00006	-00004	PC-RELATIVE	CPU1	CPU		READ	001011BC	41C64E6D	...	
-00005	-00003	MEMORY	CPU1	CPU		WRITE	FFFA0408	20DA7756	...	
-00004	-00002	BRANCH	CPU1	CPU	SUBROUTINE		001011B4		RTV/N R6 ...	
-00003		DESTINATION	CPU1	CPU			0010103A		CMP/PZ R0 ...	if(j < 0){
-00002	-00001	BRANCH	CPU1	CPU	GENERAL		0010103C		BT/S @H'101042:8 ...	
-00001		DESTINATION	CPU1	CPU			00101042		MOV R13,R2 ...	a[i] = j;
+00000	+00000	MEMORY	CPU1	CPU		WRITE	FFFA7FFC	000020DA	...	

Figure 6.49 [Trace] Window

- If necessary, adjust the column widths by dragging borders in the header bar (immediately below the title bar).

Note: The type and the amount of information that can be acquired by a trace differ according to the product. For details on the specifications of each product, refer to the online help.

6.20.3 AUD Trace Function

This function is available when the AUD pin of the MPU is connected to the emulator.

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

(1) Setting the trace acquisition mode

Display the [Trace] window.

Click the [Trace] window with the right-hand mouse button and select [Acquisition] from the popup menu to display the [Trace Acquisition] dialog box.

Select [AUD function] as the [Trace type].

The trace acquisition condition is set in the [Trace mode] page.

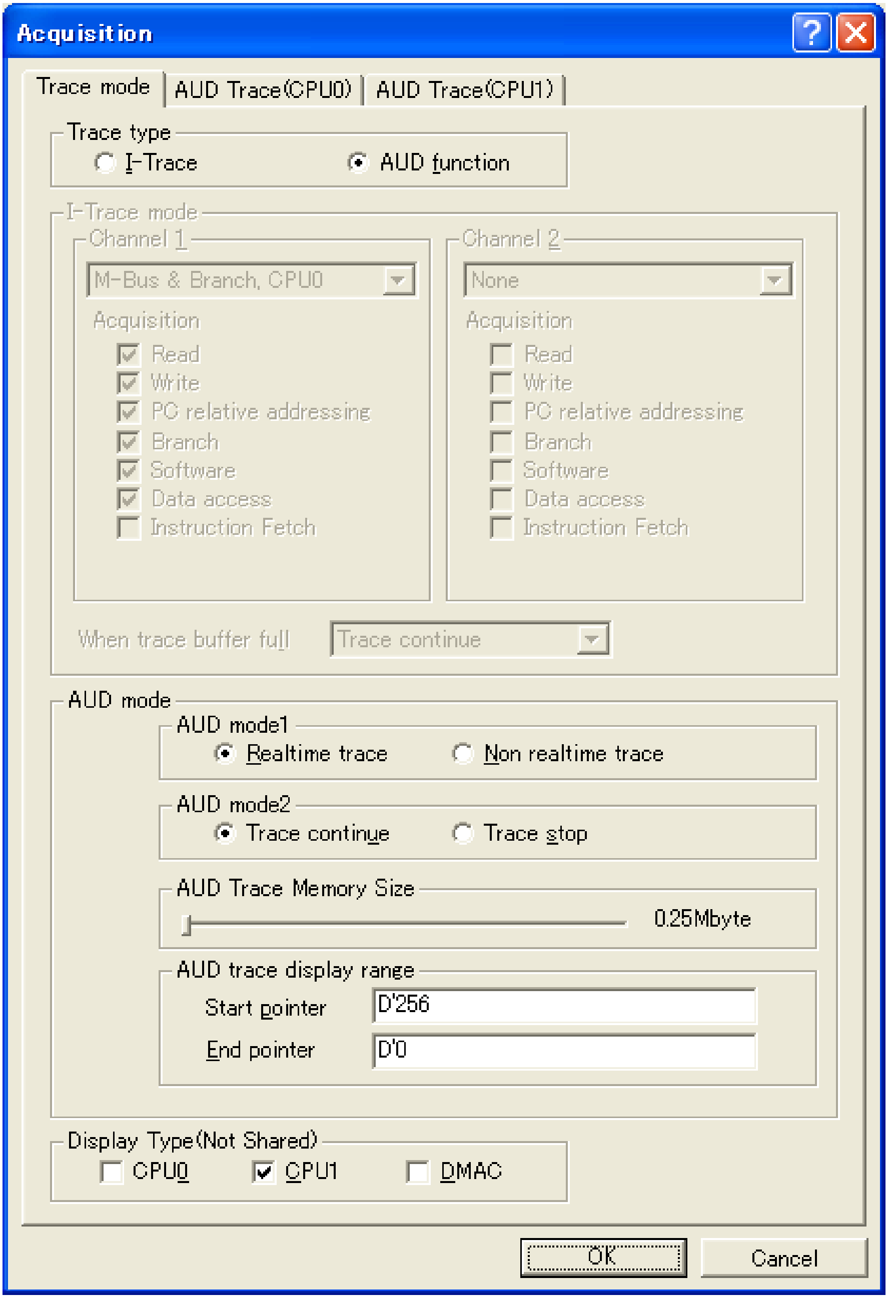


Figure 6.50 [Acquisition] Dialog Box

Note: This dialog box cannot be used in a product that does not support the AUD trace function. The items that can be set in this window differ according to the product. For details on the settings for each product, refer to the online help.

The following table shows the options.

AUD Trace Acquisition Mode

Type	Mode	Description
Continuous trace occurs	Realtime trace	When the trace information is being generated intensely that the output from the AUD pin incapable of keeping up, the CPU temporarily suspends the output of trace information. Therefore, although the user program is run in real time, the acquisition of some trace information might not be possible.
	Non realtime trace	When trace information is being generated so intensely that the output from the AUD pin is incapable of keeping up, CPU operations are temporarily suspended and the output of trace information takes priority. In such cases, the realtime characteristics of the user program are lost.
Trace buffer full	Trace continue	This function always overwrites the oldest trace information to acquire the latest trace information.
	Trace stop	When the trace buffer becomes full, the trace information is no longer acquired. The user program is continuously executed.

AUD Trace Display Range

	Type	Description
	Start pointer	Set the pointer to the start of the region for AUD tracing. The default is D′255.
	End pointer	Set the pointer to the end of the region for AUD tracing. The default is D′0

Note: The items that can be set in this window differ according to the product. For details on the settings for each product, refer to the online help.

(2) Displaying the trace result

- Run the program as shown in the example of section 6.19, Hardware Break Function. The trace results are displayed in the [Trace] window after the program execution is completed.

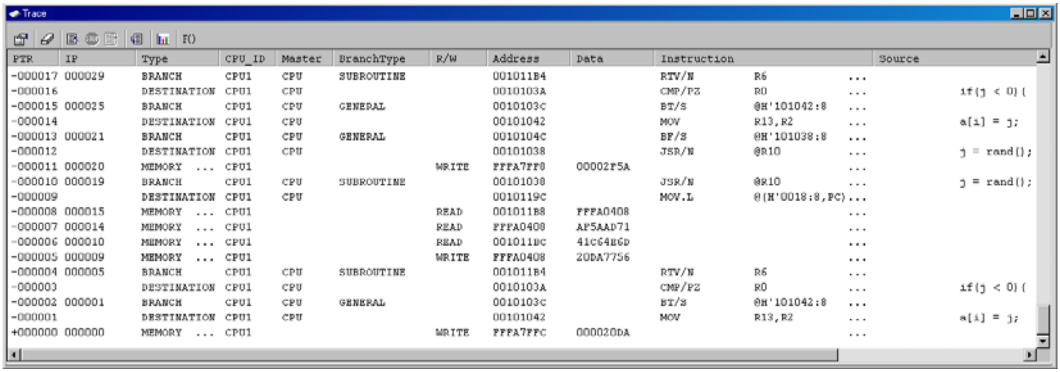


Figure 6.51 [Trace] Window (Example)

6.21 Stack Trace Function

The emulator uses the information on the stack to display the names of functions in the sequence of calls that led to the function to which the program counter is currently pointing.

Note: This function can be used only when the load module that has the Elf/Dwarf2-type debugging information is loaded. Such load modules are supported in SHC/C++ compiler (including OEM and bundle products) V6.0 or later.

- Double-click the [Event] column in the `sort` function in the High-performance Embedded Workshop for CPU1 and set an Event point.

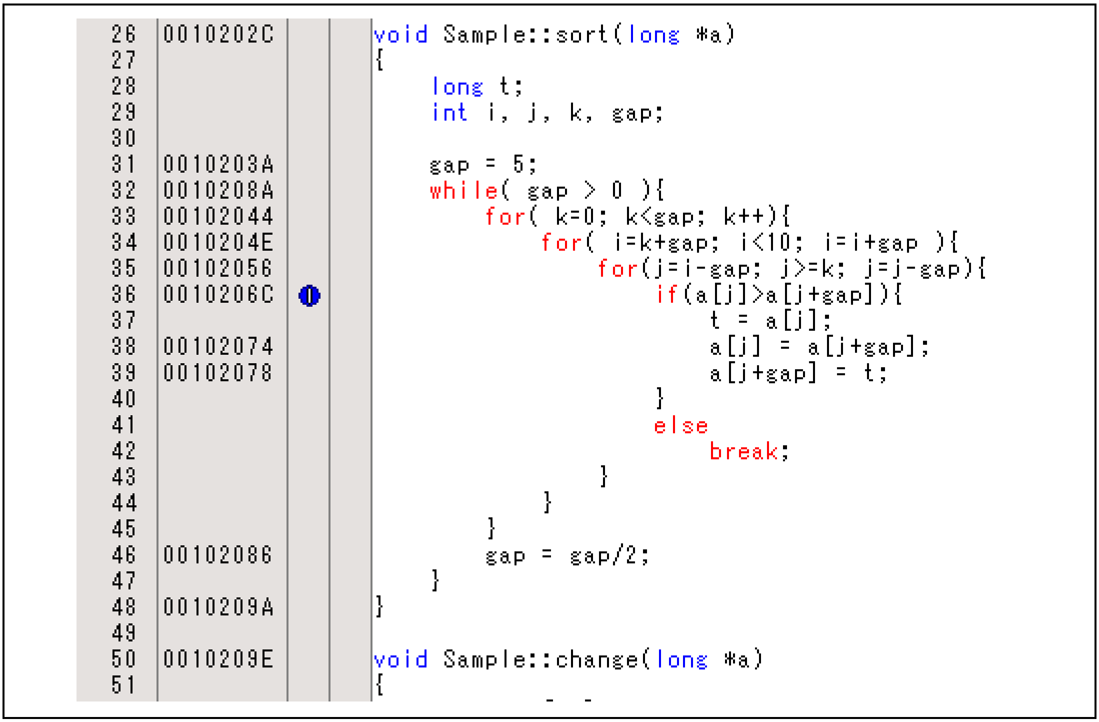


Figure 6.52 [Editor] Window (Hardware Break Setting)

- Set the same program counter and stack pointer values (PC = H'00000800 and R15 = H'FFF90000) as were set in section 6.8, Setting Registers (again, use the [Register] windows in the High-performance Embedded Workshops for CPU0 and CPU1). After that, click on the [Go] buttons in the High-performance Embedded Workshops for CPU0 and CPU1.
  The internal RAM area differs according to the MCU. Refer to the hardware manual for the MCU in use.
- If program execution is failed, reset the device and execute again the procedures above.
- After the break in program execution, select [Stack Trace] from the [Code] submenu of the [View] menu to open the [Stack Trace] window.

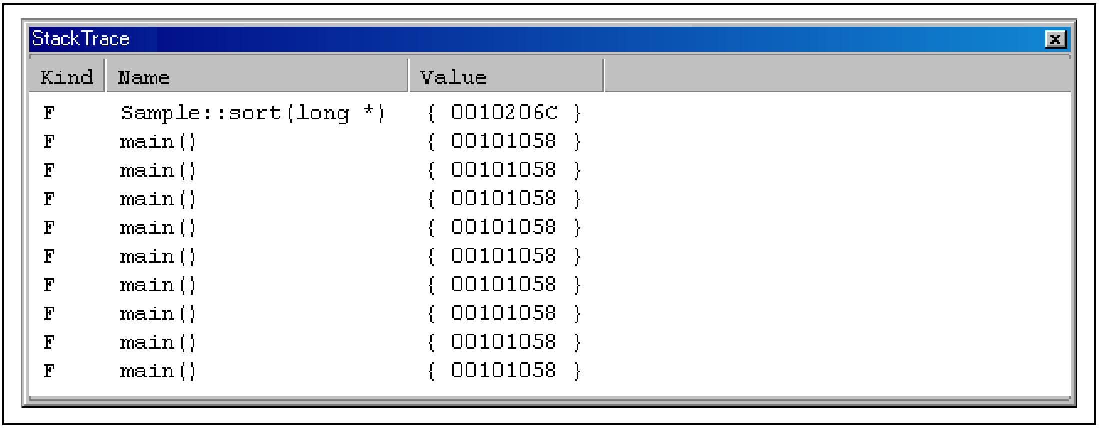


Figure 6.53 [Stack Trace] Window

Figure 6.49 shows that the position of the program counter is currently at the selected line of the `sort()` function, and that the `sort()` function is called from the `main()` function.

To remove the hardware break, double-click the [Event] column in the `sort` function again.

Note: For details on this function, refer to the online help.

6.22 Performance Measurement Function

The emulator has performance measurement functions.

- Performance measurement function

  This function applies a counter in the MPU to measure the number of times various events have occurred and cycle count. A start and end condition for counting can be set.

  Various items that can be measured differ according to the supported MPU.

6.22.1 Performance Measurement Function

The following is an example of the use of a counter in the MPU to measure the number of times various events have occurred and cycle count.

(1) Setting method

Select [Performance Analysis] from the [Performance] submenu of the [View] menu of the High-performance Embedded Workshop for CPU1.
When the [Select Performance Analysis Type] dialog box will open, click the [OK] button.

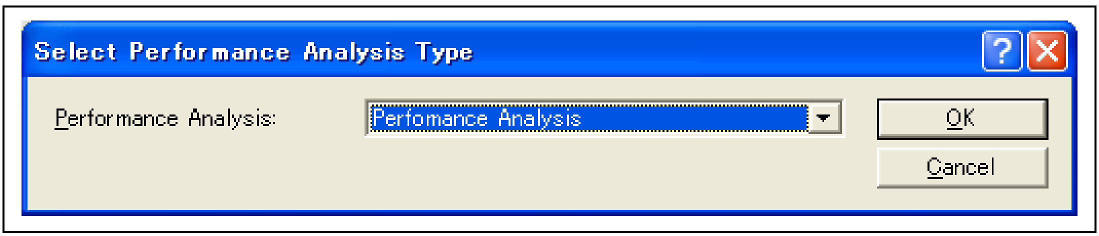


Figure 6.54 [Select Performance Analysis Type] Dialog Box

- The [Performance Analysis] window will be displayed.
- Double-click on the channel column in this window. The [Performance Analysis] dialog box will open. The events to be measured and measuring conditions can be set in this dialog box.

Note: The items that can be displayed in this dialog box differ according to the product. For details on the settings for each product, refer to the online help.

After the conditions have been set, clicking the [OK] button and executing the user program will display the result of measurement in the [Performance Analysis] window.

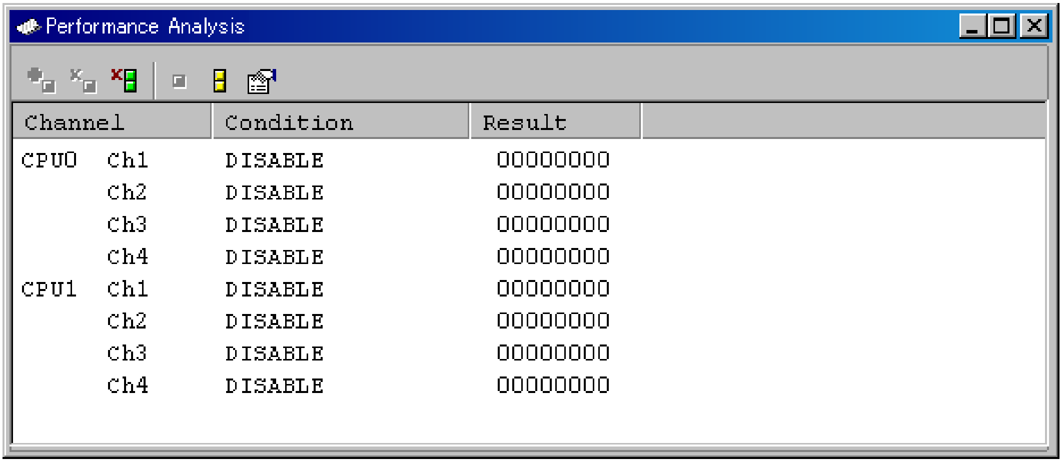


Figure 6.55 [Performance Analysis] Window

Note: The items that can be displayed in this window differ according to the product. For details on the settings for each product, refer to the online help.

6.23 Download Function to the Flash Memory Area

The emulator enables downloading to the external flash memory area. This function requires a program for programming the flash memory (hereinafter referred to as a write module), a program for erasing the flash memory (hereinafter referred to as an erase module), and the RAM area for downloading and executing these modules.

Notes: 1. The write and erase modules must be prepared by the user.

2. This function is not available depending on the MCU. For such an MCU, the [Loading flash memory] page shown in figure 6.56 will not be displayed.

- Interface with write and erase modules and emulator firmware

  The write and erase modules must be branched from the emulator firmware. To branch from the emulator firmware to the write and erase modules, or to return from the write and erase module to the emulator firmware, the following conditions must be observed:

  — Describe all the write and erase modules with the assembly language.

  — Save and return all the general register values and control register values before and after calling the write or erase module.

  — Return the write or erase module to the calling source after processing.

  — The write and erase module must be a Motorola-type file.

  The module interface must be as follows to pass correctly the information that is required for flash memory accessing.

Table 6.2 Module Interface

Module Name	Argument	Return Value
Write module	R4(L): Write address R5(L): Access size 0x4220 = byte, 0x5720 = word, 0x4C20 = longword R6(L): Write data	R0(L): End code Normal end = 0, Abnormal end = other than 0
Erase module	R4(L): Access size 0x4220 = byte, 0x5720 = word, 0x4C20 = longword	None

Note: The (L) means the longword size.

Note: Write module: The write data for the access size is set to the R6 register. When the access size is word or byte, 0 is set to the upper bits of the R6 register.

- Flash memory download method

  For downloading to the flash memory, set the items on the [Loading flash memory] page in the [Configuration] dialog box, which is opened from [System…], then [Emulator] from the [Setup] menu.

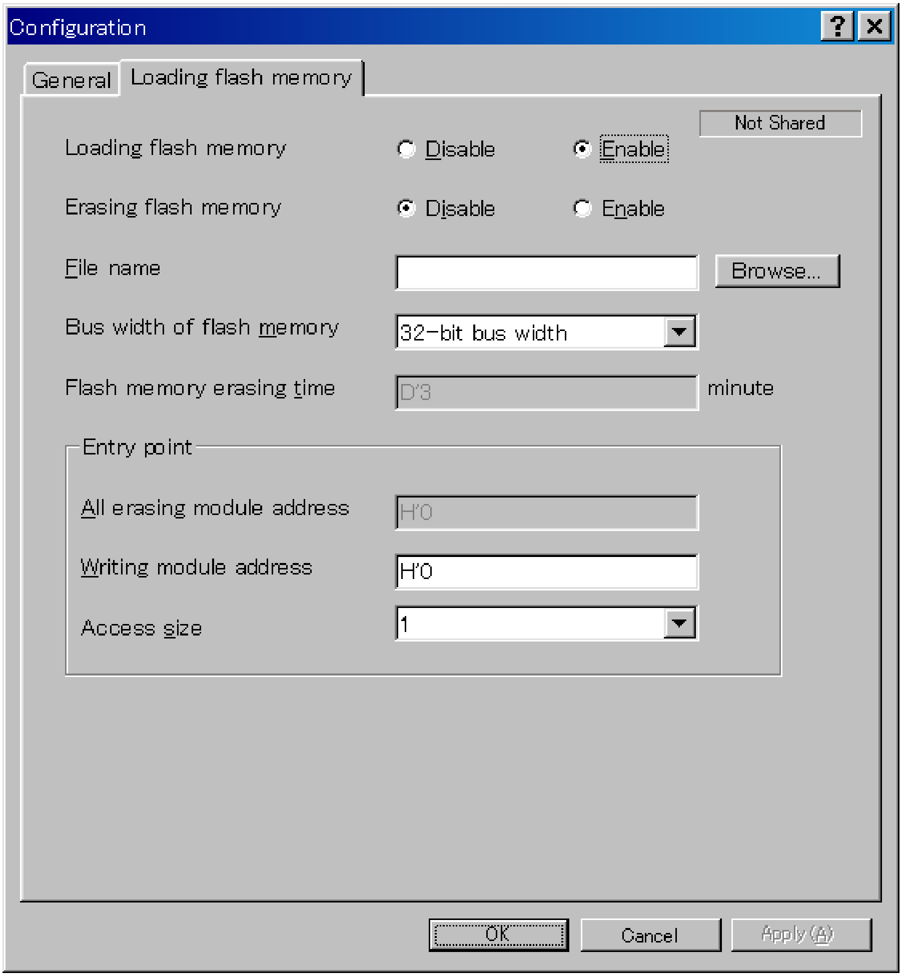


Figure 6.56 [Loading flash memory] Page

Table 6.3 shows the options for the [Loading flash memory] page.

Table 6.3 [Loading flash memory] Page Options

Option	Description
[Loading flash memory] radio button	Sets Enable for flash memory downloading. When Enable is selected, and [File load] is selected from the [File] menu for downloading, the write module is always called. Enable: Download to the flash memory Disable: Not download to the flash memory
[Erasing flash memory] radio button	Sets Enable for erasing before the flash memory is programmed. When Enable is selected, the erase module is called before calling the write module. Enable: Erase the flash memory Disable: Not erase the flash memory
[File name] edit box	Sets the file name of the S-type load module including the write and erase modules. The file that has been set is loaded to the RAM area before loading to the flash memory. A maximum of 128 characters can be input for the file name.
[Bus width of flash memory] list box	Sets the bus width of the flash memory.
[Flash memory erasing time] edit box*	Sets the TIMEOUT value for erasing the flash memory. Set a larger value if erasing requires much time; the default time is three minutes. The radix for the input value is decimal. It becomes hexadecimal by adding H'.
[Entry point] group box	Sets the calling destination address or access size of the write and erase modules. [All erasing module address] edit box: Inputs the calling destination address of the erase module. [Writing module address] edit box: Inputs the calling destination address of the write module. [Access size] combo box: Selects the access size of the RAM area where the write/erase module is loaded.

Note: Although the values that can be set are D'1 to D'65535, the TIMEOUT period may be extended according to the set value. Therefore, it is recommended to input the minimum value by considering the erasing time of the flash memory in use.

- Notes on using the flash memory download function

  The following are notes on downloading to the flash memory.

  — When the flash memory download is enabled, downloading to areas other than the flash memory area is disabled.

  — Downloading is only enabled to the flash memory area. Perform memory write or PC break only to the RAM area.

  — When the flash memory erase is enabled, the [Stop] button cannot stop erasing.

  — The area for the write and erase modules must be set in an MMU-disabled space.

- An example of downloading to the flash memory

  The following is an example of downloading to the flash memory manufactured by Intel Corporation (type number: G28F640J5-150). A sample is provided in the \Fmtool folder in the installation destination folder. Create a program that suits the user specifications by referring to this sample.

Table 6.4 Board Specifications

Item		Contents
SDRAM address		H'0C000000 to H'0FFFFFFF
Flash memory address		H'00000000 to H'01FFFFFF
Bus width of flash memory		32 bits
Operating environment	CPU internal frequency	167 MHz
	Bus frequency	55.7 MHz
	CPU internal module frequency	27.84 MHz
	Endian	Big endian

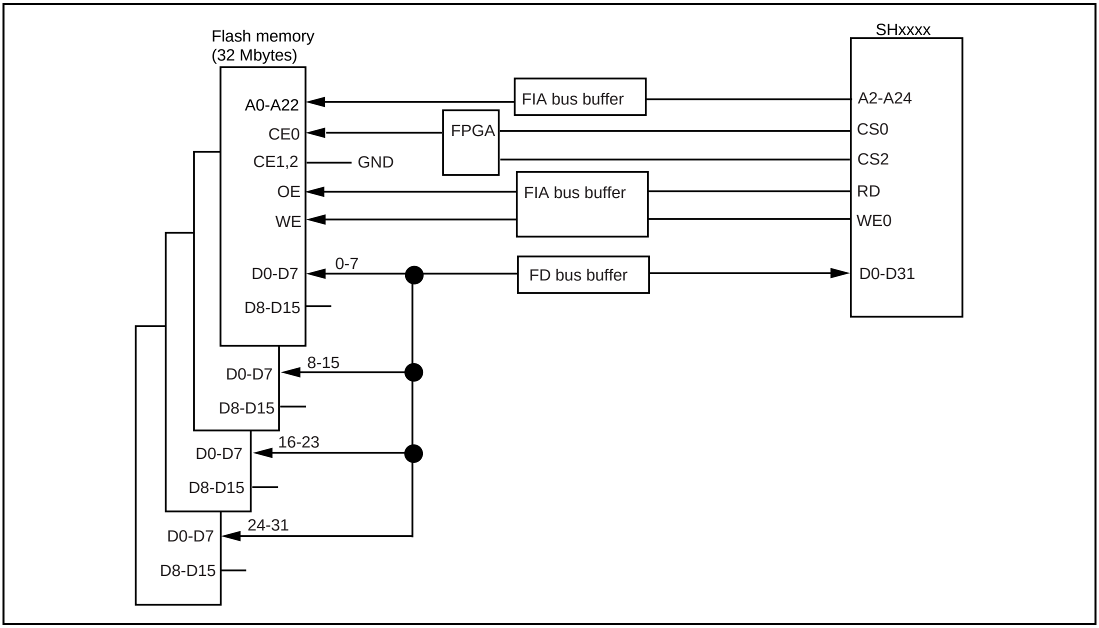


Figure 6.57 Flash Memory Wiring

Table 6.5 Sample Program Specifications

Item	Contents
RAM area to be used	H'0C001000 to H'0C0015BF
Write module start address	H'0C001100
Erase module start address	H'0C001000

- Since the SDRAM is used, the bus controller must be set.
- Set the options on the [Loading flash memory] page in the [Configuration] dialog box as follows:

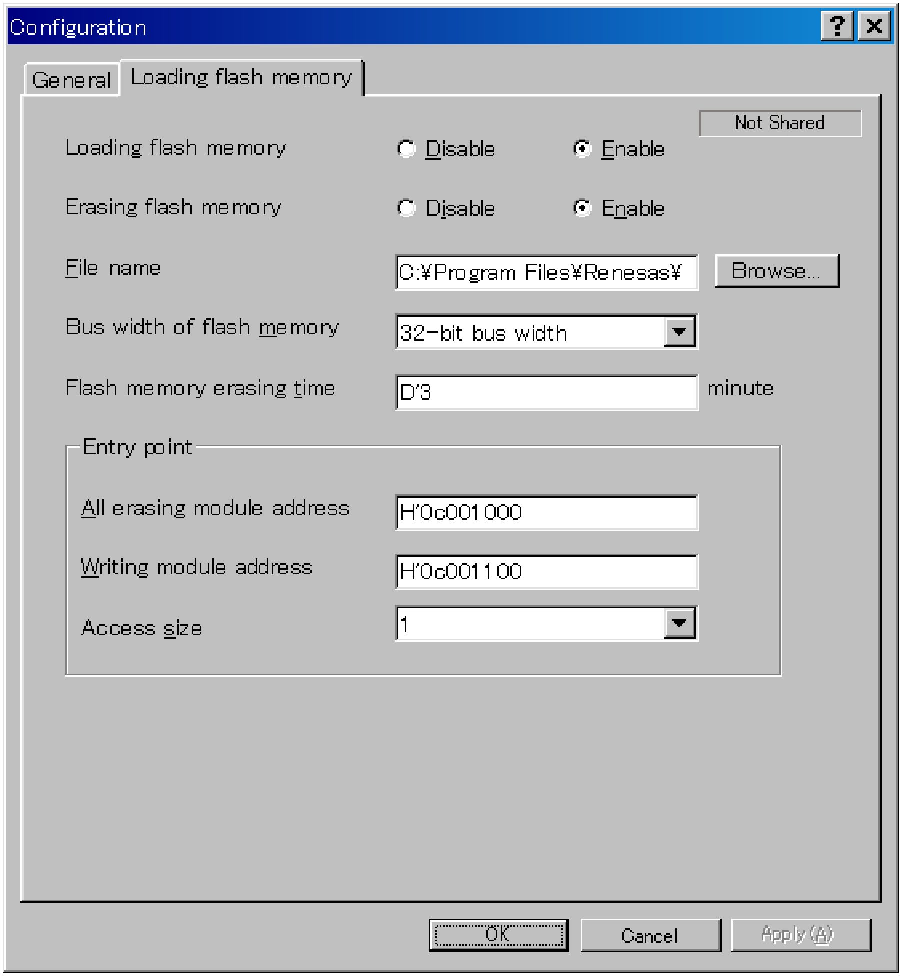


Figure 6.58 [Loading flash memory] Page

Notes: 1. When the data has already been written in the flash memory, be sure to select [Enable] for [Erasing flash memory]. If [Disable] is selected, a verify error occurs.

2. When [Erasing flash memory] is selected, it takes about one minute to erase the flash memory (in this example).

- Select the object for downloading to the flash memory area.

Section 7 Tutorial [SH-4A]

7.1 Introduction

This section describes the main functions of the emulator by using a tutorial program.

Explanations where something else is not stated apply to operations of the High-performance Embedded Workshop for CPU0. The tutorial program is written in C++ and runs through the High-performance Embedded Workshops for CPU0 and CPU1 to sort ten random data items into ascending or descending order. The tutorial program performs the following actions:

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

The file `tutorial.cpp` contains source code for the tutorial program. The file `Tutorial.abs` is a compiled load module in the Elf/Dwarf2 format.

Notes: 1. Operation of `Tutorial.abs` is big endian. For little-endian operation, `Tutorial.abs` must be recompiled. After recompilation, the addresses may differ from those given in this section.

2. This section describes general usage examples for the emulator. For the specifications of particular products, refer to the additional document, ″Supplementary Information on Using the SHxxxx″, or the online help.

3. The operation address of `Tutorial.abs` attached to each product differs depending on the product. Replace the address used in this section with upper 16 bits of the actually loaded address.
Example: Although the PC address is H’0000006c in the manual, enter H’0C00xxxx when the loaded address of `Tutorial.abs` is H’0C00006c (upper bit H’0000 is changed to H’0C00).

4. The displayed addresses and data may differ from those given in this section depending on the MCU to be used.

7.2 Running the High-performance Embedded Workshop

Selects [Renesas] → [High-performance Embedded Workshop] → [High-performance Embedded Workshop from the [Program] item in the [Start] menu.

7.3 Setting up Synchronized Debugging

(1) The [Welcome!] dialog box is displayed.

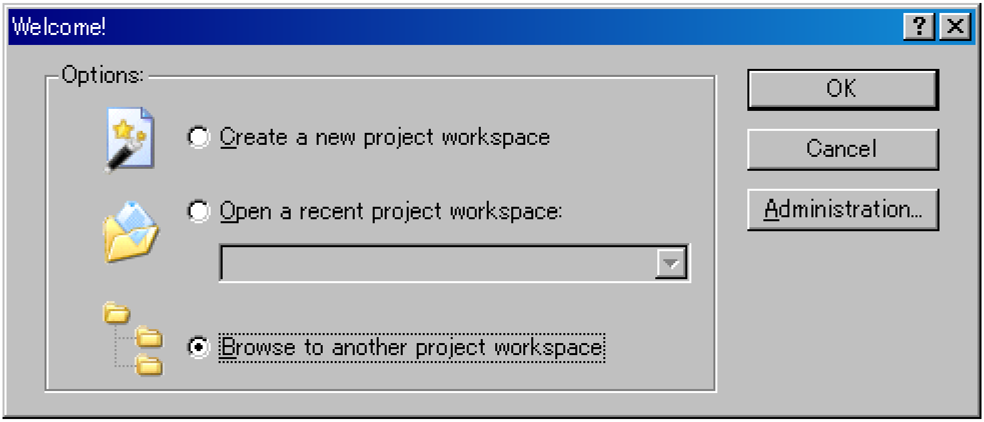


Figure 7.1 [Welcome!] Dialog Box

Click the [Cancel] button here.

(2) Select the [Synchronized debugs from the [Debug] menu to open the dialog box shown below.

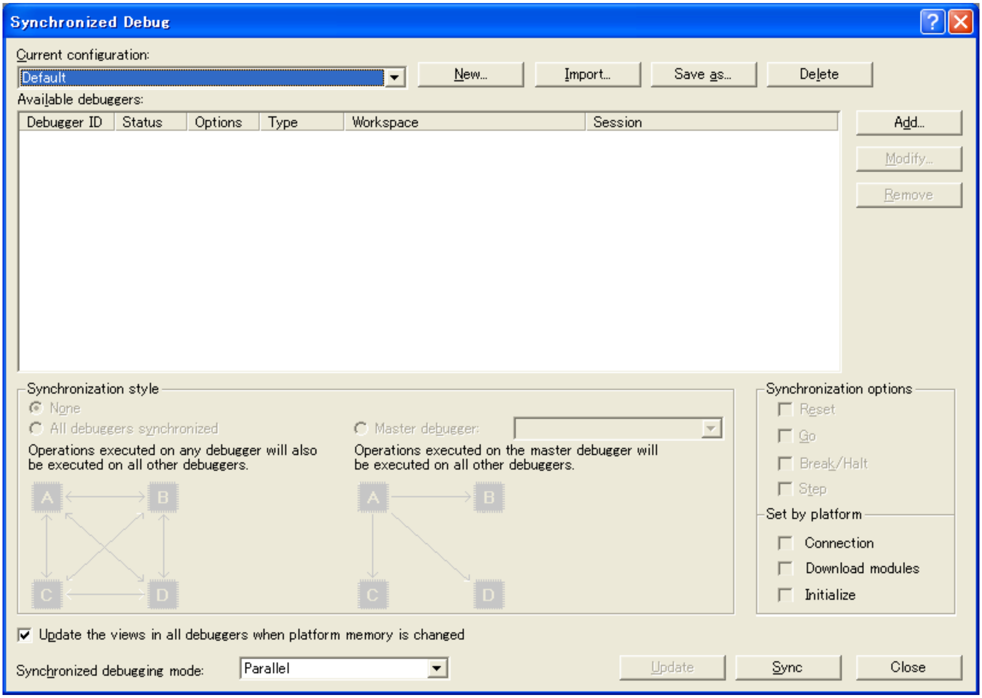


Figure 7.2 [Synchronized Debug] Dialog Box

(3) Click on [New…] and enter the [Setting name for the Synchronized Debug]; for this tutorial, we have used SH4AM_Tutorial.

(4) Click on the [Add] button to open the [Add debugger] dialog box.
Click on the [Browse] button then find and read in:
<Drive where the OS has been installed>:
\WorkSpace\Tutorial\E10A-USBM\SH4AM\Tutorial\SH4AM_Internal\SH4AM_Internal.hws
(**** represents the user name).
Click on the [OK] button, and select the [tutorial_CPU0] from the [Project] drop-down list.

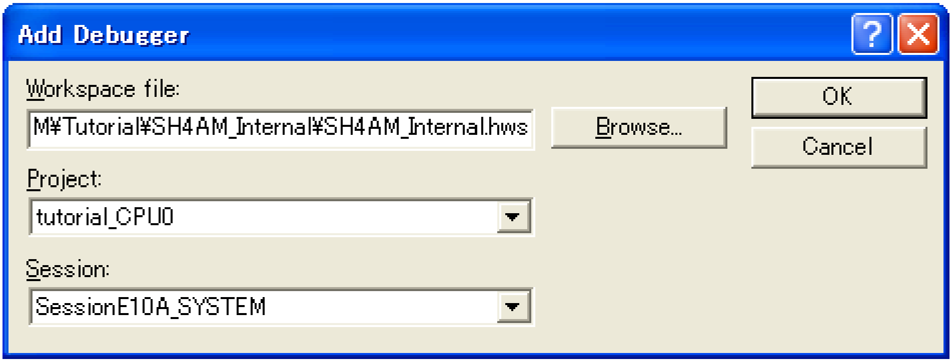


Figure 7.3 [Add Debugger] Dialog Box

Note: In case of failure to read the workspace, open the workspace and then read it, in accord with the procedure described in section 3.9, System Check.

(5) Click on the [Add] button again to open the [Add debugger] dialog box.
Check whether the workspace which was read previously to the [Workspace file] is displayed. If this is not the case, read the workspace again by following the procedure 4, above.
Selects [tutorial_CPU1] from the [Project] drop-down list and then click on the [OK] button.

(6) Set up the synchronized debugging state.
Select [All debuggers synchronized] under [Synchronization style] and check all of the following check boxes under [Synchronization options]: [Go], [Break/Halt], and [Step]. Select “Parallel” from the [Synchronization debugging mode] drop-down list.

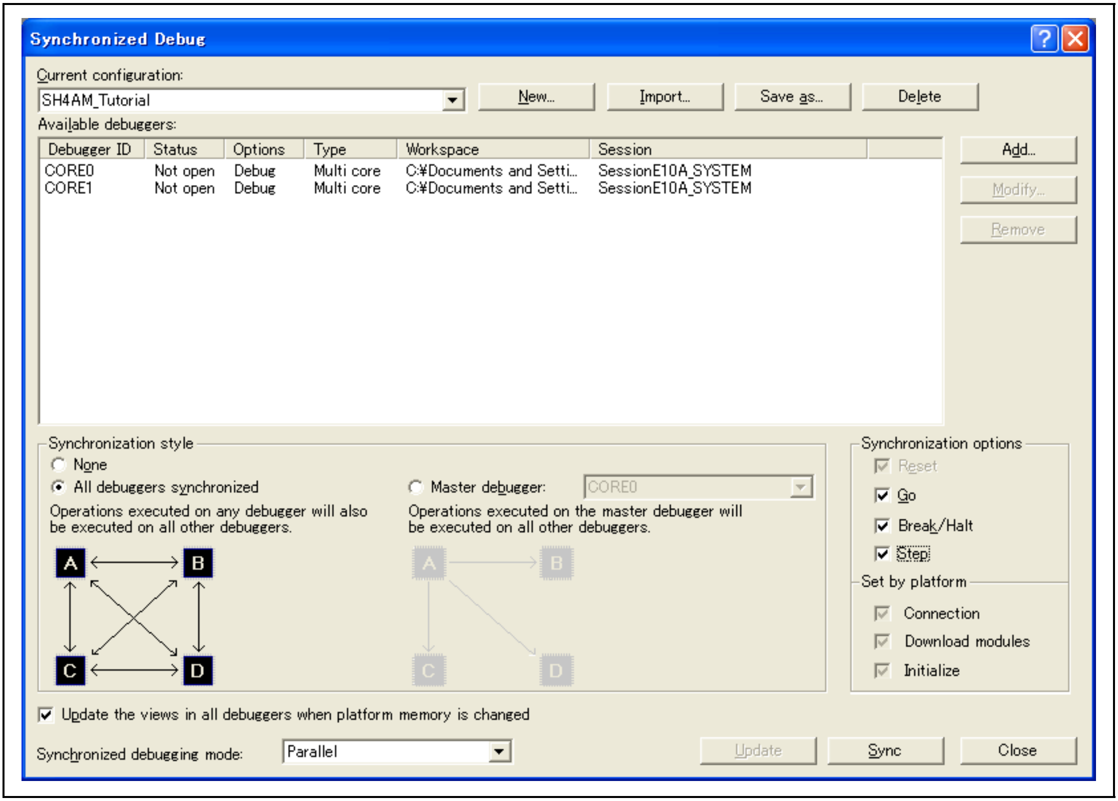


Figure 7.4 [Synchronized Debug] Dialog Box

Click on the [Synchronized Debug] button and start up the High-performance Embedded Workshop by following the procedure in section, 3.9 System Check.

7.4 Setting up the Emulator

The clocks which are used for data communications must be set up on the emulator before the program is downloaded.

- AUD clock

  A clock used in acquiring AUD traces.

  If its frequency is set too low, complete data may not be acquired during realtime tracing.

  The frequency setting must also not exceed the supported device’s upper limit for the AUD clock.

  The AUD clock is only required for the AUD trace function.
- JTAG (H-UDI) clock (TCK)

  A communication clock used except for acquiring AUD trace.

  If its frequency is set too low, the speed of downloading will be lowered.

  Set the frequency not to exceed the upper limit for the MPU’s guaranteed TCK range.

  For details of the limitations on both clocks, refer to section 2.2.4, Notes on Using the JTAG (H-UDI) Clock (TCK) and AUD Clock (AUDCK), in the additional document, ″Supplementary Information on Using the SHxxxx″.

The following is a description of the procedure used to set the clocks.

7.5 Setting the [Configuration] Dialog Box

- Select [Emulator] then [Systems…] from the [Setup] menu in the High-performance Embedded Workshop for CPU0 to set a communication clock. The [Configuration] dialog box is displayed. Open the [Common Setting] page.

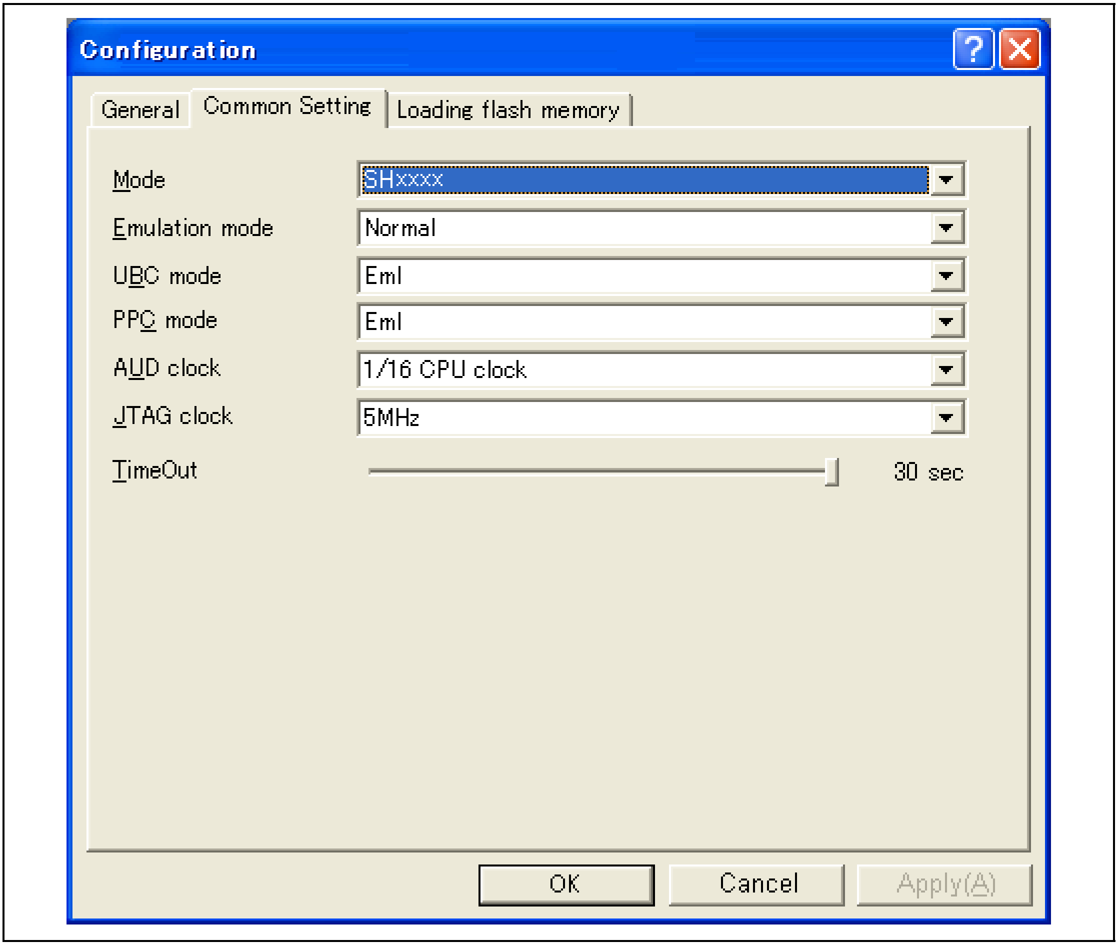


Figure 7.5 [Configuration] Dialog Box

- Set appropriate values in the [AUD clock] and [JTAG clock] combo boxes. The clock also operates with the default value.

Note: The items that can be set in this dialog box differ according to the product. For details on the settings for each product, refer to the online help.

- Click the [OK] button to set a configuration.

7.6 Checking the Operation of the Target Memory for Downloading

Check that the destination memory area for downloading is operating correctly.

When the destination memory is SDRAM or DRAM, a register in the bus controller of the CPU must be set before downloading. Set the bus controller correctly in the [IO] window according to the memory type to be used.

When the required settings, such as the settings for the bus controller, have been completed, display and edit the contents of the destination memory in the [Memory] window in the High-performance Embedded Workshop for CPU0 to check that the memory is operating correctly.

Note: The above way of checking the operation of memory may be inadequate. It is recommended that a program for checking the memory be created.

- Select [Memory…] from the [CPU] submenu of the [View] menu and enter ***H'00000000, H'00000000,*** and ***H'FFFFFFFF*** in the [Display address], [Scroll Start Address], and [Scroll End Address] edit boxes, respectively.

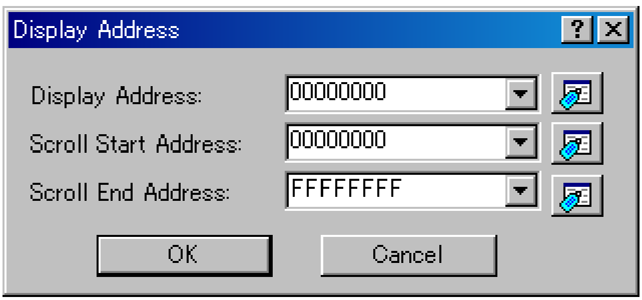


Figure 7.6 [Display Address] Dialog Box

- Click the [OK] button. The [Memory] window is displayed and shows the specified memory area.

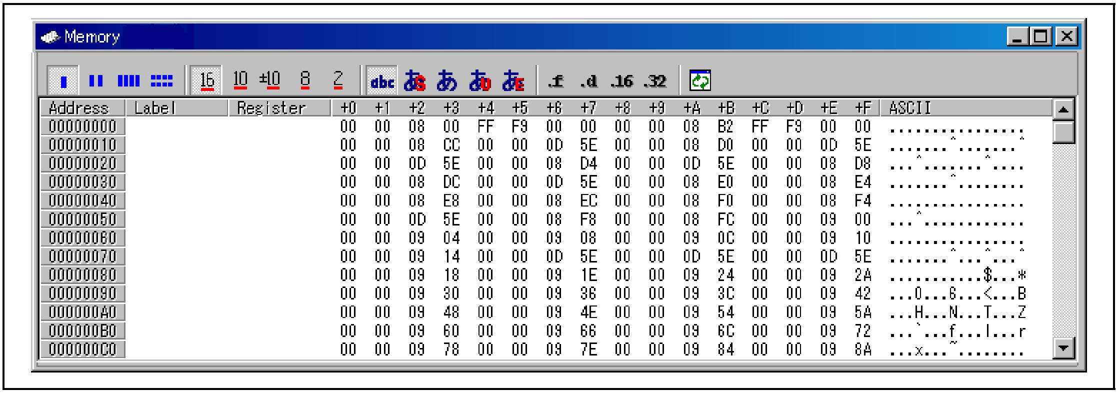


Figure 7.7 [Memory] Window

- Placing the mouse cursor on a point in the display of data in the [Memory] window and double-clicking allows the values at that point to be changed. Data can also be directly edited around the current position of the text cursor.

7.7 Downloading the Tutorial Program

7.7.1 Downloading the Tutorial Program

Download the object program to be debugged.

To proceed with source-level debugging with the High-performance Embedded Workshop for CPU0 or the High-performance Embedded Workshop for CPU1, download the debugging information file for the corresponding CPU.

- Select [Download module] from [Tutorial.abs] under [Download modules].

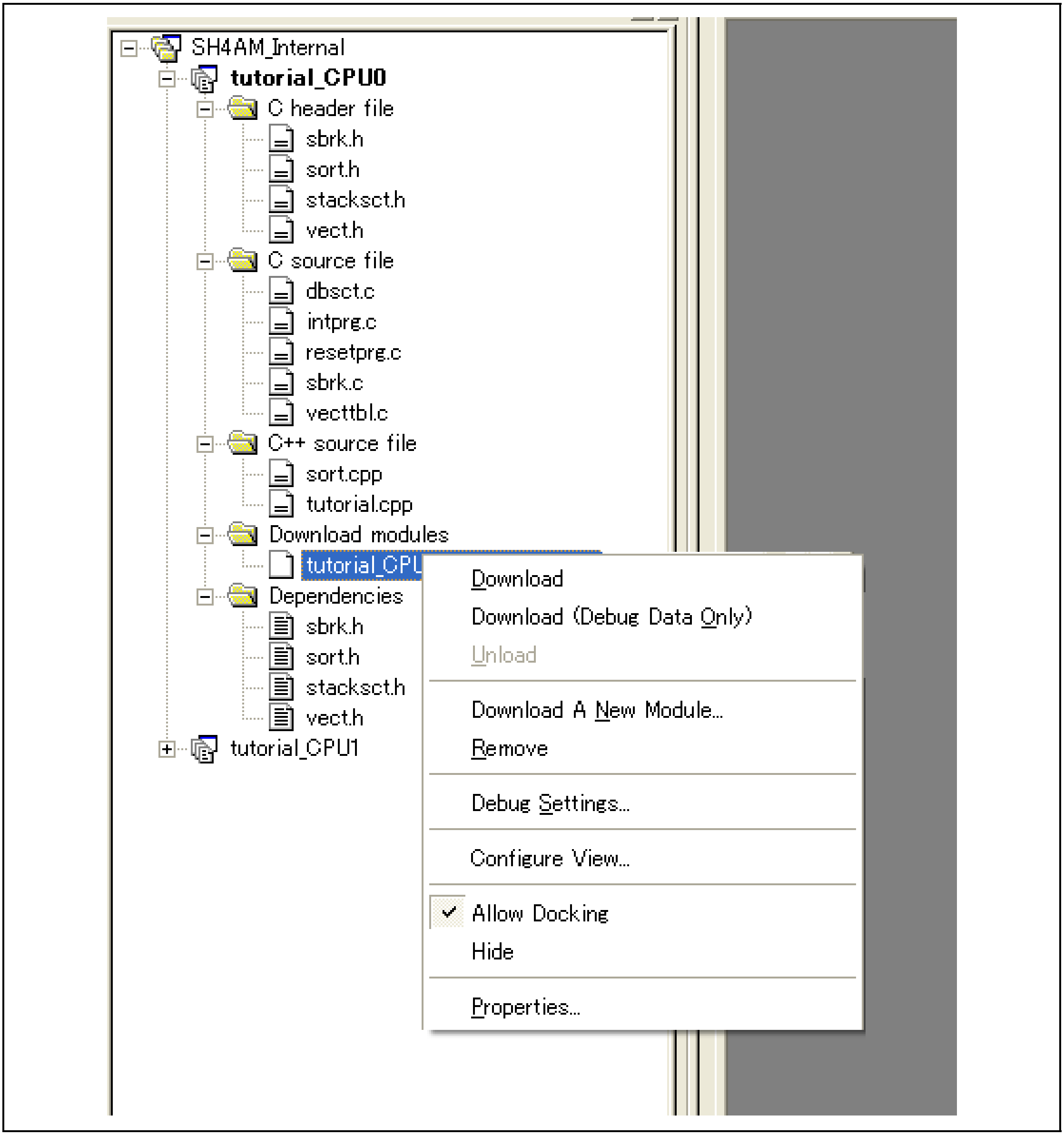


Figure 7.8 Downloading the Tutorial Program

7.7.2 Displaying the Source Program

The High-performance Embedded Workshop allows the user to debug a user program at the source level.

- Double-click [tutorial.cpp] under [C++ source file] in the High-performance Embedded Workshop for CPU0.

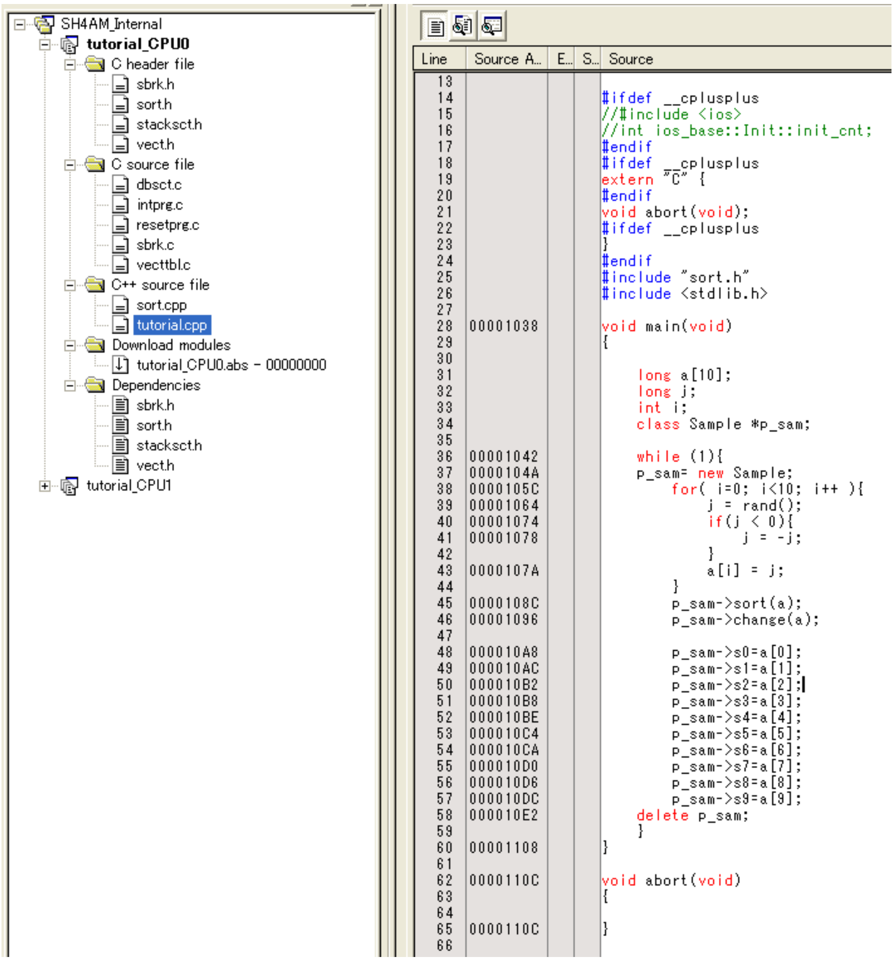


Figure 7.9 [Editor] Window (Displaying the Source Program)

Select a font and size that are legible from the [Format…] option in the [Setup] menu if necessary.

Initially the [Editor] window shows the start of the user program, but the user can use the scroll bar to scroll through the user program and look at the other statements.

7.8 Setting a PC Breakpoint

A PC breakpoint is a simple debugging function.

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

- Select by double-clicking the [S/W breakpoint] column on the line containing the `sort` function call in the High-performance Embedded Workshop for CPU0.

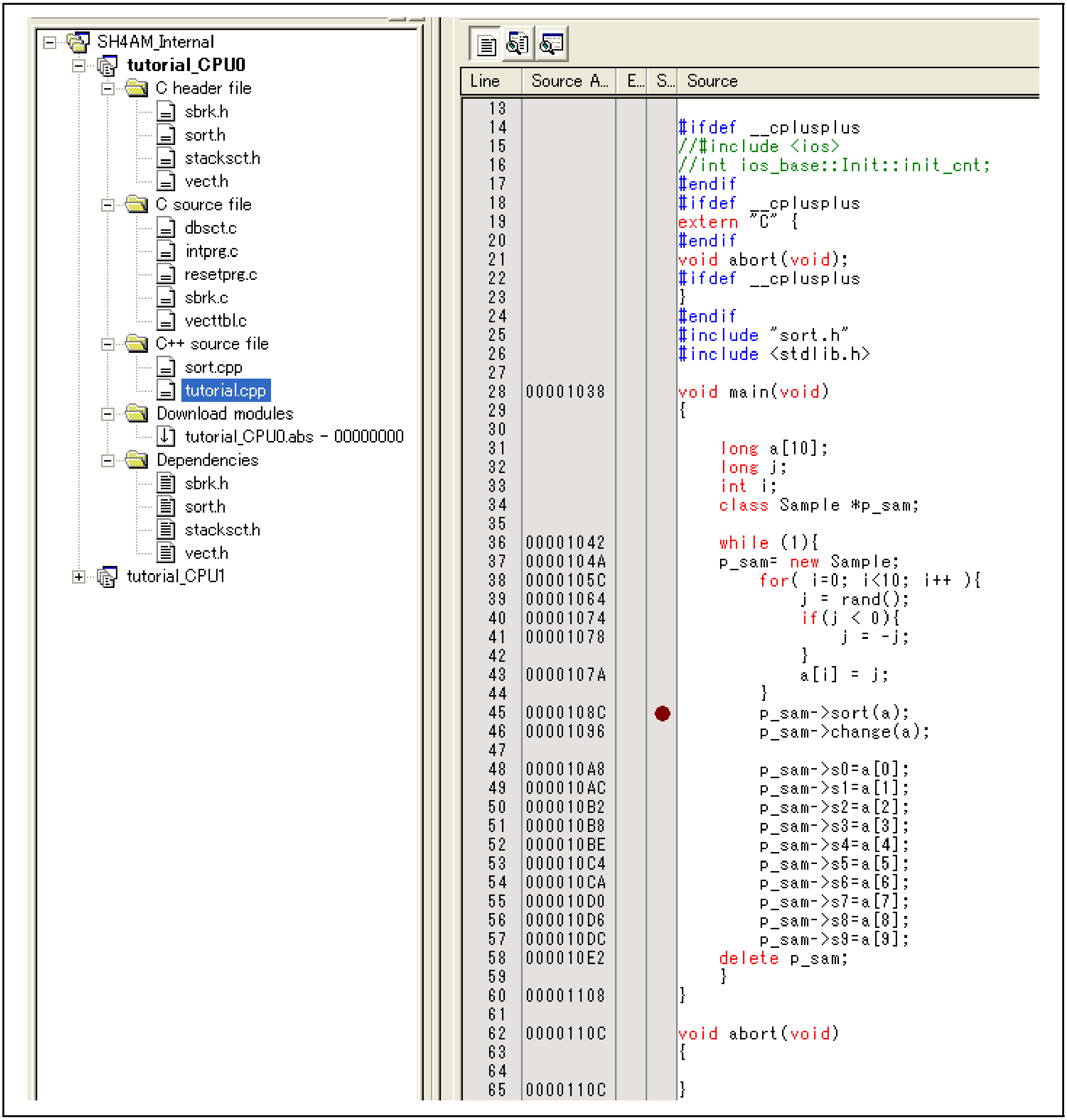


Figure 7.10 [Editor] Window (Setting a PC Breakpoint)

The symbol • will appear on the line containing the `sort` function. This shows that a PC breakpoint has been set.

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

7.9 Setting Registers

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

- Select [Registers] from the [CPU] submenu of the [View] menu in the High-performance Embedded Workshop for CPU0 and CPU1. The [Register] window is displayed.

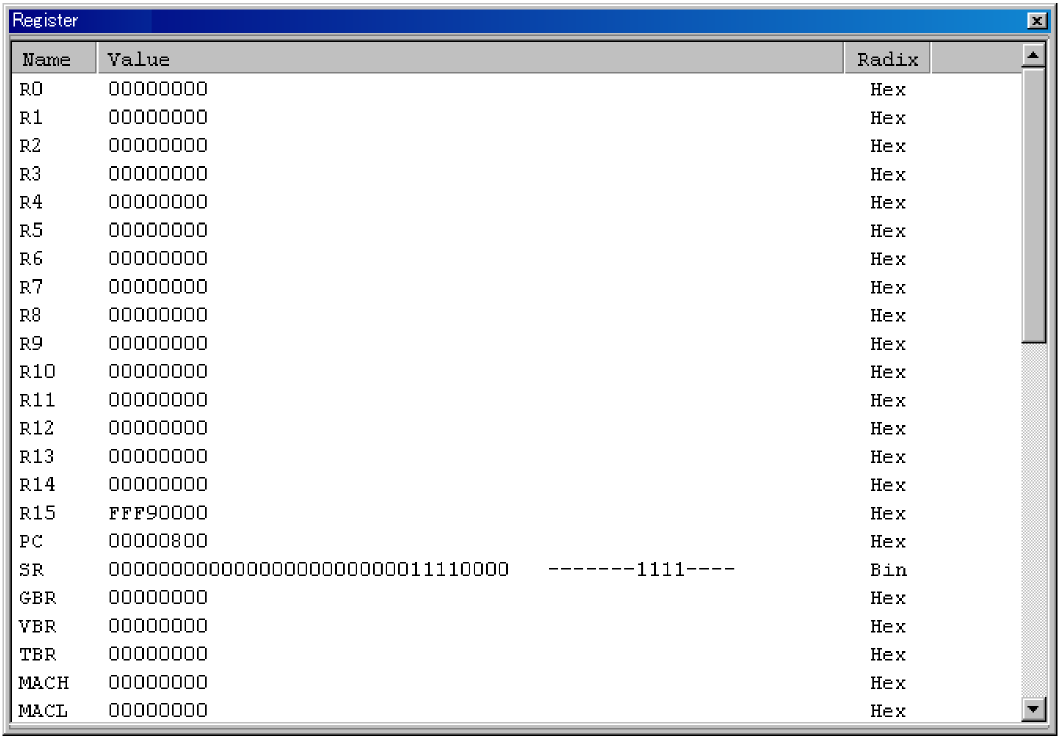


Figure 7.11 [Register] Window

- To change the value of the program counter (PC), double-click the value area in the [Register] window with the mouse. The following dialog box is then displayed, and the value can be changed. For the tutorial program, set the program counters to H'00000800 on the CPU0 side and H'00100800 on the CPU1 side, and then click on the [OK] button.

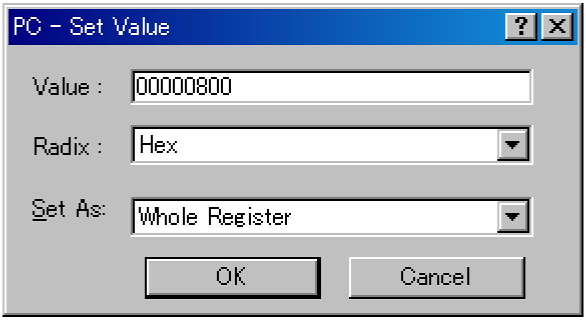


Figure 7.12 [Register] Dialog Box (PC)

- Change the value of the stack pointer (SP) in the same way. For the tutorial program, set the stack pointers to H'00010000 on the CPU0 side and H'00110000 on the CPU1 side.
  When using the MCU with flash memory, specify the end address of the internal RAM for the stack pointer (SP). The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.

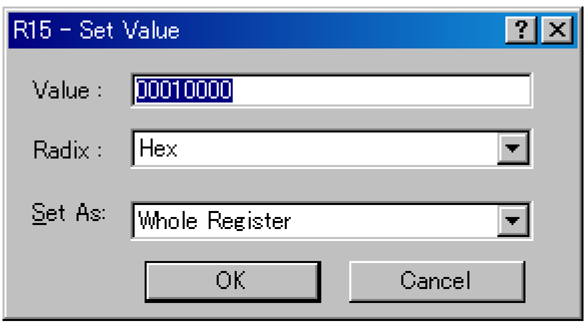


Figure 7.13 [Register] Dialog Box (R15)

7.10 Executing the Program

Execute the program as described in the following:

- Since synchronized execution has been enabled in the [Synchronized debugging] dialog box (by the [All debuggers synchronized] checkbox and the [Go] checkbox under [Synchronization options]), execute the program by selecting [Go] from the [Debug] menu of the High-performance Embedded Workshop for either core, and then selecting [Go] from the [Debug] menu or selecting the [Go] button on the toolbar. For this tutorial, start execution by selecting [Go] in the High-performance Embedded Workshop for CPU0.

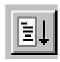


Figure 7.14 [Go] Button

Once program execution has started, '**RUNNING' will be displayed on the status bar, along with the address of the instruction being executed (PC value) and the value of the status register (SR) in the case of products that support acquisition of the CPU state. The program will be executed up to any breakpoint that was set in the High-performance Embedded Workshop for CPU0. Since synchronized breaking has been selected (by the [All debuggers synchronized] checkbox and the [Go] checkbox under [Synchronization options]) in the High-performance Embedded Workshop for CPU1, execution by CPU0 will break at the same time. An arrow will be displayed in the [S/W breakpoint] column of each High-performance Embedded Workshop to indicate the positions where execution of the programs on the respective CPUs was suspended. The messages [BREAKPOINT] and [SYNCHRONIZATION BREAK CPU#0] will appear in the status bars of the High-performance Embedded Workshops for CPU0 and CPU1, respectively.

Notes: 1. When the source file is displayed after a break, a path of the source file may be inquired. The location of the source file is as follows:
<Drive where the OS has been installed>:
\WorkSpace\Tutorial\E10A-USBM\SH4AM\SH4AM\Tutorial\CPU0\source (**** is the user name).

2. If program execution is failed, select [Reset CPU] from the [Debug] menu, reset the device, and restart the procedure from figure 7.8.

Figure 7.15 [Editor] Window (Break State)

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

- Select [Status] from the [CPU] submenu of the [View] menu. After the [Status] window is displayed, open the [Platform] sheet, and check the `Status` of `Cause of last break`.

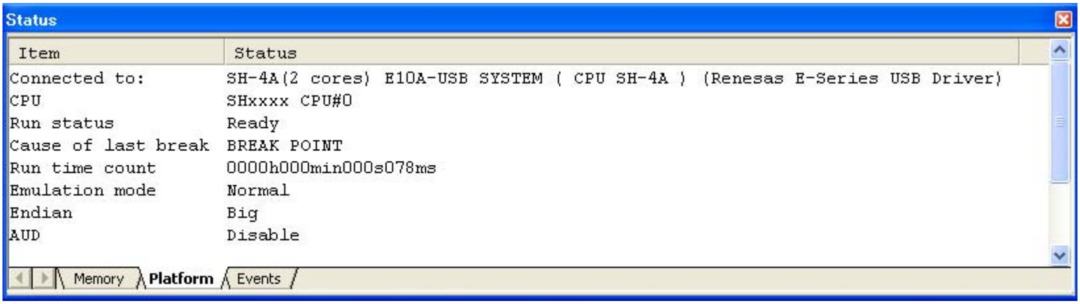


Figure 7.16 [Status] Window

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

7.11 Reviewing Breakpoints

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

- Select [Eventpoints] from the [Code] submenu of the [View] menu of the High-performance Embedded Workshop for CPU0. The [Event] window is displayed. Select the [Breakpoint] sheet.

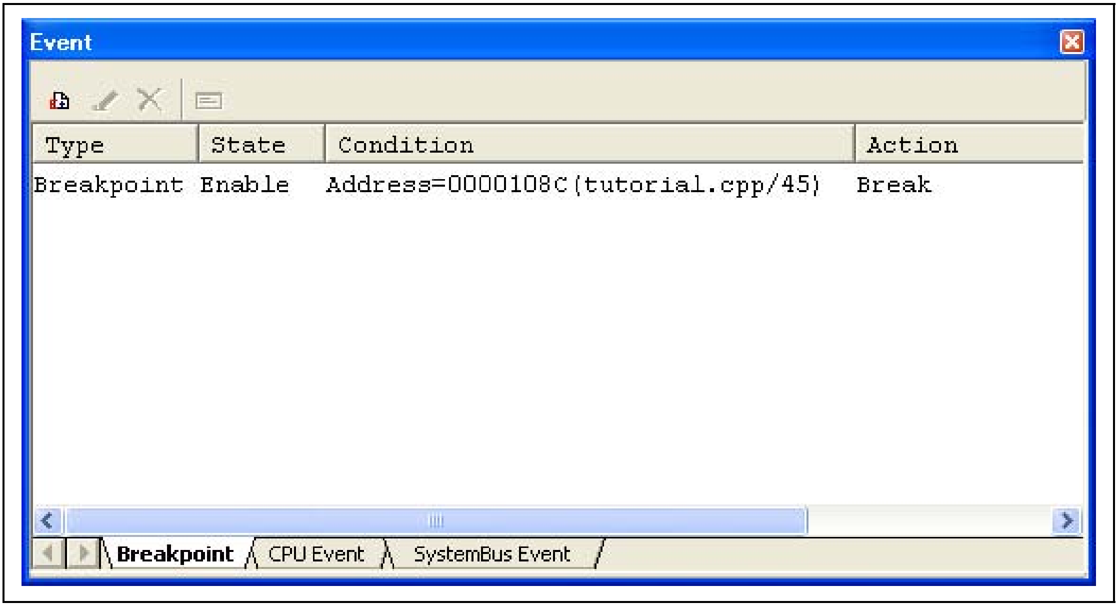


Figure 7.17 [Event] Window

The popup menu, opened by clicking the [Event] window with the right-hand mouse button, allows the user to set or change breakpoints, define new breakpoints, and delete, enable, or disable breakpoints.

7.12 Referring to Symbols

The [Label] window can be used to display the information on symbols in modules.

Select [Label] from the [Symbol] submenu of the [View] menu of the High-performance Embedded Workshop for CPU1. The [Label] window is displayed so that the user can refer to the addresses of symbols in modules.

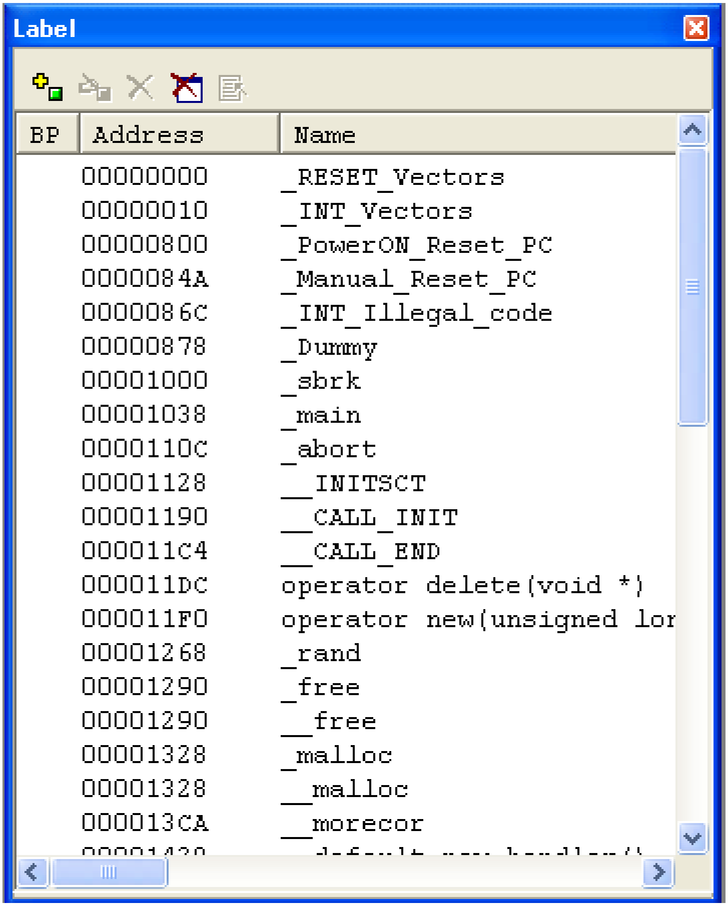


Figure 7.18 [Label] Window

7.13 Viewing Memory

When the label name is specified, the user can view the memory contents that the label has been registered in the [Memory] window. For example, to view the memory contents corresponding to `_main` in word size:

- Select [Memory …] from the [CPU] submenu of the [View] menu in the High-performance Embedded Workshop for CPU0, enter ***`_main`*** in the [Display Address] edit box, ***`00000000`*** in the [Scroll Start Address] edit box, and ***`FFFFFFFF`*** in the [Scroll End Address] edit box.

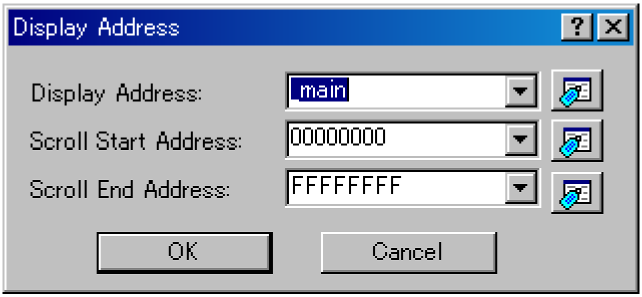


Figure 7.19 [Display Address] Dialog Box

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

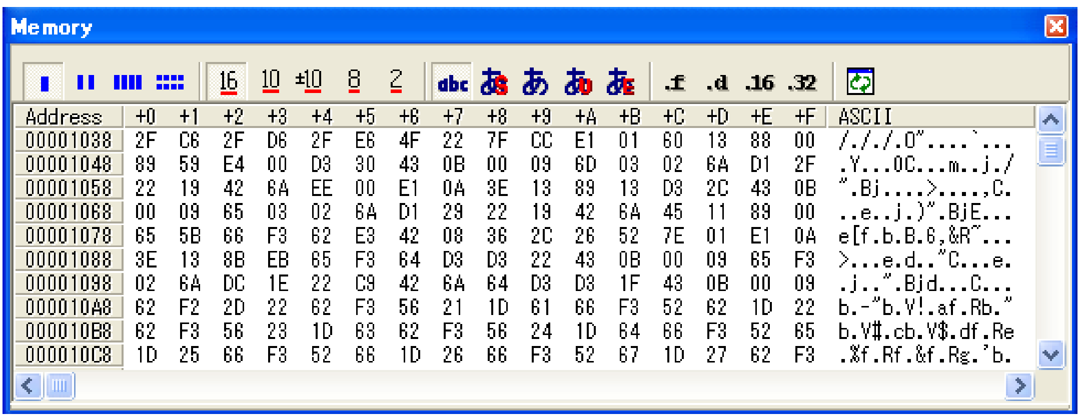


Figure 7.20 [Memory] Window

7.14 Watching Variables

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

- Place the cursor in the column to the left of where array a is displayed in the [Editor] window of the High-performance Embedded Workshop for CPU0.
- Click the right-hand mouse button and select [Instant Watch...].

The following dialog box will be displayed.

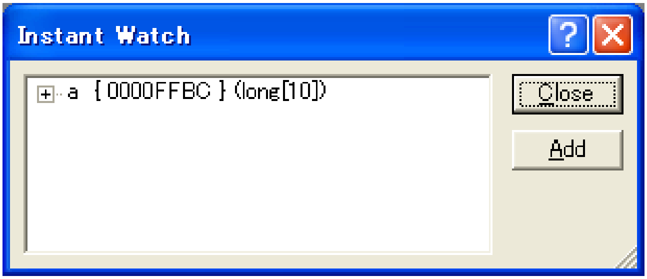


Figure 7.21 [Instant Watch] Dialog Box

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

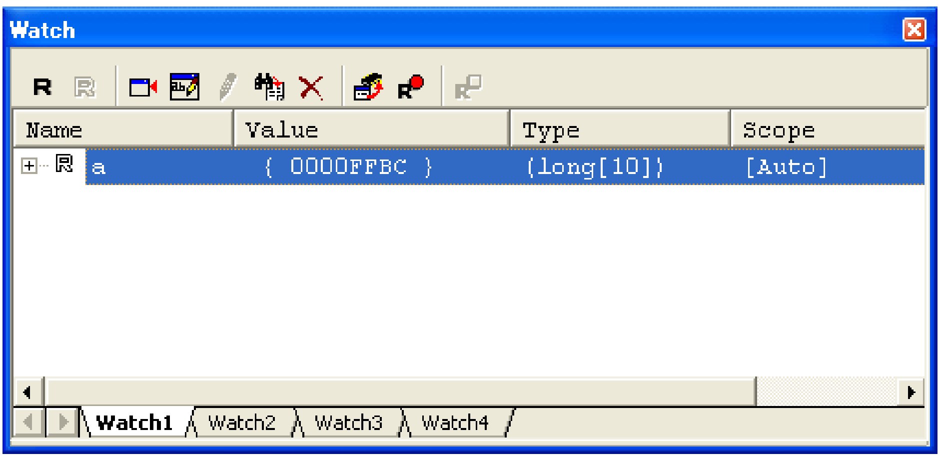


Figure 7.22 [Watch] Window (Displaying the Array)

The user can also add variables to the [Watch] window by specifying those name.

- Click the [Watch] window with the right-hand mouse button and select [Add Watch…] from the popup menu.

The following dialog box will be displayed. Enter variable `p_sam`.

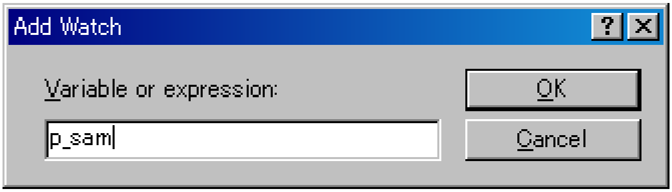


Figure 7.23 [Add Watch] Dialog Box

- Click the [OK] button.

The [Watch] window will now also show the instance `p_sam`.

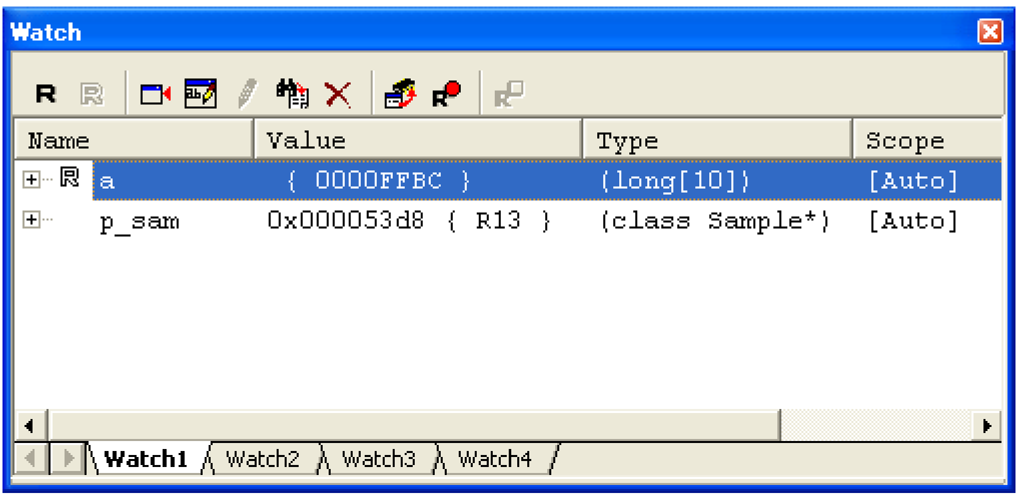


Figure 7.24 [Watch] Window (Displaying the Variables)

The user can click mark '+' at the left side of array `a` in the [Watch] window to watch all the elements.

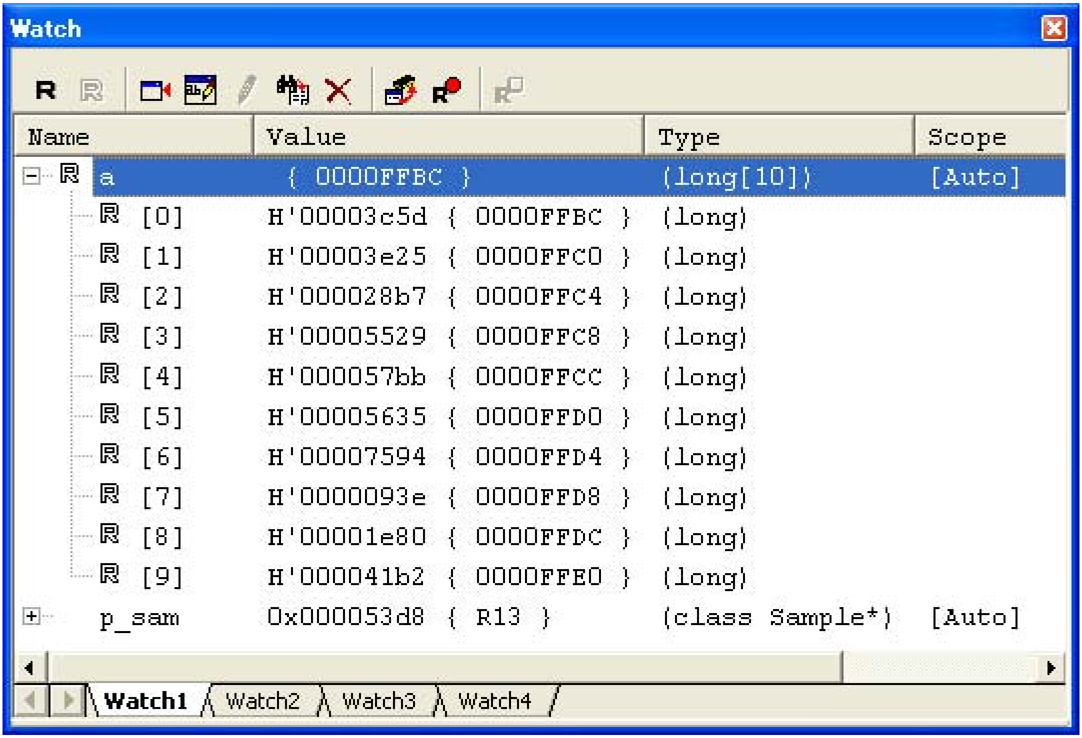


Figure 7.25 [Watch] Window (Displaying Array Elements)

7.15 Displaying Local Variables

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

- Select [Locals] from the [Symbol] submenu of the [View] menu of the High-performance Embedded Workshop for CPU0. The [Locals] window is displayed.

The [Locals] window shows the local variables in the function currently pointed to by the program counter, along with their values. Note, however, that the [Locals] window is initially empty because local variables are yet to be declared.

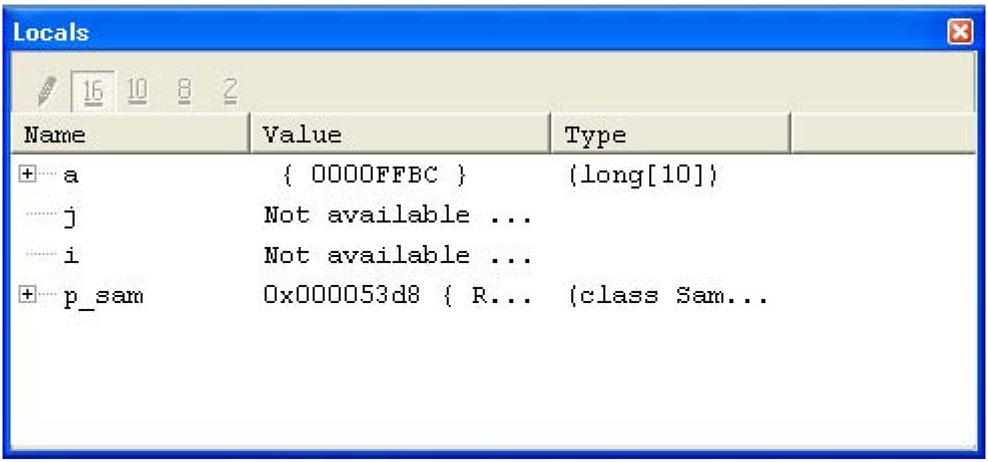


Figure 7.26 [Locals] Window

- Click mark '+' at the left side of array `a` in the [Locals] window to display the elements.
- Refer to the elements of array `a` before and after the execution of the `sort` function, and confirm that random data is sorted in descending order.

7.16 Stepping Through a Program

The High-performance Embedded Workshop provides a range of step menu commands that allow efficient program debugging.

Table 7.1 Step Option

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

7.16.1 Executing [Step In] Command

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

- To step through the `sort` function, select [Step In] from the [Debug] menu in the High-performance Embedded Workshop for CPU0, or click the [Step In] button on the toolbar. Since synchronized stepping has been enabled in the [Synchronized debugging] dialog box (the [Step] check box of [Synchronized debugging functions] under [All debugging synchronization] has been selected), these operations will lead to synchronized stepping in.

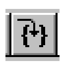


Figure 7.27 [Step In] Button

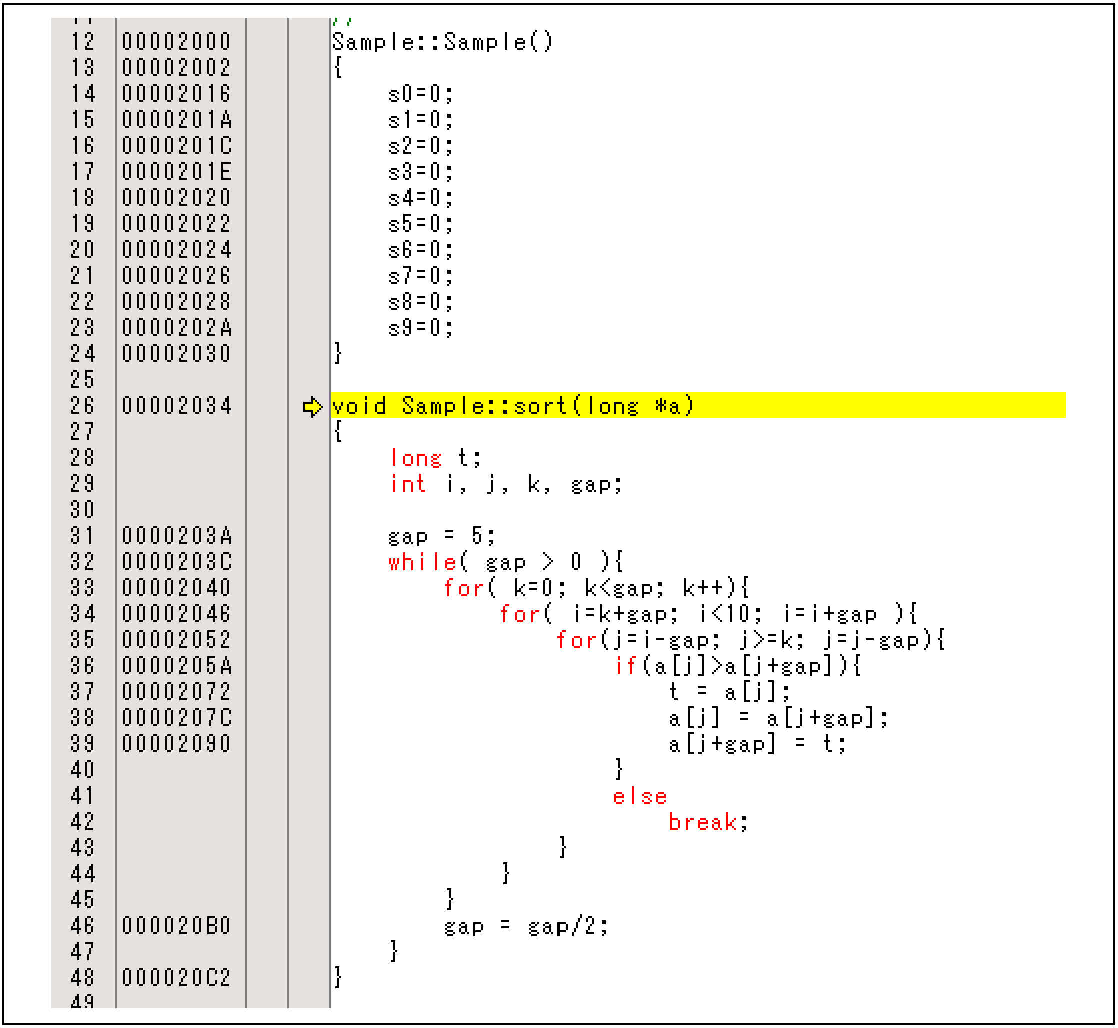


Figure 7.28 [Editor] Window (Step In)

- The highlighted line moves to the first statement of the `sort` function in the [Editor] window in the High-performance Embedded Workshop for CPU0.

7.16.2 Executing [Step Out] Command

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

To step out of the `sort` function, select [Step Out] from the [Debug] menu in the High-performance Embedded Workshop for CPU0, or click the [Step Out] button on the toolbar.

Note: It takes time to execute this function. When the calling source is clarified, use [Go To Cursor].

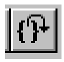


Figure 7.29 [Step Out] Button

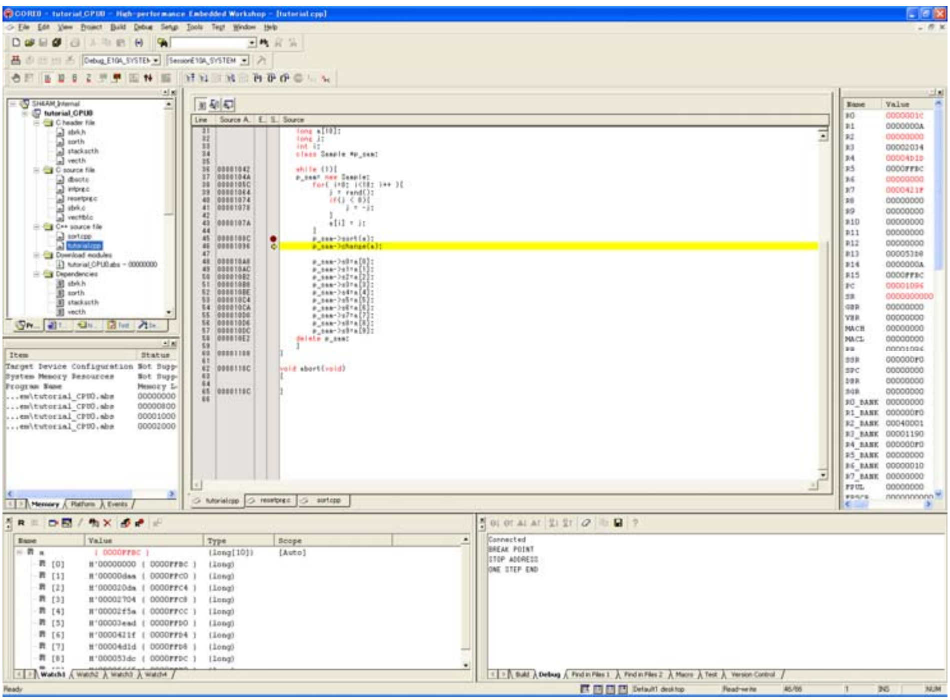


Figure 7.30 [High-performance Embedded Workshop] Window (Step Out)

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

Stepping out on the CPU1 side might not be completed. Whether it is or is not depends on the location in the source code when synchronization of High-performance Embedded Workshop for CPU1 is started. In such cases, complete stepping out by selecting the [STOP] button on the toolbar.

7.16.3 Executing [Step Over] Command

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

- Move to the `change` function following the procedures described in section 7.16.1, Executing [Step In] Command.
- To step through all statements in the `change` function at a single step, select [Step Over] from the [Debug] menu of the High-performance Embedded Workshop for CPU0, or click the [Step Over] button on the toolbar.

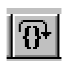


Figure 7.31 [Step Over] Button

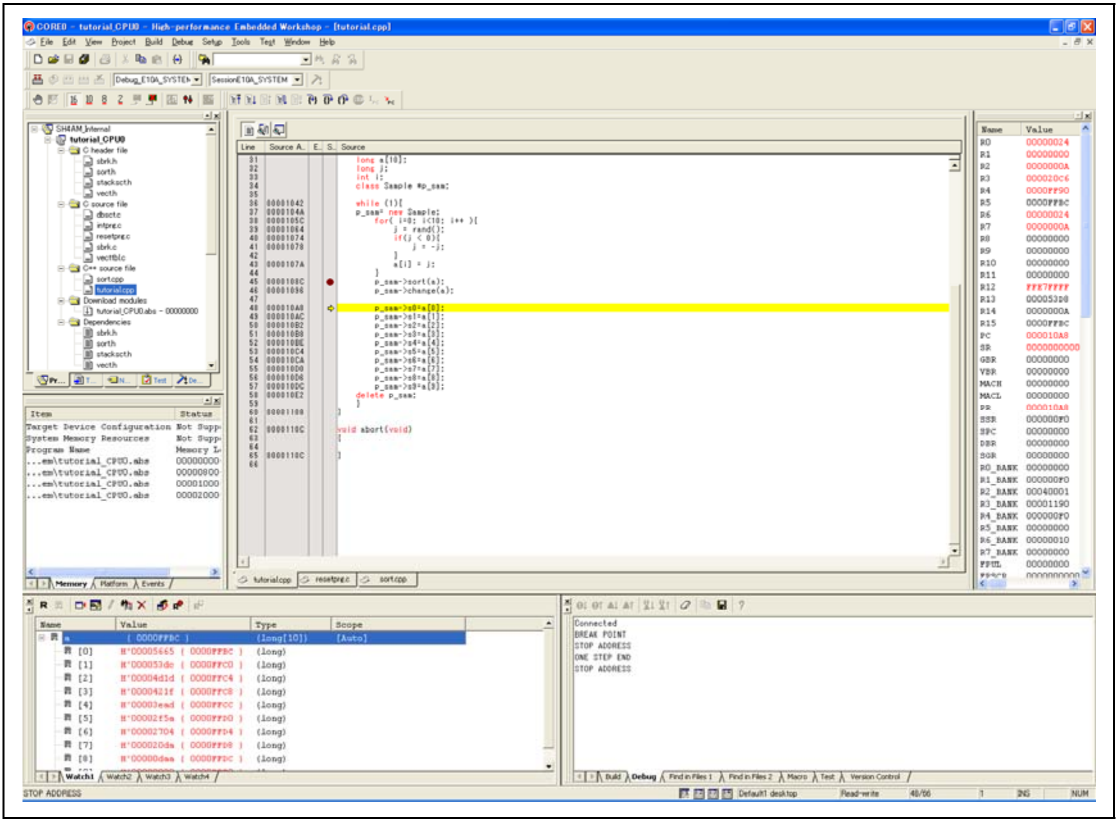


Figure 7.32 [High-performance Embedded Workshop] Window (Step Over)

7.17 Forced Breaking of Program Executions

The High-performance Embedded Workshop can force a break in the execution of a program.

- Cancel all breaks.
- To execute the remaining sections of the `main` function, select [Go] from the [Debug] menu in the High-performance Embedded Workshop for CPU0, or the [Go] button on the toolbar.

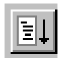


Figure 7.33 [Go] Button

- The program goes into an endless loop. To force a break in execution, select [Halt] from the [Debug] menu of the High-performance Embedded Workshop for CPU0, or click on the [STOP] button on the toolbar. Since synchronized breaking has been selected (by the [All debuggers synchronized] checkbox and the [Go] checkbox under [Synchronization options]) in the High-performance Embedded Workshop for CPU0, execution by CPU0 will break at the same time as execution by CPU1.

Figure 7.34 [STOP] Button

7.18 Break Function

The emulator has PC and hardware break functions. With the High-performance Embedded Workshop, a PC breakpoint can be set using the [Breakpoint] sheet of the [Event] window, and a hardware break condition can be set using the [Event condition] sheet.

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

7.18.1 PC Break Function

The emulator can set up to 255 PC breakpoints. Other methods for setting a PC breakpoint than in section 7.8, Setting a PC Breakpoint, are described below.

- Select [Eventpoints] from the [Code] submenu of the [View] menu in the High-performance Embedded Workshop for CPU0. The [Event] window is displayed.
- Select the [Breakpoint] sheet.

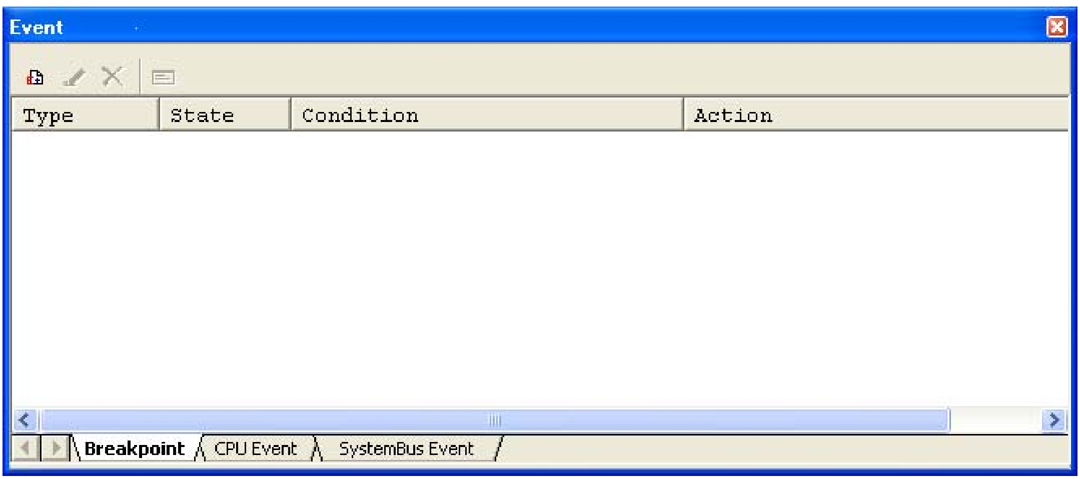


Figure 7.35 [Event] Window (Before PC Breakpoint Setting)

- Click the [Event] window with the right-hand mouse button and select [Add…] from the popup menu.
- Enter ***H'000010A8*** in the [Address] edit box.

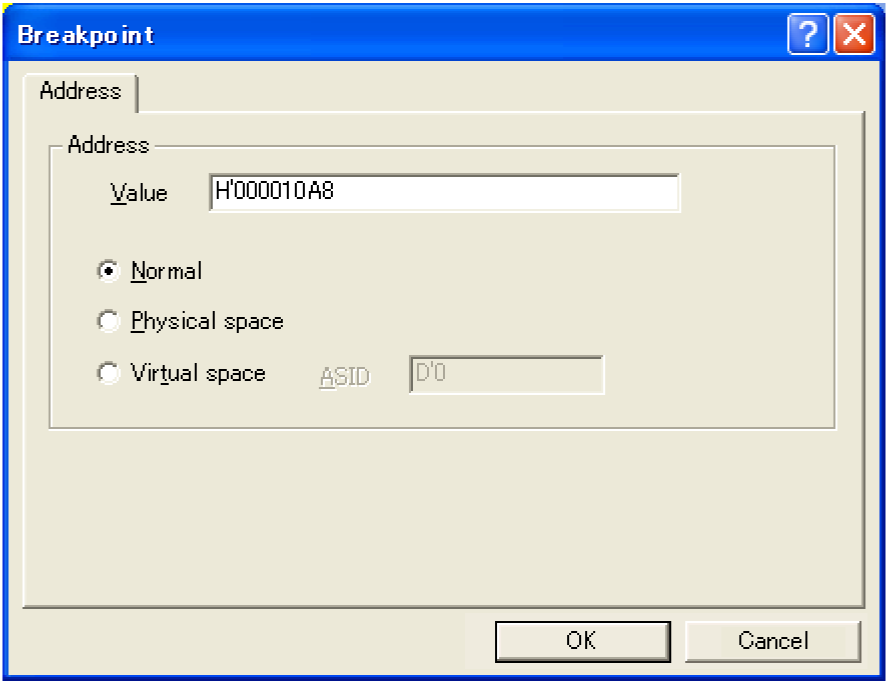


Figure 7.36 [Breakpoint] Dialog Box

Note: This dialog box differs according to the product. For the items of each product, refer to the online help.

- Click the [OK] button.

The PC breakpoint that has been set is displayed in the [Event] window.

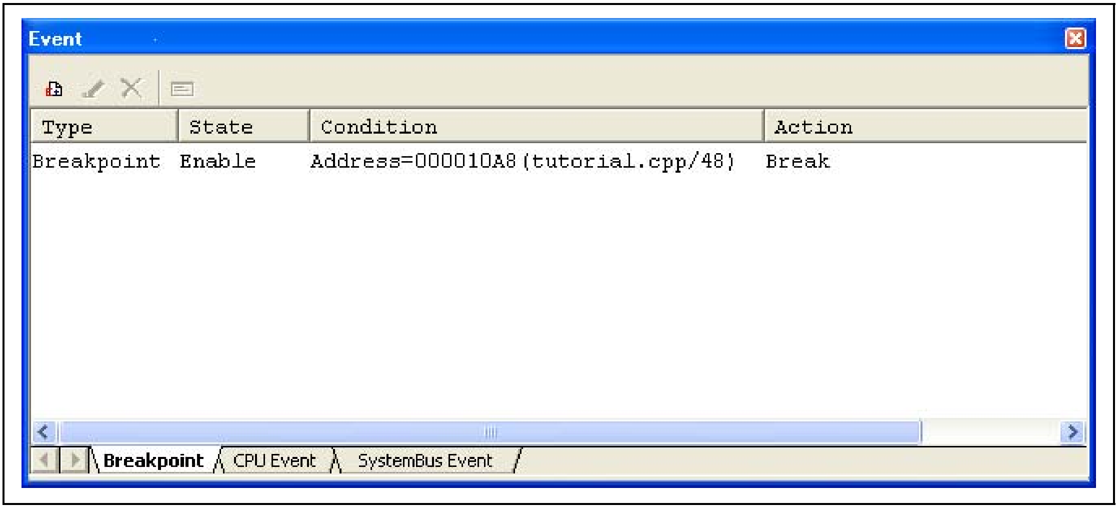


Figure 7.37 [Event] Window (PC Breakpoint Setting)

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

To stop the tutorial program at the PC breakpoint, the following procedure must be executed:

- Set the program counter and stack pointer values (CPU0: PC = H'00000800, R15 = H'00010000 and CPU1: PC = H'00100800, R15 = H'00110000) that were set in section 6.8, Setting Registers, in the [Register] window of the High-performance Embedded Workshops for CPU0 or CPU1. Click the [Go] button in the High-performance Embedded Workshops for either CPU0 or CPU1.

- If program execution is failed, reset the device and execute again the procedures above.

The program runs, and stops at the set PC breakpoint.

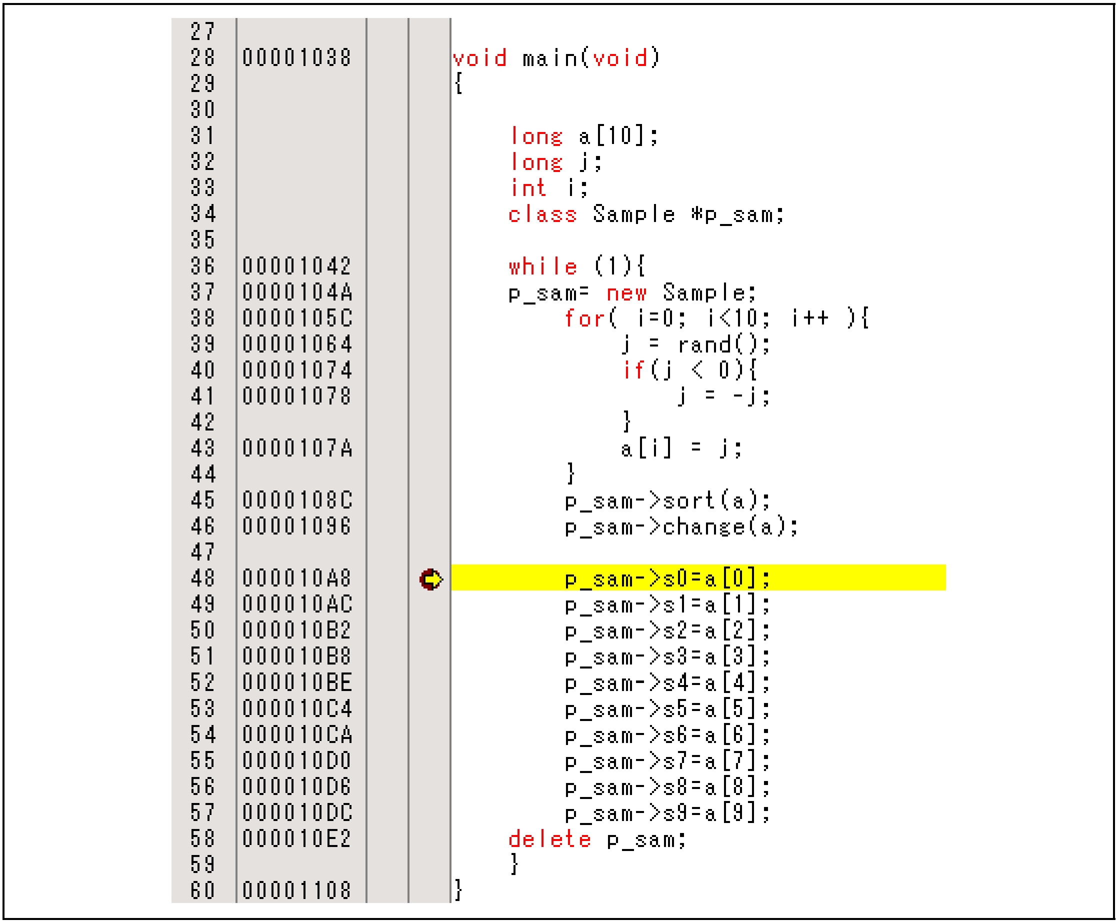


Figure 7.38 [Editor] Window at Execution Stop (PC Break)

The [Status] window displays the following contents.

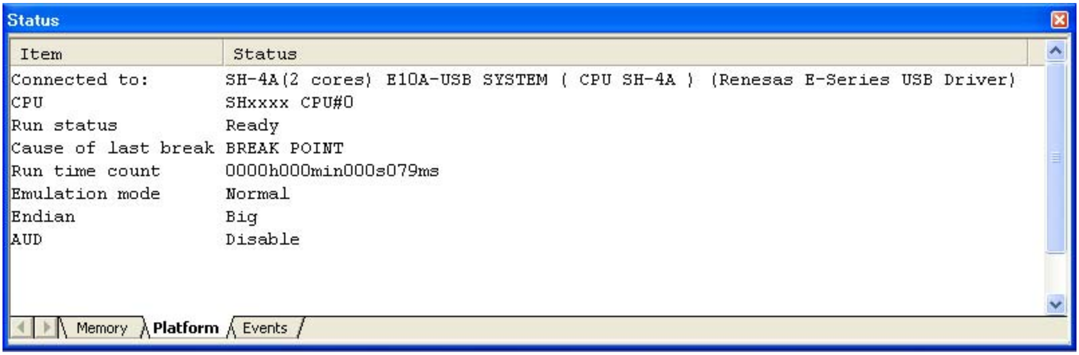


Figure 7.39 Displayed Contents of the [Status] Window (PC Break)

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

7.19 Hardware Break Function

A method is given below in which the address bus condition is set under Ch1 (IA_OA) as hardware break conditions.

- Select [Eventpoints] from the [Code] submenu of the [View] menu of the High-performance Embedded Workshop for CPU0. The [Event] window is displayed.
- The PC breakpoint that has been previously set is deleted. Click the [Event] window with the right-hand mouse button and select [Delete All] from the popup menu to cancel all PC breakpoints that have been set.
- To set a Ch1 (IA_OA), click the [CPU Event] tab.

Up to ten event points can be independently set as event conditions for the hardware break conditions on each CPU. In this example, set the hardware break condition for Ch1 (IA_OA).

Note: The number of hardware break conditions differs according to the product. For the number that can be specified for each product, refer to the online help.

- Select a line of Ch1 (IA_OA) in the [Event] window. When highlighted, double-click this line.

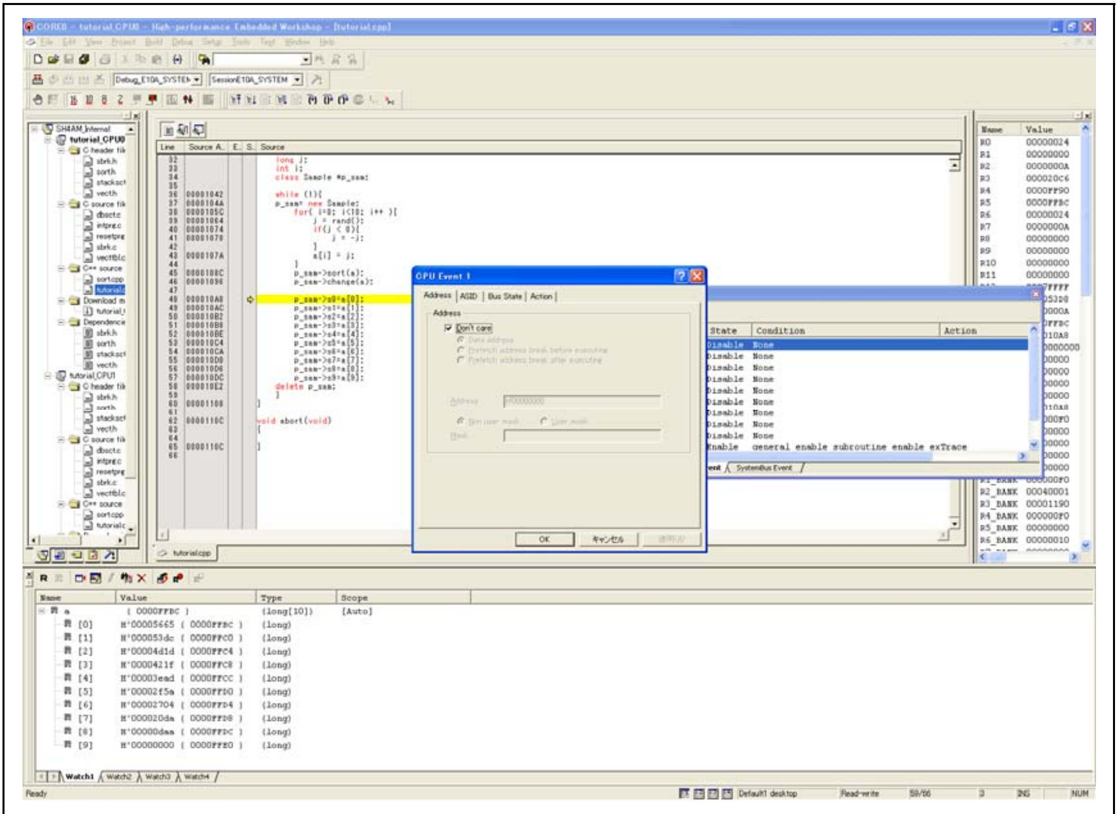


Figure 7.40 [High-performance Embedded Workshop] Window ([Ch1 (IA_OA])

- The [Ch1 (IA_OA)] dialog box is displayed.
- Clear the [Don't care] check box in the [Address] page.
- Select the [Prefetch address break before executing] radio button and enter ***H'0000108C*** as the value in the [Address] edit box.

Figure 7.41 [Address] Page ([CPU Event 1] Dialog Box)

Note: The items that can be set in this dialog box differ according to the product. For details on the settings for each product, refer to the online help.

- Click the [OK] button.
- The first point display in the State line changes from `Disable` to `Enable`.
- The first point display in the Condition line changes from `None` to `Address = H’0000108C (tutorial.cpp/45).`
- The first point display in the Action line displays as `Break).`
- Set the program counter and stack pointer values (CPU0: PC = H’00000800, R15 = H'00010000 and CPU1: PC = H’00100800, R15 = H'00110000) that were set in section 7.9, Setting Registers, in the [Register] window of the High-performance Embedded Workshops for CPU0 and CPU1. Click on the [Go] button of the High-performance Embedded Workshop for either CPU0 or CPU1.
  The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.
- If program execution is failed, reset the device and execute again the procedures above.

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

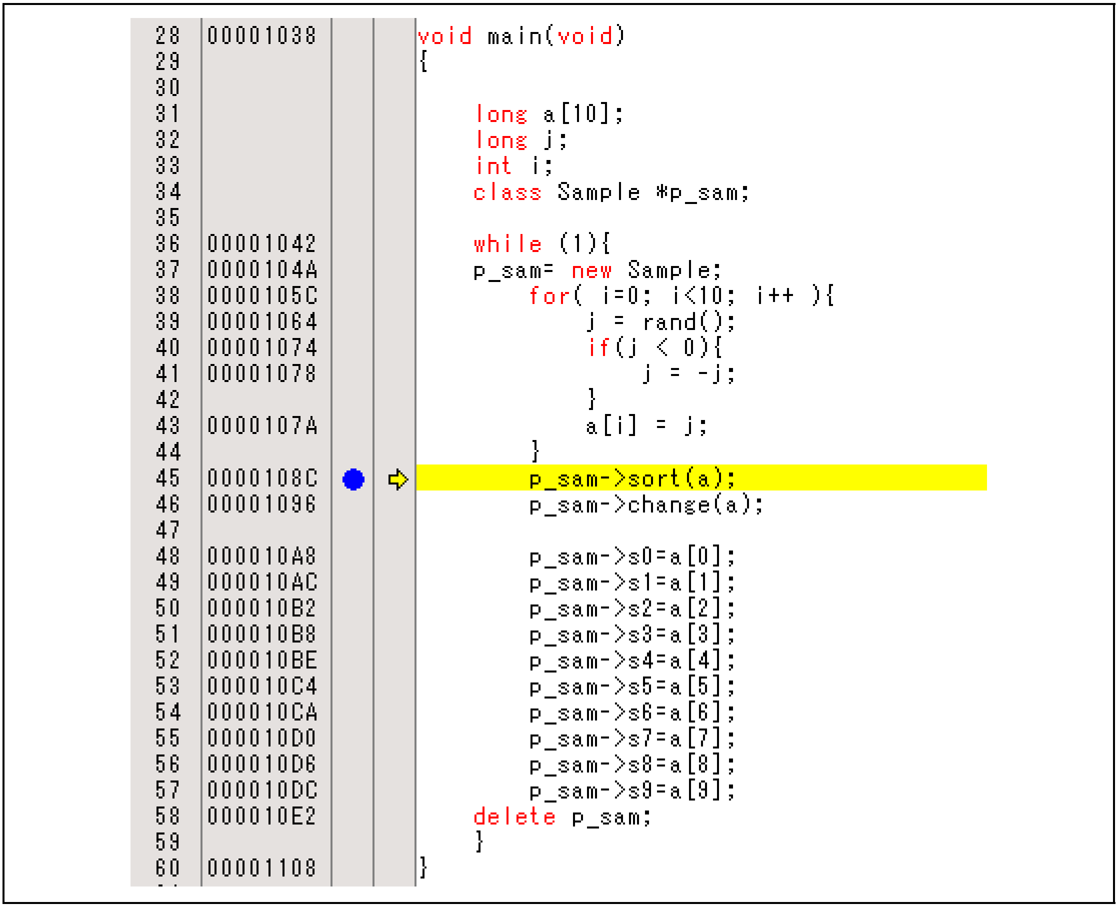


Figure 7.42 [Editor] Window at Execution Stop ([Ch1 (IA_OA)])

The [Status] window displays the following contents.

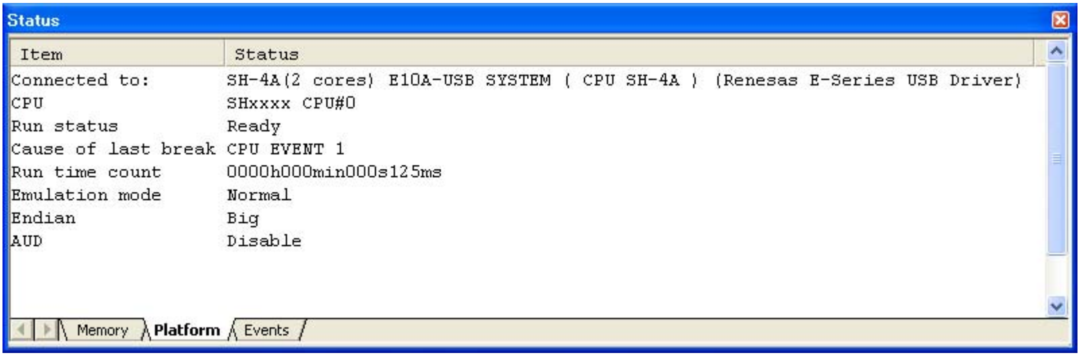


Figure 7.43 Displayed Contents of the [Status] Window ([Ch1 (IA_OA])

Note: The items that can be displayed in this window differ according to the product. For the items that can be displayed, refer to the online help.

7.20 Trace Functions

The emulator has three branch-instruction trace functions.
Trace function can acquired the event shown in below.

(a) Branch generation information

The branch source and branch destination addresses are acquired.

(b) Memory access information within the specified range

Memory access in the specified range can be acquired by trace.

Two memory ranges can be specified for channels Ch5(OA) or Ch6(OA). The read, write, or read/write cycle can be selected as the bus cycle for trace acquisition.

This function is called the window trace function.

(c) Software trace

When a specific instruction is executed, the PC value at execution and the contents of one general register are acquired by trace. Describe the Trace(x) function (x is a variable name) to be compiled and linked beforehand. For details, refer to the SHC/C++ compiler manual.

When the load module is loaded on the emulator and a valid software trace function is executed, the PC value that has executed the Trace(x) function, the variable for x, and the source lines are displayed.

7.20.1 Internal Trace Function

This function is achieved by using the MPU’s internal trace buffer.

Notes: 1. The number of branch instructions that can be acquired by a trace differs according to the product. For the number that can be specified for each product, refer to the online help.

2. The internal trace function is not supported for all products. For details on the specifications of each product, refer to the online help.

3. The internal trace function is not extended for all products. For details on the specifications of each product, refer to the online help.

7.20.2 AUD Trace Function

This is the large-capacity trace function that is enabled when the AUD pin is connected to the emulator. When an event for which trace information is to be acquired occurs, trace information is output through the AUD pins in realtime. When each pair of a branch source and branch destination instruction is treated as a unit, the maximum number of events for which trace information can be acquired is 1,048,544. The number displayed in a given trace window is a maximum of 65,535.

(1) Trace acquisition mode

The AUD trace function has the following modes to acquire a trace.

Table 7.2 shows the AUD trace acquisition mode that can be set in each trace function.

Table 7.2 AUD Trace Acquisition Mode

Type	Mode	Description
Continuous trace occurs	Realtime trace	When the trace information is being generated intensely that the output from the AUD pin incapable of keeping up, the CPU temporarily suspends the output of trace information. Therefore, although the user program is run in real time, the acquisition of some trace information might not be possible.
	Non realtime trace	When trace information is being generated so intensely that the output from the AUD pin is incapable of keeping up, CPU operations are temporarily suspended and the output of trace information takes priority. In such cases, the realtime characteristics of the user program are lost.
Trace buffer full	Trace continue	This function overwrites the latest trace information to store the oldest trace information.
	Trace stop	After the trace buffer becomes full, the trace information is no longer acquired. The user program is continuously executed.

(2) Trace display contents

When the program breaks, the following trace results are displayed in the [Trace] window.

- PTR: The trace-buffer pointer (+0 from the last instruction to have been executed)
- IP: Indicates the number of cycles that have elapsed since the latest trace information was gathered. For branch instructions, the branch source and destination are counted together as one.
- Master: Type of bus master that accessed the memory.
- Type: Displays the type of trace acquisition information.
- Branch Type: Branch type (only displayed for a branch trace)
  For an AUD trace, this item is only displayed if the PPC option has been enabled.
- Bus: Displays which bus was accessed.
- R/W: Displays whether the access involved reading or writing.
- Address: Displays the addresses from which the trace data was acquired.
- Data: Displays the data acquired in the trace.
- PPC: Output from a performance counter
- Instruction, Source, Label: Displays the mnemonic of the instruction at the trace acquisition address, along with the corresponding source code and label information. Double-clicking on the [Source] column moves the cursor to the corresponding position in the [Editor] window.

The Type, BUS, R/W, Address, and Data columns have different meanings according to the type of AUD trace that has been selected.

Table 7.3 [Trace Window] Display Contents

Trace Type	Type Column	BUS Column	R/W Column	Address Column	Data Column
Branch trace	BRANCH*1	No display	No display	Branch source address*1	No display
	DESTINATION	No display	No display	Branch destination address	No display
Memory-range access trace	MEMORY	Bus through which access is proceeding	Read/write	Memory access address	Memory access data*1
Software trace	S_TRACE	No display	No display	Trace(x) function execution address	Variable x data
System bus trace	MEMORY	No display	Read/write	Memory access address	Memory access data (write only)*1
Data lost*2	LOST	No display	No display	No display	No display
CPU wait generation*2	CPU-WAIT	No display	No display	No display	No display

Notes: 1. Not displayed when the PPC option is in use.

2. According to the device being debugged, there may be no output for the [Lost] or [CPU-WAIT] type. In such a case, it is not possible to clarify whether the trace data was not output in time or the CPU generated a wait state for the output trace data.

7.20.3 Memory Output Trace Function

This function is used to write the trace result to the specified memory range.

The following is the procedure for setting the memory output trace function.

(1) Setting the trace acquisition mode

- Display the [Trace] window.
- Click the [Trace] window with the right-hand mouse button and select [Acquisition] from the popup menu to display the [Acquisition] dialog box.

The trace acquisition condition is set in the [Trace mode] page.

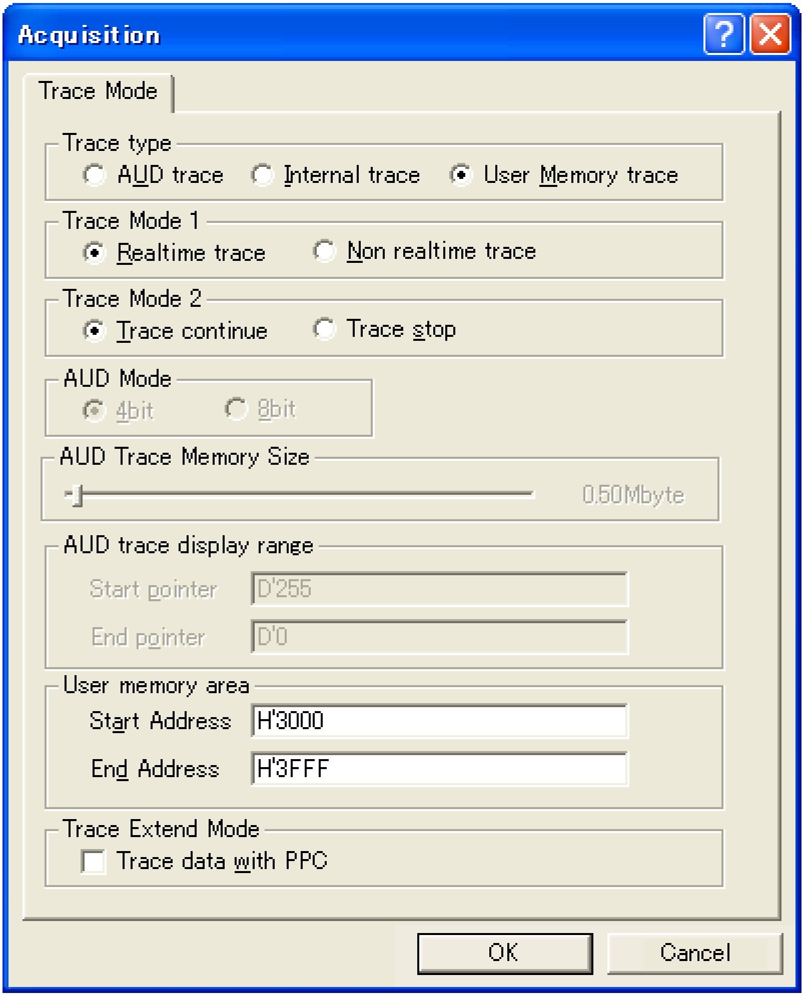


Figure 7.44 [Trace mode] Page

The following table shows the options.

Trace Acquisition Mode

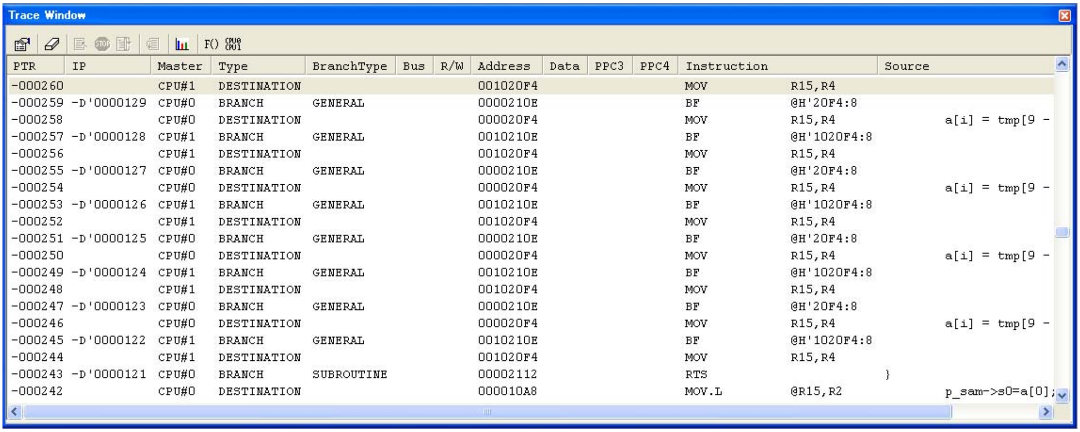
Type	Mode	Description
Continuous trace occurs	Realtime trace	When trace information is being generated so intensely that the output from the memory is incapable of keeping up, all the information may not be output. The user program can be executed in realtime, but some trace information will be lost.
	Non realtime trace	When trace information is being generated so intensely that the output from the memory is incapable of keeping up, the CPU stops operations until the trace information is output. The user program is not executed in realtime.
Trace buffer full	Trace continue	This function always overwrites the oldest trace information to acquire the latest trace information.
	Trace stop	When the trace buffer becomes full, the trace information is no longer acquired. The user program is continuously executed.

Note: The items that can be set in this window differ according to the product. For details on the settings for each product, refer to the online help.

(2) Displaying the trace result

- Set the addresses that where the range of memory for the output of results of tracing starts and ends in the [Start Address] and [End Address] edit boxes of [User Memory area], respectively.
- Enter the addresses depending on your environment. Do not specify the range that the program has been downloaded to, as the memory contents are overwritten by the trace output result.
- Run the program as shown in the example in section 7.18.1, PC Break Function. The trace results are displayed in the [Trace] window after program execution is completed.

The following is an example of the trace display.



Trace Window

PTR	IP	Master	Type	BranchType	Bus	R/W	Address	Data	PPC3	PPC4	Instruction	Source
-000260		CPU#1	DESTINATION				001020F4				MOV R15,R4	
-000259	-D'0000129	CPU#0	BRANCH	GENERAL			0000210E				BF @H'20F4:8	
-000258		CPU#0	DESTINATION				000020F4				MOV R15,R4	a[i] = tmp[9 -
-000257	-D'0000128	CPU#1	BRANCH	GENERAL			0010210E				BF @H'1020F4:8	
-000256		CPU#1	DESTINATION				001020F4				MOV R15,R4	
-000255	-D'0000127	CPU#0	BRANCH	GENERAL			0000210E				BF @H'20F4:8	
-000254		CPU#0	DESTINATION				000020F4				MOV R15,R4	a[i] = tmp[9 -
-000253	-D'0000126	CPU#1	BRANCH	GENERAL			0010210E				BF @H'1020F4:8	
-000252		CPU#1	DESTINATION				001020F4				MOV R15,R4	
-000251	-D'0000125	CPU#0	BRANCH	GENERAL			0000210E				BF @H'20F4:8	
-000250		CPU#0	DESTINATION				000020F4				MOV R15,R4	a[i] = tmp[9 -
-000249	-D'0000124	CPU#1	BRANCH	GENERAL			0010210E				BF @H'1020F4:8	
-000248		CPU#1	DESTINATION				001020F4				MOV R15,R4	
-000247	-D'0000123	CPU#0	BRANCH	GENERAL			0000210E				BF @H'20F4:8	
-000246		CPU#0	DESTINATION				000020F4				MOV R15,R4	a[i] = tmp[9 -
-000245	-D'0000122	CPU#1	BRANCH	GENERAL			0010210E				BF @H'1020F4:8	
-000244		CPU#1	DESTINATION				001020F4				MOV R15,R4	
-000243	-D'0000121	CPU#0	BRANCH	SUBROUTINE			00002112				RTS	}
-000242		CPU#0	DESTINATION				000010A8				MOV.L @R15,R2	p_sam->s0=a[0];

Figure 7.45 [Trace] Window (Example)

7.20.4 Useful Functions of the [Trace] Window

The trace window provides the following useful functions.

(1) Searches for the specified data.

(2) Extracts the specified data.

(3) Filters and displays again the specified data.

(4) Supplements the information from the branch destination address to the next branch source address.

For the usage of those functions, refer to section 5.7, Viewing the Trace Information.

(5) Changes the trace settings during user program execution.

In some devices to be debugged, trace settings can be changed during user program execution. For details on the specifications of each product, refer to the online help.]

7.21 MMU Support

This function can be used when the supported MPU has an MMU.

- TLB window

In the emulator, the contents of the TLB table can be easily displayed and edited by selecting [CPU -> TLB] from the [View] menu. For details, refer to the online help.

- VP_MAP translation function

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

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

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

Example:

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

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

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

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

Here, virtual addresses correspond to physical addresses as shown in figure 7.46.

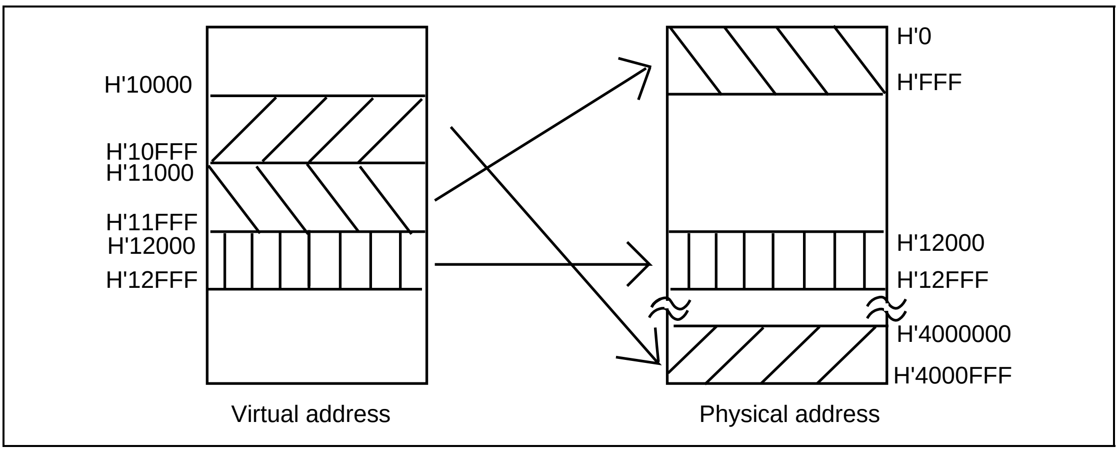


Figure 7.46 Address Translation according to VP_MAP Tables

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

- When the `Normal` radio button is selected:

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

  The address is not translated.
- When the `Virtual` radio button is selected:

  The address is translated according to the TLB. If the specified address is outside the TLB table settings, a TLB error will occur.

Table 7.4 Address Translation Tables

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

Note: Specified by the [Memory area] group box in the [Configuration] dialog box.

7.22 Stack Trace Function

The emulator uses the information on the stack to display the names of functions in the sequence of calls that led to the function to which the program counter is currently pointing.

Note: This function can be used only when the load module that has the Elf/Dwarf2-type debugging information is loaded. Such load modules are supported in SHC/C++ compiler (including OEM and bundle products) V6.0 or later.

- Double-click the [Event] column in the `sort` function in the High-performance Embedded Workshop for CPU0 and set an Event point.

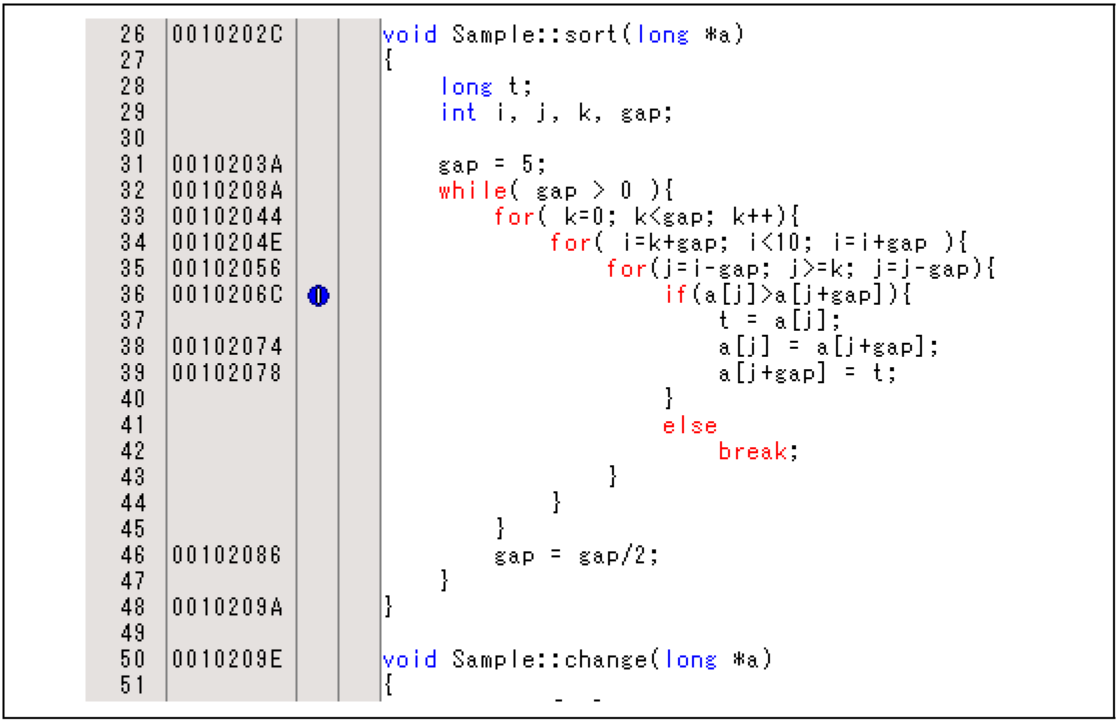


Figure 7.47 [Editor] Window (Hardware Break Setting)

- Set the same program counter and stack pointer values (CPU0: PC = H'00000800 and R15 = H'00010000, CPU1: PC = H'00100800 and R15 = H'00110000) as were set in section 7.9, Setting Registers (again, use the [Register] windows in the High-performance Embedded Workshops for CPU0 and CPU1). After that, click on the [Go] buttons in the High-performance Embedded Workshops for CPU0 and CPU1.

- If program execution is failed, reset the device and execute again the procedures above.
- After the break in program execution, select [Stack Trace] from the [Code] submenu of the [View] menu to open the [Stack Trace] window.

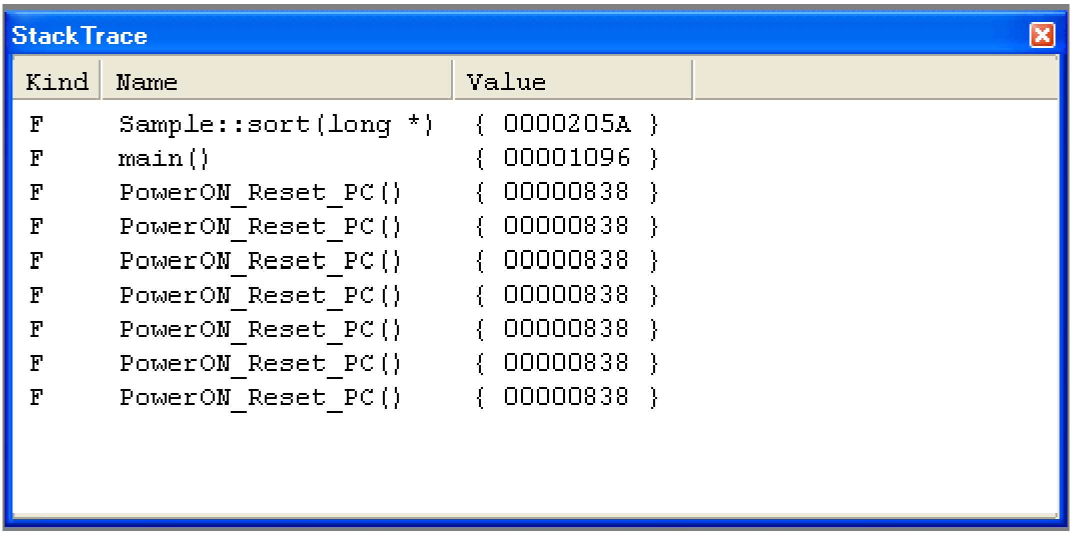


Figure 7.48 [Stack Trace] Window

Figure 7.48 shows that the position of the program counter is currently at the selected line of the `sort()` function, and that the `sort()` function is called from the `main()` function.

To remove the hardware break, double-click the [Event] column in the `sort` function again.

Note: For details on this function, refer to the online help.

7.23 Performance Measurement Function

The emulator has performance measurement functions.

- Performance measurement function

  This function applies a counter in the MPU to measure the number of times various events have occurred and cycle count. A start and end condition for counting can be set.

  Various items that can be measured differ according to the supported MPU.

7.23.1 Performance Measurement Function

The following is an example of the use of a counter in the MPU to measure the number of times various events have occurred and cycle count.

(1) Setting method

Select [Performance Analysis] from the [Performance] submenu of the [View] menu of the High-performance Embedded Workshop for CPU0.
When the [Select Performance Analysis Type] dialog box will open, click the [OK] button.

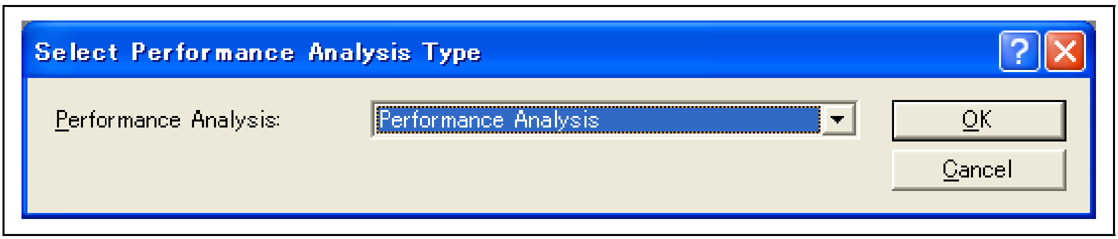


Figure 7.49 [Select Performance Analysis Type] Dialog Box

- The [Performance Analysis] window will be displayed.
- Place the mouse cursor anywhere within this window, click the right-hand mouse button, and then select [Set] from the popup menu. The [Performance Analysis] dialog box will open. The events to be measured and measuring conditions can be set in this dialog box.

Note: The items that can be displayed in this dialog box differ according to the product. For details on the settings for each product, refer to the online help.

After the conditions have been set, clicking the [OK] button and executing the user program will display the result of measurement in the [Performance Analysis] window.

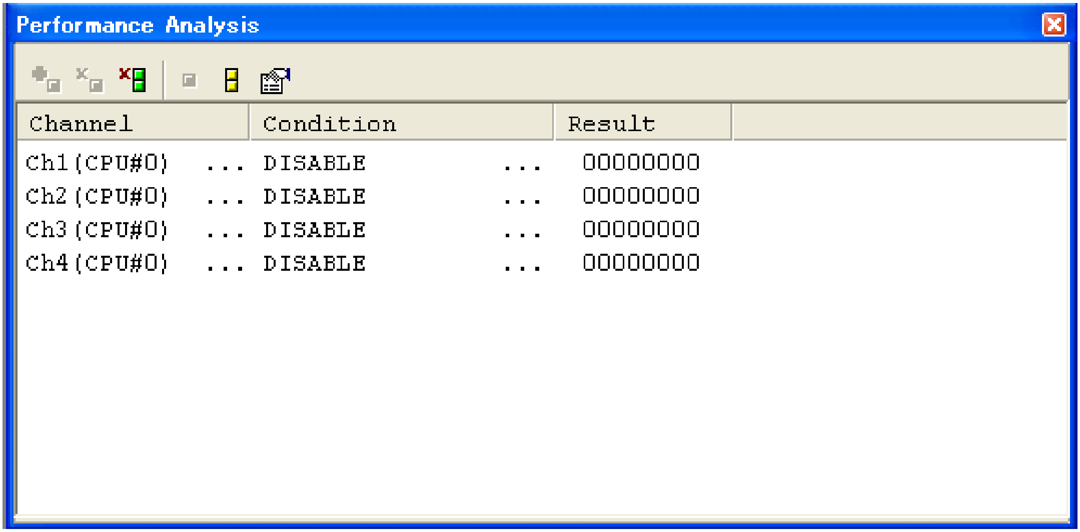


Figure 7.50 [Performance Analysis] Window

Note: The items that can be displayed in this window differ according to the product. For details on the settings for each product, refer to the online help.

7.24 Download Function to the Flash Memory Area

The emulator enables downloading to the external flash memory area. This function requires a program for programming the flash memory (hereinafter referred to as a write module), a program for erasing the flash memory (hereinafter referred to as an erase module), and the RAM area for downloading and executing these modules.

Notes: 1. The write and erase modules must be prepared by the user.

2. This function is not available depending on the MCU. For such an MCU, the [Loading flash memory] page shown in figure 7.51 will not be displayed.

- Interface with write and erase modules and emulator firmware

  The write and erase modules must be branched from the emulator firmware. To branch from the emulator firmware to the write and erase modules, or to return from the write and erase module to the emulator firmware, the following conditions must be observed:

  — Describe all the write and erase modules with the assembly language.

  — Save and return all the general register values and control register values before and after calling the write or erase module.

  — Return the write or erase module to the calling source after processing.

  — The write and erase module must be a Motorola-type file.

  The module interface must be as follows to pass correctly the information that is required for flash memory accessing.

Table 7.5 Module Interface

Module Name	Argument	Return Value
Write module	R4(L): Write address R5(L): Access size 0x4220 = byte, 0x5720 = word, 0x4C20 = longword R6(L): Write data	R0(L): End code Normal end = 0, Abnormal end = other than 0,
Erase module	R4(L): Access size 0x4220 = byte, 0x5720 = word, 0x4C20 = longword	None

Note: The (L) means the longword size.

Note: Write module: The write data for the access size is set to the R6 register. When the access size is word or byte, 0 is set to the upper bits of the R6 register.

- Flash memory download method

  For downloading to the flash memory, set the items on the [Loading flash memory] page in the [Configuration] dialog box, which is opened from [System…], then [Emulator] from the [Setup] menu.

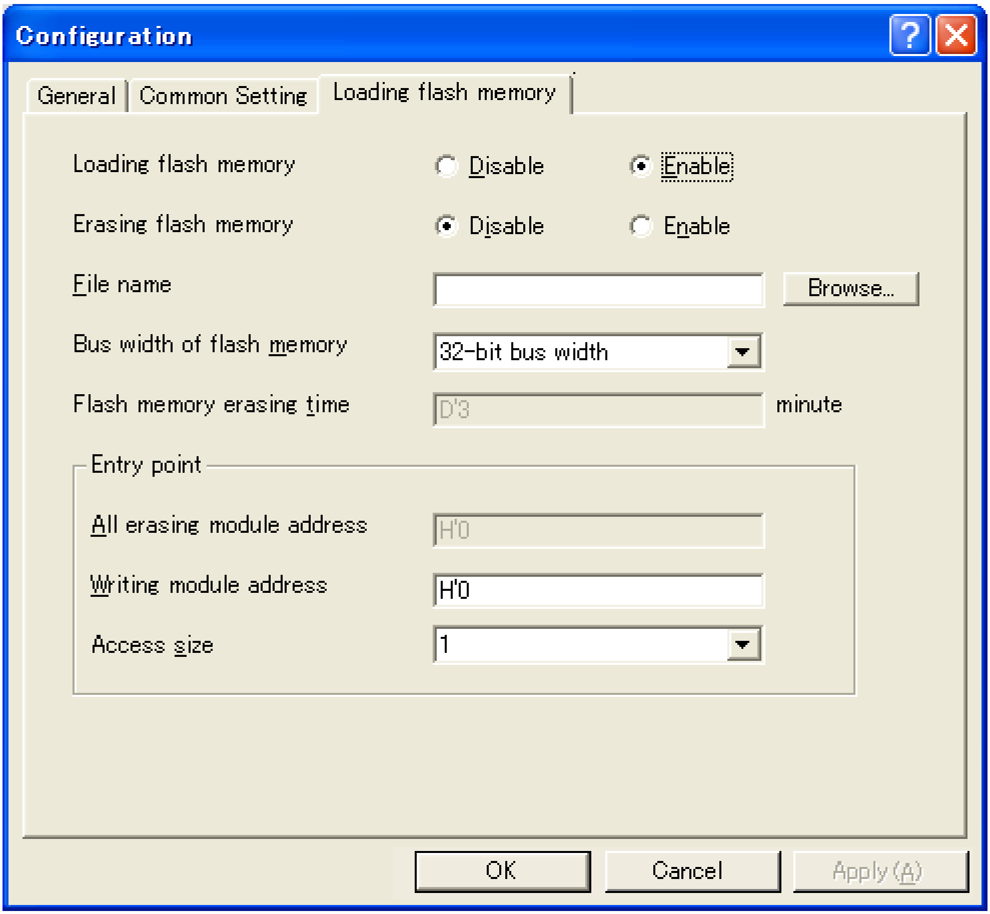


Figure 7.51 [Loading flash memory] Page

Table 7.6 shows the options for the [Loading flash memory] page.

Table 7.6 [Loading flash memory] Page Options

Option	Description
[Loading flash memory] radio button	Sets Enable for flash memory downloading. When Enable is selected, and [File load] is selected from the [File] menu for downloading, the write module is always called. Enable: Download to the flash memory Disable: Not download to the flash memory
[Erasing flash memory] radio button	Sets Enable for erasing before the flash memory is programmed. When Enable is selected, the erase module is called before calling the write module. Enable: Erase the flash memory Disable: Not erase the flash memory
[File name] edit box	Sets the file name of the S-type load module including the write and erase modules. The file that has been set is loaded to the RAM area before loading to the flash memory. A maximum of 128 characters can be input for the file name.
[Bus width of flash memory] list box	Sets the bus width of the flash memory.
[Flash memory erasing time] edit box*	Sets the TIMEOUT value for erasing the flash memory. Set a larger value if erasing requires much time; the default time is three minutes. The radix for the input value is decimal. It becomes hexadecimal by adding H'.
[Entry point] group box	Sets the calling destination address or access size of the write and erase modules. [All erasing module address] edit box: Inputs the calling destination address of the erase module. [Writing module address] edit box: Inputs the calling destination address of the write module. [Access size] combo box: Selects the access size of the RAM area where the write/erase module is loaded.

Note: Although the values that can be set are D'1 to D'65535, the TIMEOUT period may be extended according to the set value. Therefore, it is recommended to input the minimum value by considering the erasing time of the flash memory in use.

- Notes on using the flash memory download function

  The following are notes on downloading to the flash memory.

  — When the flash memory download is enabled, downloading to areas other than the flash memory area is disabled.

  — Downloading is only enabled to the flash memory area. Perform memory write or PC break only to the RAM area.

  — When the flash memory erase is enabled, the [Stop] button cannot stop erasing.

  — The area for the write and erase modules must be set in an MMU-disabled space.

- An example of downloading to the flash memory

  The following is an example of downloading to the flash memory manufactured by Intel Corporation (type number: G28F640J5-150). A sample is provided in the \Fmtool folder in the installation destination folder. Create a program that suits the user specifications by referring to this sample.

Table 7.7 Board Specifications

Item		Contents
SDRAM address		H'0C000000 to H'0FFFFFFF
Flash memory address		H'00000000 to H'01FFFFFF
Bus width of flash memory		32 bits
Operating environment	Endian	Big endian

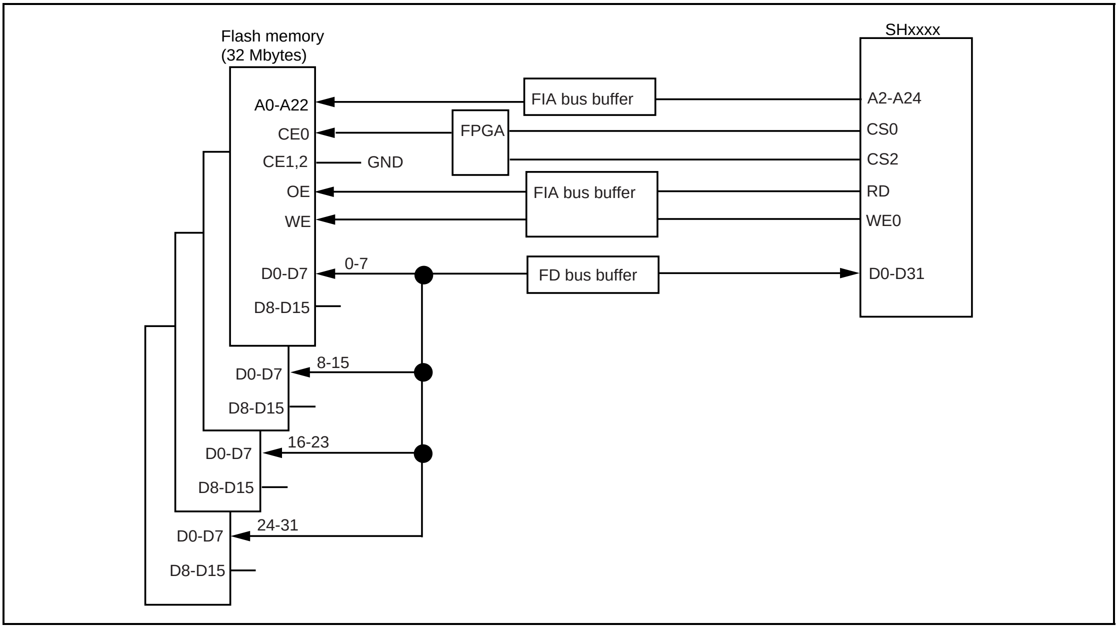


Figure 7.52 Flash Memory Wiring

Table 7.8 Sample Program Specifications

Item	Contents
RAM area to be used	H'0C001000 to H'0C0015BF
Write module start address	H'0C001100
Erase module start address	H'0C001000

- Since the SDRAM is used, the bus controller must be set.
- Set the options on the [Loading flash memory] page in the [Configuration] dialog box as follows:

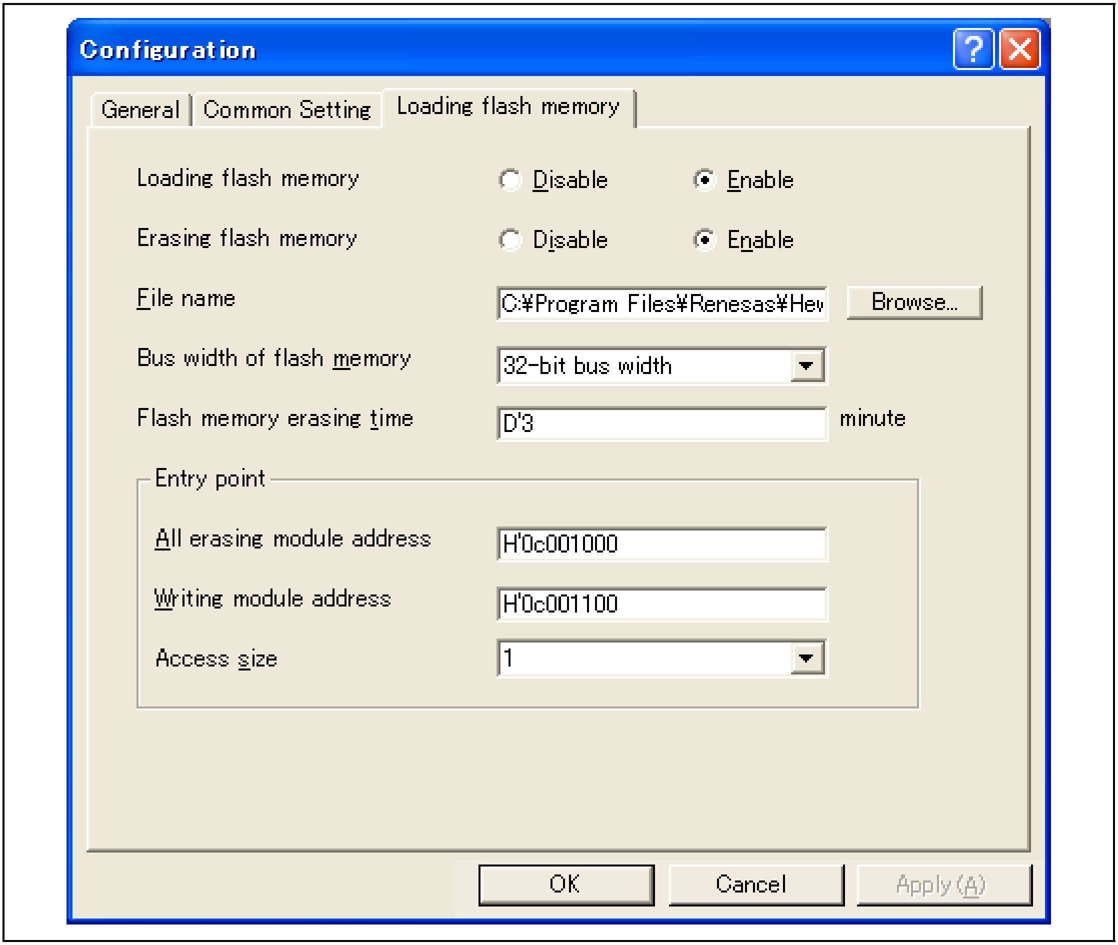


Figure 7.53 [Loading flash memory] Page

Notes: 1. When the data has already been written in the flash memory, be sure to select [Enable] for [Erasing flash memory]. If [Disable] is selected, a verify error occurs.

2. When [Erasing flash memory] is selected, it takes about one minute to erase the flash memory (in this example).

- Select the object for downloading to the flash memory area.

Section 8 Maintenance and Guarantee

This section describes maintenance, guarantee, repair provisions, and how to request for repair of the emulator.

8.1 User Registration

When you purchase our product, be sure to register as a user. For user registration, refer to the section of ‘User Registration’ (p. iii) of this user's manual.

8.2 Maintenance

(1) If dust or dirt collects on any equipment of this product, wipe the board dry with a soft cloth. Do not use thinner or other solvents because these chemicals can cause the equipment's surface coating to separate.

(2) When you do not use this product for a long period, for safety purposes, disconnect the power cable from the power supply.

8.3 Guarantee

If your product becomes faulty within one year after its purchase while being used under good conditions by observing ‘IMPORTANT INFORMATION’ described in this user's manual, we will repair or replace your faulty product free of charge. Note, however, that if your product's fault is raised by any one of the following causes, we will repair it or replace it with new one with extra-charge:

- Misuse, abuse, or use under extraordinary conditions
- Unauthorized repair, remodeling, maintenance, and so on
- Inadequate user's system or misuse of it
- Fires, earthquakes, and other unexpected disasters

In the above cases, contact your local distributor. If your product is being leased, consult the leasing company or the owner.

8.4 Repair Provisions

(1) Repair with Extra-Charge

The products elapsed more than one year after purchase can be repaired with extra-charge.

(2) Replacement with Extra-Charge

If your product's fault falls in any of the following categories, the fault will be corrected by replacing the entire product instead of repair, or you will be advised to purchase new one, depending on the severity of the fault.

- Faulty or broken mechanical parts
- Flaw, separation, or rust in coated or plated parts
- Flaw or cracks in plastic parts
- Faults or breakage caused by improper use or unauthorized repair or modification
- Heavily damaged electric circuits due to overvoltage, overcurrent or shorting of power supply
- Cracks in the printed circuit board or burnt-down patterns
- Wide range of faults that makes replacement less expensive than repair
- Unlocatable or unidentified faults

(3) Expiration of the Repair Period

When a period of one year elapses after the model was dropped from production, repairing products of the model may become impossible.

(4) Transportation Fees at Sending Your Product for Repair

Send your product to us for repair at your expense.

8.5 How to Make a Request for Repair

If your product is found faulty, follow the procedure below to send your product for repair.

Fill in the Repair Request Sheet included with this product, then send it along with this product for repair to your local distributor. Make sure that information in the Repair Request Sheet is written in as much detail as possible to facilitate repair.

⚠ CAUTION

Note on Transporting the Product:

When sending your product for repair, use the packing box and cushion material supplied with this product when delivered to you and specify handling caution for it to be handled as precision equipment. If packing of your product is not complete, it may be damaged during transportation. When you pack your product in a bag, make sure to use conductive polyvinyl supplied with this product (usually a blue bag). When you use other bags, they may cause a trouble on your product because of static electricity.

Appendix A Troubleshooting

1. ***I have a text file open in the editor but syntactic color-coding is not being displayed.***

   Ensure that you have named the file (i.e. saved it) and that the "Syntax coloring" check box is set on the "Editor" tab of the "Options" dialog box, which is launched via **[Setup -> Options...]**. The High-performance Embedded Workshop looks up the filename extension to determine the group to which the file belongs and decides whether or not coloring should be applied to the file. To view the currently defined filename extensions and file groups, select **[Project -> File Extensions...]** to launch the "File Extensions" dialog box. To view the coloring information, select **[Setup -> Format]** to display the "Color" tab of the "Format" dialog box.

2. ***I want to change the settings of a tool but the [Tools->Administration…] menu option is not selectable.***

   [Tools->Administration...] is not selectable while a workspace is open. To open the "Tool Administration" dialog box, close the current workspace.

3. ***I opened a workspace from my PC, and one of my colleagues opened the same workspace simultaneously from another PC. I changed the settings of the workspace and saved it. My colleague saved the workspace after me. I opened the workspace again and found that the settings of the workspace differed from those I had made.***

   The last settings to be saved are effective. While a workspace is open in the High-performance Embedded Workshop, updating of the workspace is within the memory. The settings are not saved in a file unless the user intentionally saves the workspace.

In addition to above, refer to FAQs on the emulator and High-performance Embedded Workshop on the Renesas web site (www.renesas.com).

Appendix B Menus

Table B.1 shows GUI menus.

Table B.1 GUI Menus

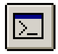
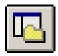
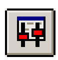
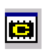
Menu	Option		Shortcut	Toolbar Button	Remarks
View	Disassembly		Ctrl + D		Opens the [Disassembly] window.
	Command Line		Ctrl + L		Opens the [Command Line] window.
	TCL toolkit		Shift + Ctrl + L		Opens the [Console] window.
	Workspace		Alt + K		Opens the [Workspace] window.
	Output		Alt + U		Opens the [Output] window.
	Difference				Opens the [Difference] window.
	CPU	Registers	Ctrl + R		Opens the [Register] window.
		Memory…	Ctrl + M		Opens the [Memory] window.
		IO	Ctrl + I		Opens the [IO] window.
		Status	Ctrl + U		Opens the [Status] window.
		Cache	Shift + Ctrl + C		Opens the [Cache] window.
		TLB	Shift + Ctrl + X		Opens the [TLB] window.
	Sym-bol	Labels	Shift + Ctrl + A		Opens the [Labels] window.
		Watch	Ctrl + W		Opens the [Watch] window.
		Locals	Shift + Ctrl + W		Opens the [Locals] window.

Table B.1 GUI Menus (cont)

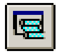
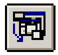
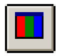
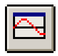
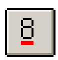
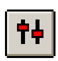
Menu	Option		Shortcut	Toolbar Button	Remarks
View (cont)	Code	Eventpoints	Ctrl + E		Opens the [Event] window.
		Trace	Ctrl + T		Opens the [Trace] window.
		Stack Trace	Ctrl + K		Opens the [Stack Trace] window.
	Gra-phic	Image…	Shift + Ctrl + G		Opens the [Image] window.
		Waveform…	Shift + Ctrl + V		Opens the [Waveform] window.
	Per-form-ance	Performance Analysis	Shift + Ctrl + P		Opens the [Performance Analysis] window.
Setup	Radix	Hexadecimal			Uses a hexadecimal for displaying a radix in which the numerical values will be displayed and entered by default.
		Decimal			Uses a decimal for displaying a radix in which the numerical values will be displayed and entered by default.
		Octal			Uses an octal for displaying a radix in which the numerical values will be displayed and entered by default.
		Binary			Uses a binary for displaying a radix in which the numerical values will be displayed and entered by default.
	Emu-lator	System…			Opens the [Configuration] dialog box allowing the user to modify the debugging platform settings.

Table B.1 GUI Menus (cont)

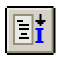
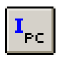
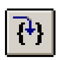
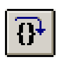
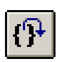
Menu	Option	Shortcut	Toolbar Button	Remarks
Debug	Debug Sessions…			Opens the [Debug Sessions] dialog box to list, add, or remove the debug session.
	Debug Settings…			Opens the [Debug Settings] dialog box to set the debugging conditions or download modules.
	Reset CPU			Resets the target hardware and sets the PC to the reset vector address.
	Go	F5		Starts executing the user program at the current PC.
	Reset Go	Shift + F5		Resets the target microcomputer and executes the user program from the reset vector address.
	Go To Cursor			Starts executing the user program at the current PC until the PC reaches the address indicated by the current text cursor position.
	Set PC To Cursor			Sets the PC to the address at the row of the text cursor.
	Run…			Launches the [Run Program] dialog box allowing the user to enter the PC or PC breakpoint during executing the user program.
	Step In	F11		Executes a block of user program before breaking.
	Step Over	F10		Executes a block of user program before breaking. If a subroutine call is reached, then the subroutine will not be entered.
	Step Out	Shift + F11		Executes the user program to reach the end of the current function.
	Step…			Launches the [Step Program] dialog box allowing the user to modify the settings for stepping.

Table B.1 GUI Menus (cont)

Menu	Option		Shortcut	Toolbar Button	Remarks
Debug (cont)	Step Mode	Auto			Steps only one source line when the [Source] window is active. When the [Disassembly] window is active, stepping is executed in a unit of assembly instructions.
		Assembly			Executes stepping in a unit of assembly instructions.
		Source			Steps only one source line.
	Halt Program		Esc		Stops the execution of the user program.
	Connect				Connects the debugging platform.
	Initialize				Disconnects the debugging platform and connects it again.
	Disconnect				Disconnects the debugging platform.
	Download Modules				Downloads the object program.
	Unload Modules				Unloads the object program.

Appendix C Command-Line Functions

The emulator supports the commands that can be used in the command-line window.

For details, refer to the online help.

Appendix D Notes

1. Note on Moving Source File Position after Creating Load Module

   When the source file is moved after creating the load module, the [Open] dialog box may be displayed to specify the source file during the debugging of the created load module. Select the corresponding source file and click the [Open] button.

2. Source-Level Execution

   — Source file

   Do not display source files that do not correspond to the load module in the program window. For a file having the same name as the source file that corresponds to the load module, only its addresses are displayed in the program window. The file cannot be operated in the program window.

   — Step

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

3. Operation During Accessing Files

   Do not perform other operations during downloading the load module, operating [Verify Memory] or [Save Memory] in the [Memory] window, or saving in the [Trace] window because this will not allow correct file accessing to be performed.

4. Watch

   — Local variables at optimization

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

   Example: The variable name is asc.

   asc = ? - target error 2010 (xxxx)

— Variable name specification

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

Example: The function name is main.

main =

5. Line Assembly

— Input radix

Regardless of the Radix setting, the default for line assembly input is decimal. Specify H' or 0x as the radix for a hexadecimal input.

6. Command Line Interface

— Batch file

To display the message "Not currently available" while executing a batch file, enter the sleep command. Adjust the sleep time length which differs according to the operating environment.

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

sleep d'3000

memory_fill 0 ffff 0

— File specification by commands

The current directory may be altered by file specifications in commands. It is recommended to use absolute paths are recommended to be used to specify the files in a command file so that the current directory alteration is not affected.

Example: FILE_LOAD C:\Hew3\Tools\Renesas\DebugComp\Platform \E10A-USBM\Tutorial\Tutorial\Debug_SHxxxx_E10A-USBM_ SYSTEM\tutorial.abs

7. Memory Save During User Program Execution

Do not execute memory save or verifying during user program execution.

8. Load of Motorola S-type Files

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

9. Note on [Register] Window Operation During Program Execution

   The register value cannot be changed in the [Register] window during program execution. Even if the changed value is displayed, the register contents are not changed actually.

10. Break Functions

    — When the PC breakpoint is set in the internal flash memory area, the program is written to the internal flash memory each time the user program is executed. At this time, note that the number of rewritable times will be decreased.

    — BREAKPOINT cancellation

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

    BREAKPOINT IS DELETED A=xxxxxxxx

    If the above message is displayed, cancel all BREAKPOINT settings with the [Delete All] or [Disable] button in the [Eventpoint] window.

11. Number of BREAKPOINT and [Stop At] Settings in the [Run...] Menu

    The maximum number of BREAKPOINTs and [Stop At] settings allowed in the [Run...] menu is 255. Therefore, when 255 BREAKPOINTs are set, specification by [Stop At] in the [Run...] menu becomes invalid. Use the BREAKPOINTs and [Stop At] in the [Run...] menu with 255 or less total settings.

12. Note on RUN-TIME Display

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

13. Note on Displaying `Timeout error`

    If `Timeout error` is displayed, the emulator cannot communicate with the target microcomputer. Turn off the user system and connect the USB connector of the emulator again by using the HEW.

14. Note on Using the [Run Program] Dialog Box

    When [Run...] is selected from the [Debug] menu to specify the stop address, there is the following note:

    — When the breakpoint that has been set as Disable is specified as the stop address, note that the breakpoint becomes Enable when the user program stops.

15. Memory Access during User Program Execution

When a memory is accessed from the memory window, etc. during user program execution, the user program is resumed after it has stopped in the emulator to access the memory. Therefore, realtime emulation cannot be performed.

The stopping time of the user program is as follows:

Environment:

Host computer: 3 GHz (Pentium® 4)

SH7265R: System clock frequency 66.6 MHz

JTAG clock: 2.5 MHz

When a one-byte memory is read from the command-line window, the stopping time will be about 70 ms.

16. BREAKPOINT Setting for SLEEP Instruction

When a break is set for the SLEEP instruction, use the Break Condition not the BREAKPOINT.

17. Note on Session Save in the [Configuration] Dialog Box

The following settings are not saved as a session:

— JTAG clock in the [General] page

— Loading flash memory in the [Loading flash memory] page

18. Scrolling Window During User Program Execution

Do not scroll the [Memory] and [Disassembly] windows by dragging the scroll box during user program execution. This generates many memory reads causing the user program to stop execution until the memory reads have been completed.

19. Memory Test Function

This product does not support the memory test function, which is used by selecting [Test…] from the [Memory] menu.

20. Memory Access during Flash Memory Programming

During flash memory programming (e.g., user program execution), operation for memory accessing such as opening the [Memory] window is not allowed. Values displayed here are dummy. Access the memory again after flash memory programming has been completed.

21. Sleep States for the PC While the Emulator is in Use

Do not place the PC in sleep or hibernation states while the emulator is in use. Once the PC has entered those states, the emulator is unusable. If the PC does enter the sleep or hibernation state while the emulator is in use, reconnect the emulator after the PC has recovered from the given state.

22. Manual Navigator

Follow the procedure below to execute his program under Windows Vista® or Windows® 7.

Work-around:

(1) Log in with administrative rights.

(2) Open the properties window for file man_navi.exe in the Manuals folder under the installation folder for the High-performance Embedded Workshop.

(3) On the [Compatibility] tabbed page, check the [Run this program as an administrator] box.

Note: The manual navigator is not for use with 64-bit versions of Windows Vista®.

23. Points for caution when installing provided software products under Windows Vista® or Windows® 7

If a host machine running Windows Vista® or Windows® 7 (on which software products for the Renesas emulator have been installed) is not connectable to the emulator via the USB driver for the following reason, manual installation of the USB driver included with the provided software products will enable correct operation.

Cause:

When the emulator is connected to the host machine and the [Properties] dialog box for the USB driver is displayed from the device manager, the message below is displayed under [Device state].

″Windows cannot start this hardware device because its configuration information (in the registry) is incomplete or damaged. (Code 19) Click 'Check for solutions' to send data about this device to Microsoft and to see if there is a solution available″.

24. Notes on Using the Parallel Mode

When using the parallel mode in synchronized debugging, do not display or perform operations in a window for a CPU which is not connected to the emulator yet.

The procedure for manually installing the USB driver is given below.

1. Double click on <name of drive containing the CD-R with the provided software products>: \drivers\Renesas_E_Series_USB\dpinst.exe, and execute dpinst.exe.
2. The [User Account Control] dialog box is displayed. Even though the message "An unidentified program wants access to your computer. Don't run the program unless you know where it's from or you've used it before." is displayed, click on the [Allow] button.

   Note: The file dpinst.exe is the driver package installation utility provided by Microsoft Corp.
3. The [Device Driver Installation Wizard] is displayed so click on the [Next] button.
4. The [Windows Security] dialog box with the message "Would you like to install this device software? "is displayed: click on the [Install] button.
5. Select the [Finish] button of the [Device Driver Installation Wizard].

Appendix E Diagnostic Test Procedure

For the diagnostic test procedure using the emulator test program, refer to the test program manual for the emulator (file name: E10A-USBMTME.pdf) on the CD-R (E10A-USB Emulator for Multi-core Microcomputers).

.

Appendix F Repair Request Sheet

Thank you for purchasing the E10A-USB emulator (HS0005KCU04, HS0005KCU14H).

In the event of a malfunction, fill in the repair request sheet on the following pages and send it to your distributor.

Repair Request Sheet

To Distributor

Your company name:

Person in charge:

Tel.:

Item	Symptom
1. Date and time when the malfunction occurred	Month/Day/Year {at system initiation, in system operation} *Circle either of items in the braces { }.
2. Frequency of generation of the malfunction	( ) times in ( ) {day(s), week(s), or month(s)} *Enter the appropriate numbers in the parentheses ( ) and circle one of the three items in the braces { }.
3. System configuration when the malfunction occurred	System configuration of the emulator: • E10A-USB emulator (HS0005KCU04, HS0005KCU14H): Serial No.: Revision: The above items are written on the label for product management at the bottom of the emulator unit; the serial no. is the five-digit number and the revision is the string of letters following the number. • Provided CD-R (HS0005KCU04SR): Version: V. Shown as 'V.x.xx Release xx' on the CD-R (x: numeral). • Host computer in use: Manufacturer: Type number: OS: {Windows® XP, Windows Vista®, or Windows® 7}

Item	Symptom
4. Settings when the malfunction occurred	(1) MCU: Type number: (2) Operating frequency: MHz
5. Failure phenomenon	
6. Error in debugging	
7. Error in the diagnostic program	
8. The High-performance Embedded Workshop does not link-up with the emulator.	Content of the error message

For errors other than the above, fill in the box below.

SuperH™ Family E10A-USB Emulator
for Multi-core Microcomputers
User's Manual

Publication Date: Rev.5.00 August 10, 2012

Published by: Renesas Electronics Corporation

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.2

SuperH™ Family E10A-USB Emulator
for Multi-core Microcomputers
User's Manual

Renesas Electronics Corporation

R20UT0363EJ0500