To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

To all our customers

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

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

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

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

RENESAS
Renesas Technology Corp.

Cautions

Keep safety first in your circuit designs!

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

Notes regarding these materials

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

User's Manual

SuperH™ RISC engine Simulator/Debugger

User's Manual

Renesas Microcomputer
Development Environment
System

Renesas Electronics
www.renesas.com

Rev.2.0 2002.05

Cautions

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

Trademarks:

Microsoft® and Windows® are registered trademarks of Microsoft Corporation in the United States and/or other countries.

IBM PC is the name of a computer administered by International Business Machines Corporation.

ELF/DWARF2 is the name of an object format developed by the Tool Interface Standards Committee.

All products or brand names used in the manual are trademarks or registered trademarks of their respective companies.

Read First:

1. Hitachi, Ltd. (including its subsidiaries, hereafter collectively referred to as Hitachi) pursues a policy of continuing improvement in design, performance, and safety of the system. Hitachi reserves the right to change, wholly or partially, the specifications, design, user's manual, and other documentation at any time without notice.
2. This user's manual and this system are copyrighted and all rights are reserved by Hitachi. No part of this user's manual, all or part, may be reproduced or duplicated in any form, in hard-copy or machine-readable form, by any means available without Hitachi's prior written consent.
3. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.

Preface

Read First

READ this user's manual before using the simulator debugger.

KEEP the user's manual handy for future reference.

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

About this Manual

This manual explains the use of the simulator debugger and the Hitachi Embedded Workshop (HEW) for Hitachi microcomputer development tools. The following section will provide a brief *Introduction* to the debugging interface and simulator/debugger, and list its key features.

The following sections, *System Overview, Simulator/Debugger Functions*, *Menus*, *Windows and Dialog Boxes*, *Command Lines*, and *Messages*, give reference information about the operation and facilities available from these respective areas.

This manual assumes that the HEW is used on the English version of Microsoft® Windows®Me operating system running on the IBM PC.

Assumptions

It is assumed that the reader has a competent knowledge of the C/C++ programming language, assembly-language mnemonics for the processor being debugged and is experienced in using Microsoft® Windows® applications.

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 file names.
"enter this string"	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, while 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.

Contents

Figures

Tables

Section 1 Overview

The Hitachi Embedded Workshop (HEW) is a Graphical User Interface intended to ease the development and debugging of applications written in C/C++ programming language and assembly language for Hitachi microcomputers. Its aim is to provide a powerful yet intuitive way of accessing, observing and modifying the debugging platform in which the application is running.

Key Features

- **Intuitive interface**
- **On-line help**
- **Common "Look & Feel"**

Note: The HEW does not run on Windows® version 3.1.

When used with the following software, the simulator/debugger reduces the time required for software development.

- SuperH™ RISC engine series C/C++ compiler
- SuperH™ RISC engine series cross assembler
- Optimizing linkage editor

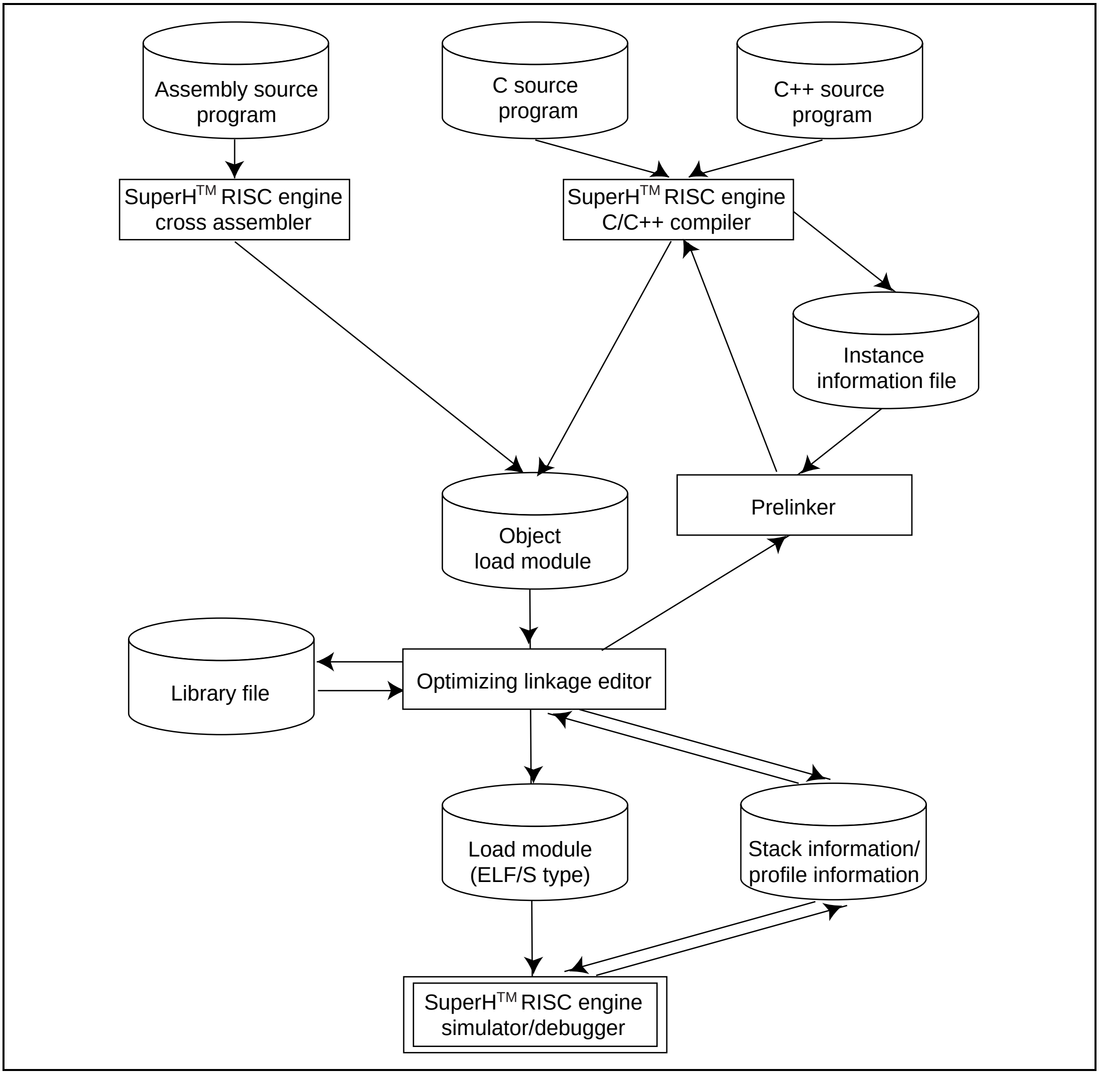
1.1 Features

- Since the simulator/debugger runs on a host computer, software debugging can start without using an actual user system, thus reducing overall system development time.
- The simulator/debugger performs a pipeline simulation to calculate the number of instruction execution cycles for a program, thus enabling performance evaluation without using an actual user system.
- The simulator/debugger offers the following features and functions that enable efficient program testing and debugging.
  - — The ability to handle all of the SuperH™ RISC engine series CPUs
  - — Functions to stop or continue execution when an error occurs during user program execution
  - — Profile data acquisition and function-unit performance measurement
  - — A comprehensive set of break functions (Pseudo interrupts are also possible)
  - — Functions to set or edit memory maps
  - — Functions to display function call history
  - — Coverage information is displayed at the C/C++ or assembly-source level
  - — Visual debugging functions provide the display of data as images or waveforms
- The breakpoint, memory map, performance, and trace can be set through the dialog box under Windows®. Environments corresponding to each memory map of the SuperH™ RISC engine microprocessors can be set through the dialog box.

1.2 Target User Program

Load modules in ELF/DWARF2 format and S-type format can be debugged with the simulator/debugger. These load modules are called user programs in this manual.

Figure 1.1 shows the creation of target user programs to be debugged.



Assembly source program
C source program
C++ source program
SuperH™ RISC engine cross assembler
SuperH™ RISC engine C/C++ compiler
Instance information file
Object load module
Prelinker
Library file
Optimizing linkage editor
Load module (ELF/S type)
Stack information/ profile information
SuperH™ RISC engine simulator/debugger

Figure 1.1 Creation of Target User Programs

1.3 Simulation Range

The simulator/debugger provides simulation functions for SuperH™ RISC engine series (SH-1, SH-2, SH-2E, SH-3, SH-3E, SH3-DSP, SH-4, and SH2-DSP series) microprocessors and provides debugging functions for programs written in C, C++, or assembly language. Therefore, the simulator/debugger promotes efficient debugging of programs. In “the SH-4 series”, there are two types of microprocessors, “SH-4” and “SH-4 (SH7750R)”, which have different cache specifications. In addition, “the SH-4” consists of two different-version microprocessors; one improves the simulation speed by limiting a part of simulation functions (called "SH-4" in this manual) and one provides high-level functions (called “SH-4BSC” in this manual). “The SH2-DSP series” consists of “the SH2-DSP (Core) ”, “the SH2-DSP (SH7410) ”, and “the SH2-DSP (SH7065) ” which do not have cache, and “the SH2-DSP (SH7612) ”, which has on-chip cache. "The SH3-DSP series" consists of "the SH3-DSP (Core)" and "the SH3-DSP", and these two types have different DSP functions. Note that, in this manual, “the SH-4 series” means “the SH-4”, “SH-4BSC”, and “SH-4 (SH7750R)”. “The SH2-DSP series” means “the SH2-DSP (Core)”, “the SH2-DSP (SH7410)”, “the SH2-DSP (SH7065)”, and “the SH2-DSP (SH7612)”. “The SH3-DSP series” means “the SH3-DSP (Core)” and “the SH3-DSP”.

The simulator/debugger supports the following SuperH™ RISC engine series microcomputer functions:

- All CPU instructions (pipeline simulation)
- Exception processing
- Registers
- All address areas
- Peripheral functions shown in table 1.1

Table 1.1 Simulator/Debugger Functions and the Corresponding CPUs

Names of Debugging Platforms	Endian	MMU	Cache	Register	BSC	DMAC
SH-1	—	—	—	—	—	—
SH-2/SH-2E	—	—	—	—	—	—
SH-3/SH-3E	○	○	○	○	—	—
SH3-DSP	○	○	○	○	—	—
SH3-DSP (Core)	○	○	○	○	—	—
SH-4	○	○	○	○	△	—
SH-4BSC	○	○	○	○	○	○
SH-4 (SH7750R)	○	○	○	○	△	—
SH2-DSP (SH7410)	—	—	—	—	—	—
SH2-DSP (Core)	—	—	—	—	—	—
SH2-DSP (SH7065)	○	—	—	—	—	—
SH2-DSP (SH7612)	○	—	○	○	—	—

Note: ○: Supported
—: Not supported
△: Partly supported

The simulator/debugger does not support the following SuperH™ RISC engine series MCU functions. Programs that use these functions must be debugged with the SuperH™ RISC engine series emulator.

- 16-bit free-running timer (FRT)
- Serial communication interface (SCI)
- I/O ports
- Interrupt controller (INTC)

Section 2 System Overview

HEW is a modular software system, utilizing self-contained modules for specific tasks. These modules are linked to a general purpose Graphical User Interface, which provides a *common look & feel* independent of the particular modules with which the system is configured.

2.1 User Interface

The HEW Graphical User Interface is a Windows® application that presents the debugging platform to you and allows you to set up and modify the system. Refer to a standard Windows® user manual for details on how to operate within a Windows® application.

2.2 Data Entry

When entering numbers in any dialog box or field you can always enter an expression instead of a simple number. This expression can contain symbols and can use the operators in the C/C++ programming languages. Use of C/C++ programming language features such as arrays and structures is only available if the ELF/DWARF2 format that supports C/C++ programming language debugging is in use.

In some dialogs, where there is a control expecting an end address, it is possible to enter a range by prefixing the value with a + sign. This will set the actual end address to be equal to the start address plus the entered the value.

2.2.1 Operators

The C/C++ programming language operators are available:

+, -, *, /, &, |, ^, ~, !, >>, <<, %, (,), <, >, <=, >=, ==, !=, &&, ||

2.2.2 Data Formats

Unprefixed data values will be taken as being in the default radix set by the **[Options->Radix]** menu option. The exception is count field which use decimal values by default (independent of the current default system radix).

Symbols may be used by name and ASCII character strings can be entered if surrounded by single quote characters, e.g. ‘demo’.

The following prefixes can be used to identify radices:

B’	Binary
O’	Octal
D’	Decimal
H’	Hexadecimal
0x	Hexadecimal

The contents of a register may be used by specifying the register name, prefixed by the # character, e.g.:

#R1, #FR2

2.2.3 Precision

All mathematics in expression evaluation is done using 64 bits (signed). Any values exceeding 64 bits are truncated.

2.2.4 Expression Examples

```
Buffer_start + 0x1000
#R1 | B’10001101
((pointer + (2 * increment_size)) & H’FFFF0000) >> D’15
!(flag ^ #R4)
```

2.2.5 Symbol Format

You can specify and reference symbols in the same format as in C/C++ programming language. Cast operators may be used together with symbols, and you can reference data after its type has been converted. Note the following limitations.

- Pointers can be specified up to four levels.
- Arrays can be specified up to three dimensions.
- No typedef name can be used.

2.2.6 Symbol Examples

`Object.value`	: Specifies direct reference of a member (C/C++)
`p_Object->value`	: Specifies indirect reference of a member (C/C++)
`Class::value`	: Specifies reference of a member with class (C++)
`*value`	: Specifies a pointer (C/C++)
`array[0]`	: Specifies an array (C/C++)
`Object.*value`	: Specifies reference of a pointer to member (C++)
`::g_value`	: Specifies reference of a global variable (C/C++)
`Class::function(short)`	: Specifies a member function (C++)
`(struct STR) *value`	: Specifies cast operation (C/C++)

Section 3 Simulator/Debugger Functions

This section describes the functions of the SuperH™ RISC engine series simulator/debugger.

3.1 Simulator/Debugger Memory Management

3.1.1 Memory Map Specification

A memory map can be specified in the **Simulator System** dialog box to calculate the number of memory access cycles during simulation.

The following items can be specified:

- Memory type
- Start and end addresses of the memory area
- Number of memory access cycles
- Memory data bus width

The memory types that can be specified depend on the CPU. For details, refer to section 5.24, Simulator System Dialog Box. The user program can be executed in all areas except for the internal I/O area.

3.1.2 Memory Resource Specification

A memory resource must be specified to load and execute a user program.

The memory resource, including the following items, can be specified in the **System Memory Resource** dialog box.

- Start address
- End address
- Access type

The access type can be read/write, read-only, or write-only. Since an error occurs if the user program attempts an illegal access (for example, trying to write to a read-only memory), such an illegal access in the user program can be easily detected.

3.2 Endian

In the SH-3, SH-3E, SH3-DSP series, SH-4 series, SH2-DSP (SH7065), and SH2-DSP (SH7612), little endian as well as big endian can be specified as the data allocation format in the memory; a user program created in the little endian format can also be simulated and debugged. Specify the endian when selecting the debugging platform.

The specified endian is valid for all accesses to external memory, and in the SH3-DSP series it is also valid for accesses to the X or Y memory; word or longword data is written to or read from the memory in the specified byte order.

Note: The specified endian is applied to all accesses to external memory in common. The actual SH2-DSP (SH7612) and SH2-DSP (SH7065) have the function for specifying endian in memory area units, but the simulator/debugger does not support this function.

3.3 Pipeline Reset Processing

The simulator/debugger, which simulates the pipeline execution, resets the pipeline when:

- The program counter (PC) is modified after the instruction simulation stops and before it restarts.
- The Run command to which the execution start address has been specified is executed.
- Initialization is performed, or a program is loaded.
- Memory data being currently fetched and decoded is rewritten.

When the pipeline is reset, data already fetched and decoded is cleared, and new data is fetched and decoded from the current PC. In addition, the number of executed instructions and the number of instruction execution cycles are zero-cleared.

3.4 Memory Management Unit (MMU)

For the SH-3, SH-3E, SH3-DSP series, and SH-4 series, the simulator/debugger simulates MMU operations such as TLB operations, address translation, or MMU-related exceptions (TLB miss exception, TLB protection exception, TLB invalid exception, and initial page write exception). The user program using address translation by the MMU can be simulated and debugged. In addition, the MMU-related exception handler routines can be simulated and debugged. The MMU functions depend on the CPU.

SH-3, SH-3E, and SH3-DSP Series:

The following dialog boxes are provided to manipulate the 32-entry 4-way TLB contents.

- **TLB** dialog box: Displays and flushes the TLB contents
- **TLB Modify** dialog box: Modifies the TLB contents
- **TLB Find** dialog box: Searches the TLB contents

For details, refer to section 5.29, TLB Dialog Box, section 5.30, TLB Modify Dialog Box, and section 5.31, TLB Find Dialog Box.

The TLB is mapped in the range H'F2000000 to H'F3FFFFFF, that is, all entries of the TLB are allocated within this range.

SH-4 Series:

The following dialog boxes are provided to manipulate the 4-entry instruction TLB (ITLB) and 64-entry unified TLB (UTLB) contents:

- **Instruction TLB** dialog box: Displays and flushes the ITLB contents
- **Instruction TLB Modify** dialog box: Modifies the ITLB contents
- **Instruction TLB Find** dialog box: Searches the ITLB contents
- **Unified TLB** dialog box: Displays and flushes the UTLB contents
- **Unified TLB Modify** dialog box: Modifies the UTLB contents
- **Unified TLB Find** dialog box: Searches the UTLB contents

For details, refer to section 5.33, Instruction TLB Dialog Box, through section 5.38, Unified TLB Find Dialog Box.

The ITLB is mapped in the range H'F2000000 to H'F3FFFFFF, and the UTLB is mapped in the range H'F6000000 to H'F7FFFFFF. The simulator/debugger does not support data array 2 for both ITLB and UTLB.

As well as during user program execution, the MMU translates virtual addresses into physical addresses during address display or input in the dialog boxes or windows. Therefore, in the dialog boxes and windows, memory can be accessed with the virtual addresses used in the user program. However, note that physical addresses must be used in the **[Memory map]** and **[System memory resource]**.

Note: If an associative write to a TLB entry is performed by using the Memory window, the entry may not be modified correctly. In this case, use the Edit dialog box in the longword format. To open the Edit dialog box in the longword format, open the Memory window in the longword format and double-click the data to be modified.

3.5 Cache

For the SH-3, SH-3E, SH3-DSP series, SH-4 series, and SH2-DSP (SH7612), the simulator/debugger simulates cache operations and displays the cache contents and cache hit rate. Cache operations during user program execution can be monitored. The cache functions depend on the CPU.

3.5.1 Displaying Cache Contents

SH-3, SH-3E, SH3-DSP series, and SH2-DSP (SH7612):

The following dialog boxes are provided to manipulate the cache:

- **Cache** dialog box: Displays and flushes the cache contents
- **Cache Modify** dialog box: Modifies the cache contents

For the SH-3 and SH-3E series, the **Cache** dialog box enables the cache capacity to be modified and the half of the cache to be used as internal RAM. Table 3.1 shows the cache capacity and the ways to be used.

Table 3.1 Specifiable Cache Capacity for SH-3 and SH-3E Series Simulator/Debugger

Cache Capacity	Ways to Be Used	Internal RAM Specification (Ways that Can Be Used as Internal RAM)
8 kbytes	Ways 0 to 3	Ways 2 and 3 can be used as internal RAM
4 kbytes	Ways 0 and 1	Way 1 can be used as internal RAM
2 kbytes	Way 0	No way can be used as internal RAM

For details, refer to section 5.39, Cache Dialog Box, and section 5.40, Cache Modify Dialog Box.

The cache is mapped in the range H'F0000000 to H'F1FFFFFF in the SH-3, SH-3E, and SH3-DSP series. In the SH2-DSP (SH7612), the address array is mapped in the range H'60000000 to H'7FFFFFFF, and the data array is mapped in the range H'C0000000 to H'C0000FFF.

SH-4/SH-4BSC:

The simulator/debugger simulates operations of the 8-kbyte instruction cache (IC), the 16-kbyte operand cache (OC), and two 32-byte store queues (SQ).

The following dialog boxes are provided to manipulate the IC and OC contents:

- **Instruction Cache** dialog box: Displays and flushes the IC contents
- **Instruction Cache Modify** dialog box: Modifies the IC contents
- **Operand Cache** dialog box: Displays and flushes the OC contents
- **Operand Cache Modify** dialog box: Modifies the OC contents

For details, refer to section 5.42, Instruction Cache Dialog Box, through section 5.45, Operand Cache Modify Dialog Box.

The IC is mapped in the range H'F0000000 to H'F1FFFFFF, the OC is mapped in the range H'F4000000 to H'F5FFFFFF, and the SQ is mapped in the range H'E0000000 to H'E3FFFFFF.

Note: If an associative write to a cache entry or modification of a cache address array is performed by using the Memory window, the entry or array may not be modified correctly. In this case, use the Edit dialog box in the longword format. To open the Edit dialog box in the longword format, open the Memory window in the longword format and double-click the data to be modified.

The simulator/debugger does not change the high-order three bits of the address tag stored in a cache address array to zeros.

When loading a program, by using the **Load Object File** dialog box, to the area where the cache is mapped, or copying memory data to this area by using the **Copy Memory** dialog box, clear the AT bit of the MMUCR to zero to disable the MMU.

SH-4 (SH7750R):

The simulator/debugger simulates operations of the 16-kbyte instruction cache (IC), the 32-kbyte operand cache (OC), and two 32-byte store queues (SQ).

The following dialog boxes are provided to manipulate the IC and OC contents:

- **Instruction Cache** dialog box: Displays and flushes the IC contents
- **Instruction Cache Modify** dialog box: Modifies the IC contents
- **Operand Cache** dialog box: Displays and flushes the OC contents
- **Operand Cache Modify** dialog box: Modifies the OC contents

For details, refer to section 5.42, Instruction Cache Dialog Box, through section 5.45, Operand Cache Modify Dialog Box.

The IC is mapped in the range H'F0000000 to H'F1FFFFFF, the OC is mapped in the range H'F4000000 to H'F5FFFFFF, and the SQ is mapped in the range H'E0000000 to H'E3FFFFFF.

Note: If an associative write to a cache entry or modification of a cache address array is performed by using the Memory window, the entry or array may not be modified correctly. In this case, use the Edit dialog box in the longword format. To open the Edit dialog box in the longword format, open the Memory window in the longword format and double-click the data to be modified.

The simulator/debugger does not change the high-order three bits of the address tag stored in a cache address array to zeros.

When loading a program, by using the **Load Object File** dialog box, to the area where the cache is mapped, or copying memory data to this area by using the **Copy Memory** dialog box, clear the AT bit of the MMUCR to zero to disable the MMU.

3.5.2 Cache Hit Rate

Checking and Displaying the Cache Hit Rate: The simulator/debugger displays the cache hit rate in percentage in the **Platform** sheet in the **System Status** window. The cache hit rate is obtained by dividing the cache hit count by the cache access count (the sum of the cache hit count and cache miss count). The cache hit count and the cache miss count are also displayed.

Initializing the Cache Hit Rate: The displayed cache hit rate is reset to zero when the simulator/debugger is initiated, the pipeline is reset, or the CCR register value is modified. In the SH3-DSP series, the cache hit rate is reset to zero also when the CCR2 control register value is modified.

3.6 Bus State Controller (BSC)

For the SH-4BSC, the simulator/debugger has the functions for specifying and modifying the memory map to use the BSC; the user program using the BSC can be debugged.

Table 3.2 lists the memory types that can be specified for the SH-4BSC.

Table 3.2 Memory Types for the SH-4BSC Simulator/Debugger

Address	Specifiable Memory Types
H'00000000 to H'03FFFFFF (area 0)	Normal memory, burst ROM, and MPX
H'04000000 to H'07FFFFFF (area 1)	Normal memory, byte control SRAM, and MPX
H'08000000 to H'0BFFFFFF (area 2)	Normal memory, DRAM, SDRAM, and MPX
H'0C000000 to H'0FFFFFFF (area 3)	Normal memory, DRAM, SDRAM, and MPX
H'10000000 to H'13FFFFFF (area 4)	Normal memory, byte control SRAM, and MPX
H'14000000 to H'17FFFFFF (area 5)	Normal memory, burst ROM, and MPX
H'18000000 to H'1BFFFFFF (area 6)	Normal memory, burst ROM, and MPX
H'1C000000 to H'1FFFFFFF (area 7)	Cannot be specified
H'7C000000 to H'7C001FFF	Internal RAM (cannot be changed)
H'E0000000 to H'FFFFFFFF	I/O (cannot be changed)

The high-order three bits of the addresses for areas 0 to 7 in table 3.2 must be ignored; H'00000000 and H'20000000 are both in area 0.

The simulator/debugger does not support the PCMCIA.

For details on memory mapping, refer to section 5.24, Simulator System Dialog Box.

3.7 Direct Memory Access Controller (DMAC)

For the SH-4BSC, the simulator/debugger simulates the 4-channel DMAC operations; the user program using the DMAC can be debugged.

3.8 SH-4/SH-4 (SH7750R) Supporting Functions

3.8.1 BSC

For the SH-4/SH-4 (SH7750R), by eliminating the bus control function in the BSC, only SRAM, bus width, and the number of states can be specified.

Table 3.3 lists the memory types that can be specified for the SH-4/SH-4 (7750R).

Table 3.3 Memory Types for the SH-4/SH-4 (7750R) Simulator/Debugger

Address	Specifiable Memory Types
H'00000000 to H'03FFFFFF (area 0)	SRAM
H'04000000 to H'07FFFFFF (area 1)	
H'08000000 to H'0BFFFFFF (area 2)	
H'0C000000 to H'0FFFFFFF (area 3)	
H'10000000 to H'13FFFFFF (area 4)	
H'14000000 to H'17FFFFFF (area 5)	
H'18000000 to H'1BFFFFFF (area 6)	
H'1C000000 to H'1FFFFFFF (area 7)	Cannot be specified
H'7C000000 to H'7C001FFF	Internal RAM (cannot be changed)
H'E0000000 to H'FFFFFFFF	I/O (cannot be changed)

3.8.2 DMA

The DMA function cannot be used.

3.8.3 External/Internal Clock Ratio

The external/internal clock ratio is 1:1. This can be modified by the CLOCK_RATE command only for the SH-4 series. For details on the CLOCK_RATE command, refer to section 6.18, CLOCK_RATE.

3.8.4 Control Registers

Table 3.4 lists the control registers supported by the SH-4/SH-4E (SH7750R) simulator/debugger.

Table 3.4 Control Registers Supported by the SH-4/SH-4E (SH7750R) Simulator/Debugger (1)

Register Name	Whether or Not Supported
PTEH	Supported
PTEL	Supported
TTB	Supported
TEA	Supported
MMUCR	Supported
EXPEVT	Supported
INTEVT	Supported
TRA	Supported
CCR	Supported
QACR0, QACR1	Supported
SAR0-SAR3	Not supported
DAR0-DAR3	Not supported
DMATCR0-DMATCR3	Not supported
CHCR0-CHCR3	Not supported
DMAOR	Not supported
MCR	Not supported
BCR1, BCR2	Partly supported
WCR1, WCR2	Partly supported
WCR3	Not supported
RTCSR	Not supported
RTCNT	Not supported
RTCOR	Not supported
RFCR	Not supported

Note: Even if values are modified or referenced for the registers that are not supported via a dialog box that controls registers, etc., the simulator/debugger execution will not be affected.

The following shows how each control register is supported by each field.

Table 3.5 Control Registers Supported by the SH-4/SH-4E (SH7750R) Simulator/Debugger (2)

Register Name	Field Name	Whether or Not Supported
BCR1	ENDIAN	Supported
	MASTER	Not supported
	A0MPX	Not supported
	A0BST	Not supported
	A5BST	Not supported
	A6BST	Not supported
	DRAMTP	Not supported
	IPUP	Not supported
	OPUP	Not supported
	A1MBC	Not supported
	A4MBC	Not supported
	BREQEN	Not supported
	PSHR	Not supported
	MEMMPX	Not supported
	HIZMEM	Not supported
	HIZCNT	Not supported
	A56PCM	Not supported
BCR2	A6SZ-A0SZ	Supported
	PORTEN	Not supported
WCR1	DMAW	Not supported
	A6IW-A0IW	Supported
WCR2	A6W-A0W	Supported
	A6B	Not supported
	A5B	Not supported
	A0B	Not supported

Note: If values are modified or referenced for the registers that are not supported via a window such as IO, etc., the simulator/debugger execution will not be affected.

3.9 Exception Processing

The simulator/debugger detects the generation of exceptions corresponding to TRAPA instructions, general illegal instructions, slot illegal instructions, and address errors. In addition, for the SH-3, SH-3E, SH3-DSP series, and SH-4 series, the simulator/debugger simulates MMU-related exception processing (TLB miss, TLB protection exception, TLB invalid exception, and initial page write). For the SH-2E, SH-3E, and SH-4 series, the simulator/debugger also simulates FPU exception processing.

The simulator/debugger simulates exception processing with the following procedures, depending on the **[Execution Mode]** setting in the **Simulator System** dialog box.

SH-1, SH-2, SH-2E and SH2-DSP Series:

- When **[Continue]** is selected (continuation mode):
  1. Detects an exception during instruction execution.
  2. Saves the PC and SR in the stack area.
  3. Reads the start address from the vector address corresponding to the vector number.
  4. Starts instruction execution from the start address. If the start address is 0, the simulator/debugger stops exception processing, displays that an exception processing error has occurred, and enters the command input wait state.
- When the **[Stop]** is selected (stop mode):

  Executes steps 1 to 3 above, then stops.

SH-3, SH-3E, and SH3-DSP Series:

- When **[Continue]** is selected (continuation mode):
  1. Detects an exception during instruction execution.
  2. Saves the PC and SR to the SPC and SSR, respectively.
  3. Sets the BL bit, RB bit, and MD bit in the SR to 1s.
  4. Sets an exception code in control register EXPEVT. If necessary, appropriate values are set in other control registers.
  5. Sets the PC to the vector address corresponding to the exception cause. (If an exception is detected when the BL bit in the SR is 1, reset vector address H'A0000000 is set in the PC regardless of the exception cause.)
  6. Starts instruction execution from the address set in the PC.
- When the **[Stop]** is selected (stop mode):

  Executes steps 1 to 5 above, then stops.

SH-4 Series:

- When **[Continue]** is selected (continuation mode):
  1. Detects an exception during instruction execution.
  2. Saves the PC and SR to the SPC and SSR, respectively.
  3. Sets the BL bit, RB bit, and MD bit in the SR to 1s.
  4. Sets the FD (FPU disable) bit in the SR to 0 at reset.
  5. Sets an exception code in control register EXPEVT. If necessary, appropriate values are set in other control registers.
  6. Sets the PC to the vector address corresponding to the exception cause. (If an exception is detected when the BL bit in the SR is 1, reset vector address H'A0000000 is set in the PC regardless of the exception cause.)
  7. Starts instruction execution from the address set in the PC.
- When the **[Stop]** is selected (stop mode):

  Executes steps 1 to 6 above, then stops.

3.10 Control Registers

For the SH-3, SH-3E, SH3-DSP series, and SH-4 series, the simulator/debugger supports the memory-mapped control registers that are used for exception processing, MMU control, and cache control. In addition, for the SH-4 series, the simulator/debugger also supports the control registers that are used for BSC and DMAC control. For the SH2-DSP (SH7612), the simulator/debugger only supports the CCR register that is used for cache control. Therefore, a user program using exception processing, MMU control, cache control, BSC control, and DMAC control can be simulated and debugged.

The registers supported by the simulator/debugger are listed below.

MMU	PTEH:	Page table entry high register
	PTEL:	Page table entry low register
	TTB:	Translation table base register
	TEA:	TLB exception address register
	MMUCR:	MMU control register
Exception processing	TRA:	TRAPA exception register
	EXPEVT:	Exception event register
	INTEVT:	Interrupt event register
Cache	CCR:	Cache control register
	CCR2[*1]:	Cache control register 2
	QACR0 and QACR1[*2]:	Queue address control registers 0 and 1

BSC	BCR1 and BCR2*2:	Bus control registers 1 and 2
	WCR1 to WCR3*2:	Wait state control registers 1 to 3
	MCR*2:	Individual memory control register
	RTCSR*2:	Refresh timer control/status register
	RTCNT*2:	Refresh timer/counter
	RTCOR*2:	Refresh time constant register
	RFCR*2:	Refresh count register
DMAC	SAR0 to SAR3*2:	DMA source address registers 0 to 3
	DAR0 to DAR3*2:	DMA destination address registers 0 to 3
	DMATCR0 to DMATCR3*2:	DMA transfer count registers 0 to 3
	CHCR0 to CHCR3*2:	DMA channel control registers 0 to 3
	DMAOR*2:	DMA operation register

Notes: 1. The register marked with *1 is supported only for the SH3-DSP series.

**2. The registers marked with *2 are supported only for the SH-4 series.
Only the CCR register is supported for the SH2-DSP (SH7612).**

The simulator/debugger does not support the PCMCIA interface and the synchronous DRAM mode register.

To modify or display a control register value, use the **IO** window and the dialog box for each register. For details, refer to section 5.6, IO Window.

3.11 Trace

The simulator/debugger writes the results of each instruction execution into the trace buffer. The conditions for the trace information acquisition can be specified in the **Trace Acquisition** dialog box. Click the right mouse button in the **Trace** window and choose **[Acquisition...]** from the popup menu to display the **Trace Acquisition** dialog box. The acquired trace information is displayed in the **Trace** window. The trace information displayed in the **Trace** window depends on the target CPU as follows.

SH-1, SH-2, SH-2E, and SH2-DSP Series:

- Total number of instruction execution cycles
- Instruction address
- Pipeline execution status
- Instruction mnemonic
- Data access information (destination and accessed data)
- C/C++ or assembly-language source programs

SH-3 and SH-3E Series:

- Total number of instruction execution cycles
- Data on the address bus
- Data on the data bus
- Instruction code
- Instruction number
- Instruction mnemonic
- Instruction number that was fetched (enclosed by [ ] when the instruction did not access memory)
- Instruction number that was decoded
- Instruction number that was executed
- Instruction number that accessed memory
- Instruction number that wrote back data
- Data access information (destination and accessed data)
- C/C++ or assembly-language source programs

SH3-DSP Series:

- Total number of instruction execution cycles
- Program counter value
- Instruction code
- Instruction number that was fetched (enclosed by [ ] when the instruction did not access memory)
- Instruction number that was decoded
- Instruction number that was executed
- Instruction number that accessed memory
- Instruction number that wrote back data
- Instruction number
- Instruction mnemonic
- Data access information (destination and accessed data)
- C/C++ or assembly-language source programs

SH-4 Series:

- Total number of instruction execution cycles (CPU internal clock)
- Program counter value
- Fetched instruction code
- Instruction number that was executed, accessed memory, or wrote back data in the EX pipeline
- Instruction number that was executed, accessed memory, or wrote back data in the LS pipeline
- Instruction number that was executed, accessed memory, or wrote back data in the BR pipeline
- Instruction number that was executed, accessed memory, or wrote back data in the FP pipeline
- Instruction number assigned to the instruction to be executed
- Memory address, instruction code, and mnemonic of the instruction to be executed
- Data access information (destination and accessed data)
- C/C++ or assembly-language source programs

The trace information can be searched. The search conditions can be specified in the **Trace Search** dialog box. Click the right mouse button in the **Trace** window and choose **[Find...]** from the popup menu to display the **Trace Search** dialog box.

For details, refer to section 5.17, Trace Window.

3.12 Standard I/O and File I/O Processing

The simulator/debugger provides the **Simulated I/O** window to enable the standard I/O and file I/O processing listed in table 3.6 to be executed by the user program. When the I/O processing is executed, the **Simulated I/O** window must be open.

Table 3.6 I/O Functions

No.	Function Code	Function Name	Description
1	H'21	GETC	Inputs one byte from the standard input device
2	H'22	PUTC	Outputs one byte to the standard output device
3	H'23	GETS	Inputs one line from the standard input device
4	H'24	PUTS	Outputs one line to the standard output device
5	H'25	FOPEN	Opens a file
6	H'06	FCLOSE	Closes a file
7	H'27	FGETC	Inputs one byte from a file
8	H'28	FPUTC	Outputs one byte to a file
9	H'29	FGETS	Inputs one line from a file
10	H'2A	FPUTS	Outputs one line to a file
11	H'0B	FEOF	Checks for end of file
12	H'0C	FSEEK	Moves the file pointer
13	H'0D	FTELL	Returns the current position of the file pointer

To perform I/O processing, use the **[System Call Address]** in the **Simulator System** dialog box in the following procedure.

1. Set the address specialized for I/O processing in the **[System Call Address]**, select **[Enable]** and execute the program.
2. When detecting a subroutine call instruction (BSR, JSR, or BSRF), that is, a system call to the specialized address during user program execution, the simulator/debugger performs I/O processing by using the R0 and R1 values as the parameters.

Therefore, before issuing a system call, set as follows in the user program:

- Set the function code (table 3.6) to the R0 register

MSB 1 byte	1 byte		LSB
H'01	Function code	———	———

- Set the parameter block address to the R1 register (for the parameter block, refer to each function description)

MSB	LSB
Parameter block address	

- Reserve the parameter block and input/output buffer areas

Each parameter of the parameter block must be accessed in the parameter size.

After the I/O processing, the simulator/debugger resumes simulation from the instruction that follows the system call instruction.

Note: When a JSR, BSR, or BSRF instruction is used as a system call instruction, the instruction following the JSR, BSR, or BSRF instruction is executed as a normal instruction, not a slot instruction. Therefore, the instruction placed immediately after the system call instruction (JSR, BSR, or BSRF) must not be one that produces different results depending on whether executed as a normal instruction or as a slot instruction.

Each I/O function is described in the following format:

(1)	(2)	(4)
	(3)	

Parameter Block

(5)

Parameters

(6)

(1) Number corresponding to table 3.6

(2) Function name

(3) Function code

(4) I/O overview

(5) I/O parameter block

(6) I/O parameters

1	GETC	Inputs one byte from the standard input device
	H'21	

Parameter Block

	One byte	One byte
+0	Input buffer start address	
+2		

Parameters

- Input buffer start address (input)
  Start address of the buffer to which the input data is written to.

2	PUTC	Outputs one byte to the standard output device
	H'22	

Parameter Block

	One byte	One byte
+0	Output buffer start address	
+2		

Parameters

- Output buffer start address (input)
  Start address of the buffer in which the output data is stored.

3	GETS	Inputs one line from the standard input device
	H'23	

Parameter Block

	One byte	One byte
+0	Input buffer start address	
+2		

Parameters

- Input buffer start address (input)
  Start address of the buffer to which the input data is written to.

4	PUTS	Outputs one line to the standard output device
	H'24	

Parameter Block

	One byte	One byte
+0	Output buffer start address	
+2		

Parameters

- Output buffer start address (input)

  Start address of the buffer in which the output data is stored.

5	FOPEN	Opens a file
	H'25	

The FOPEN opens a file and returns the file number. After this processing, the returned file number must be used to input, output, or close files. A maximum of 256 files can be open at the same time.

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Open mode	Unused
+4	Start address of file name	
+6		

Parameters

- Return value (output)

  0: Normal completion

  –1: Error
- File number (output)

  The number to be used in all file accesses after opening.
- Open mode (input)

  H'00: "r"

  H'01: "w"

  H'02: "a"

  H'03: "r+"

H'04: "w+"

H'05: "a+"

H'10: "rb"

H'11: "wb"

H'12: "ab"

H'13: "r+b"

H'14: "w+b"

H'15: "a+b"

These modes are interpreted as follows.

"r": Open for reading.

"w": Open an empty file for writing.

"a": Open for appending (write starting at the end of the file).

"r+": Open for reading and writing.

"w+": Open an empty file for reading and writing.

"a+" : Open for reading and appending.

"b" : Open in binary mode.

- Start address of file name (input)

  The start address of the area for storing the file name.

6	FCLOSE	Closes a file
	H'06	

Parameter Block

	One byte	One byte
+0	Return value	File number

Parameters

- Return value (output)

  0: Normal completion

  –1: Error

- File number (input)

  The number returned when the file was opened.

7	FGETC	Inputs one byte from a file
	H'27	

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Unused	
+4	Start address of input buffer	
+6		

Parameters

- Return value (output)

  0: Normal completion

  –1: EOF detected
- File number (input)

  The number returned when the file was opened.
- Start address of input buffer (input)

  The start address of the buffer for storing input data.

8	FPUTC	Outputs one byte to a file
	H'28	

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Unused	
+4	Start address of output buffer	
+6		

Parameters

- Return value (output)

  0: Normal completion

  –1: Error
- File number (input)

  The number returned when the file was opened.

- Start address of output buffer (input)

  The start address of the buffer used for storing the output data.

9	FGETS	Reads character string data from a file
	H'29	

Reads character string data from a file. Data is read until either a new line code or a NULL code is read, or until the buffer is full.

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Buffer size	
+4	Start address of input buffer	
+6		

Parameters

- Return value (output)

  0: Normal completion

  –1: EOF detected
- File number (input)

  The number returned when the file was opened.
- Buffer size (input)

  The size of the area for storing the read data. A maximum of 256 bytes can be stored.
- Start address of input buffer (input)

  The start address of the buffer for storing input data.

10	FPUTS	Writes character string data to a file
	H'2A	

Writes character string data to a file. The NULL code that terminates the character string is not written to the file.

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Unused	
+4	Start address of output buffer	
+6		

Parameters

- Return value (output)

  0: Normal completion

  –1: Error
- File number (input)

  The number returned when the file was opened.
- Start address of output buffer (input)

  The start address of the buffer used for storing the output data.

11	FEOF	Checks for end of file
	H'0B	

Parameter Block

	One byte	One byte
+0	Return value	File number

Parameters

- Return value (output)

  0: File pointer is not at EOF

  –1: EOF detected
- File number (input)

  The number returned when the file was opened.

12	FSEEK	Moves the file pointer to the specified position
	H'0C	

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Direction	Unused
+4	Offset	
+6		

Parameters

- Return value (output)

  0: Normal completion

  –1: Error
- File number (input)

  The number returned when the file was opened.
- Direction (input)

  0: The offset specifies the position as a byte count from the start of the file.

  1: The offset specifies the position as a byte count from the current file pointer.

  2: The offset specifies the position as a byte count from the end of the file.
- Offset (input)

  The byte count from the location specified by the direction parameter.

13	FTELL	Returns the current position of the file pointer
	H'0D	

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Unused	
+4	Offset	
+6		

Parameters

- Return value (output)
  0: Normal completion
  –1: Error
- File number (input)
  The number returned when the file was opened.
- Offset (output)
  The current position of the file pointer, as a byte count from the start of the file.

The following shows an example for inputting one character as a standard input (from a keyboard)

```
            MOV.L     PAR_ADR,R1
            MOV.L     REQ_COD,R0
            MOV.L     CALL_ADR,R3
            JSR       @R3
            NOP
  STOP      NOP
  SYS_CALL  NOP
            .ALIGN    4
  CALL_ADR  .DATA.L   SYS_CALL
  REQ_COD   .DATA.L   H’01210000
  PAR_ADR   .DATA.L   PARM
  PARM      .DATA.L   INBUF
  INBUF     .RES.B    2
            .END
```

3.13 Break Conditions

The simulator/debugger provides the following conditions for interrupting the simulation of a user program during execution.

- Break due to the satisfaction of a break command condition
- Break due to the detection of an error during execution of the user program
- Break due to a trace buffer overflow
- Break due to execution of the SLEEP instruction
- Break due to the **[STOP]** button

3.13.1 Break Due to the Satisfaction of a Break Command Condition

There are six break commands as follows:

• BREAKPOINT:	Break based on the address of the instruction executed
• BREAK_ACCESS:	Break based on access to a range of memory
• BREAK_CYCLE	Break based on the number of execution cycles
• BREAK_DATA:	Break based on the value of data written to memory
• BREAK_REGISTER:	Break based on the value of data written to a register
• BREAK_SEQUENCE:	Break based on a specified execution sequence

If **[Stop]** is specified as the action for a break condition, user program execution stops when the break condition is satisfied. For details, refer to section 5.1, Break Window.

When a break condition is satisfied and user program execution stops, the instruction at the breakpoint may or may not be executed before a break depending on the type of break, as listed in table 3.7.

Table 3.7 Processing When a Break Condition is Satisfied

Command	Instruction When a Break Condition is Satisfied
BREAKPOINT	Not executed
BREAK_ACCESS	Executed
BREAK_CYCLE	Executed
BREAK_DATA	Executed
BREAK_REGISTER	Executed
BREAK_SEQUENCE	Not executed

For BREAKPOINT and BREAK_SEQUENCE, if a breakpoint is specified at an address other than the beginning of the instruction, the break condition will not be detected.

When a break condition is satisfied during user program execution, a break condition satisfaction message is displayed on the status bar and execution stops.

3.13.2 Break Due to the Detection of an Error During Execution of the User Program

The simulator/debugger detects simulation errors, that is, program errors that cannot be detected by the CPU exception generation functions. The **Simulator System** dialog box specifies whether to stop or continue the simulation when such an error occurs. Table 3.8 lists the error messages, error causes, and the action of the simulator/debugger in the continuation mode.

Table 3.8 Simulation Errors

Error Message	Error Cause	Processing in Continuation Mode
Memory Access Error	Access to a memory area that has not been allocated	On memory write, nothing is written; on memory read, all bits are read as 1.
	Write to a memory area having the write protect attribute	
	Read from a memory area having the read disable attribute	
	Access to an area where memory does not exist	
Illegal Operation	Zero division executed by the DIV1 instruction	Operates in the same way as the actual device operation.
	Writing zero by the SETRC instruction	
Illegal DSP Operation	Shift of more than 32 bits executed by the PSHA instruction	
	Shift of more than 16 bits executed by the PSHL instruction	
Invalid DSP Instruction Code	Invalid DSP instruction code	Always stops.
TLB Multiple Hit	Hit to multiple TLB entries at MMU address translation (only for the SH-3, SH-3E, and SH3-DSP series)	Undefined.

When a simulation error occurs in the stop mode, the simulator/debugger returns to the command wait state after stopping instruction execution and displaying the error message. Table 3.9 lists the states of the program counter (PC) at simulation error stop. The status register (SR) value does not change at simulation error stop.

Table 3.9 Register States at Simulation Error Stop

Error Message	PC Value
Memory Access Error	• When an instruction is read: — SH2-DSP and SH3-DSP series The third instruction address before the instruction that caused the error. — SH-1, SH-2, SH-2E, SH-3, SH-3E, and SH-4 series The instruction address before the instruction that caused the error. The slot address if an error occurs when a branch destination is read. • When an instruction is executed: The instruction address following the instruction that caused the error.
Illegal Operation	The instruction address following the instruction that caused the error.
Illegal DSP Operation	The second instruction address following the instruction that caused the error.
Invalid DSP Instruction Code	The second instruction address following the instruction that caused the error.
TLB Multiple Hit	The address of the instruction that caused the error.

Use the following procedure when debugging programs which include instructions that generate simulation errors.

a. First execute the program in the stop mode and confirm that there are no errors except those in the intended locations.

b. After confirming the above, execute the program in the continuation mode.

Note: If an error occurs in the stop mode and simulation is continued after changing the simulator mode to the continuation mode, simulation may not be performed correctly. When restarting a simulation, always restore the register contents (general, control, and system registers) and the memory contents to the state prior to the occurrence of the error.

3.13.3 Break Due to a Trace Buffer Overflow

After the **[Break]** mode is specified with **[Trace Buffer Full Handling]** in the **Trace Acquisition** dialog box, the simulator/debugger stops execution when the trace buffer becomes full. The following message is displayed when execution is stopped.

```
Trace Buffer Full
```

3.13.4 Break Due to Execution of the SLEEP Instruction

When the SLEEP instruction is executed during instruction execution, the simulator/debugger stops execution. The following message is displayed when execution is stopped.

```
Sleep
```

Note: When restarting execution, change the PC value to the instruction address at the restart location.

3.13.5 Break Due to the [STOP] Button

Users can forcibly terminate execution by clicking the **[STOP]** button during instruction execution. The following message is displayed when execution is terminated.

```
Stop
```

Execution can be resumed with the Go or Step command.

3.14 Floating-Point Data

Floating-point numbers can be displayed and input for the following real-number data, which makes floating-point data processing easier.

- Data in the **Set Break** dialog box when the break type is set to **[Break Data]** or **[Break Register]**
- Data in the **Memory** window
- Data in the **Fill Memory** dialog box
- Data in the **Search Memory** dialog box
- Register values displayed in the **Registers** window
- Input data in the **Register** dialog box

The floating-point data format conforms to the ANSI C standard.

In the simulator/debugger, the rounding mode for floating-point decimal-to-binary conversion can be selected in the **Simulator System** dialog box. One of the following two modes can be selected:

- Round to nearest (RN)
- Round to zero (RZ)

If a denormalized number is specified for binary-to-decimal or decimal-to-binary conversion, it is converted to zero in RZ mode, and it is left as a denormalized number in RN mode. If an overflow occurs during decimal-to-binary conversion, the maximum floating-point value is returned in RZ mode, and the infinity is returned in RN mode.

3.15 Display of Function Call History

The simulator/debugger displays the function call history in the **Stack Trace** window when simulation stops, which enables program execution flow to be checked easily. Selecting a function name in the **Stack Trace** window displays the corresponding source program in the **Source** window; the function that has called the current function can also be checked.

The displayed function call history is updated in the following cases:

- When simulation stops under the break conditions described in section 3.13, Break Conditions.
- When register values are modified while simulation stops due to the above break conditions.
- While single-step execution is performed.

For details, refer to section 5.47, Stack Trace Window.

3.16 Profiler

The simulator/debugger displays the memory address and size allocated to functions and global variables, the number of function calls and the profile data. The profile data displayed differs according to the CPU. The displayed contents are as follows.

- SH-1/SH-2/SH-2E series, SH2-DSP (Core), SH2-DSP (SH7410) and SH2-DSP (SH7065):
  - Times (the number of times a function was called or a global variable was accessed.)
  - Cycle (the number of execution cycles)
  - Ext_mem (the number of times the external memory was accessed)
  - I/O_area (the number of times the input/output area was accessed)
  - Int_mem (the number of times the internal memory was accessed)

- SH-3/SH-3E/SH3-DSP Series and SH2-DSP (SH7612):
  - Times (the number of times a function was called or a global variable was accessed.)
  - Cycle (the number of execution cycles)
  - Cache miss (the number of Instruction cache misses)
  - Ext_mem (the number of times the external memory was accessed)
  - I/O_area (the number of times the input/output area was accessed)
  - Int_mem (the number of times the internal memory was accessed)

- SH-4 Series:
  - Times (the number of times a function was called or a global variable was accessed.)
  - Cycle (the number of execution cycles)
  - ICache miss (the number of Instruction cache misses)
  - OCache miss (the number of Operand cache misses)

- Ext_mem (the number of times the external memory was accessed)
- I/O_area (the number of times the input/output area was accessed)
- Int_mem (the number of times the internal memory was accessed)

Profile information is displayed in list, tree and chart formats. Using profile information it is possible to optimize user programs by reducing the size and putting the most frequently called functions in-line. Further, using the profile information saved to a file, it is possible to optimize user programs based on operational information using the optimizing linkage editor.

For details, refer to sections 5.48, Profile Window (List Sheet), through section 5.50, Profile Chart Window, and section 4.2.3, Optimize Option Profile in the Optimizing Linkage Editor User's Manual.

3.17 Pseudo-Interrupts

The simulator/debugger can generate pseudo-interrupts during simulation in two ways:

1. Pseudo-interrupts generated by break conditions

   A pseudo-interrupt can be generated by using a break command to specify **[Interrupt]** as the action when a break condition is satisfied. For details, refer to section 5.1, Break Window.
2. Pseudo-interrupts generated from the **Trigger** window

   A pseudo-interrupt can be generated by clicking a trigger button in the **Trigger** window. For details, refer to section 5.21, Trigger Window.

If another pseudo-interrupt occurs between a pseudo-interrupt occurrence and its acceptance, only the interrupt that has a higher priority can be processed.

Note: Whether an interrupt is accepted is determined by the specifications of the CPU for the selected debugging platform. Note, however, that when H'11 is specified as the priority level of an interrupt, that interrupt is always accepted. The simulator/debugger does not simulate the operation of the interrupt controller.

3.18 Coverage

The simulator/debugger acquires instruction coverage information during instruction execution within the address range specified by the user.

The coverage window displays the following items of the acquired instruction coverage information:

- Times (instruction execution count)
- Pass (result of a conditional branch instruction)

  T: Execution has branched in all cases.

  F: Execution has not branched in any cases.

  T/F: Execution has branched in some cases, but not in others.

  -: The target instruction is not a branch instruction or the instruction has not been executed.
- Address (instruction address)
- Assembler (disassembled display)
- Source (C/C++ or assembly-language source program)

The instruction coverage information can also be displayed in the editor window by highlighting the column corresponding to the source line of the executed instruction.

The instruction coverage information can be saved in or loaded from a file. Only a file in the .COV format can be loaded.

The status of each instruction execution can be monitored through the instruction coverage information. In addition, this information can be used to determine which part of a program has not been executed. For details, refer to section 5.57, Coverage Window, through section 5.64, Save Coverage Data Dialog Box.

Section 4 Menus

This document uses the standard Microsoft® menu naming convention.

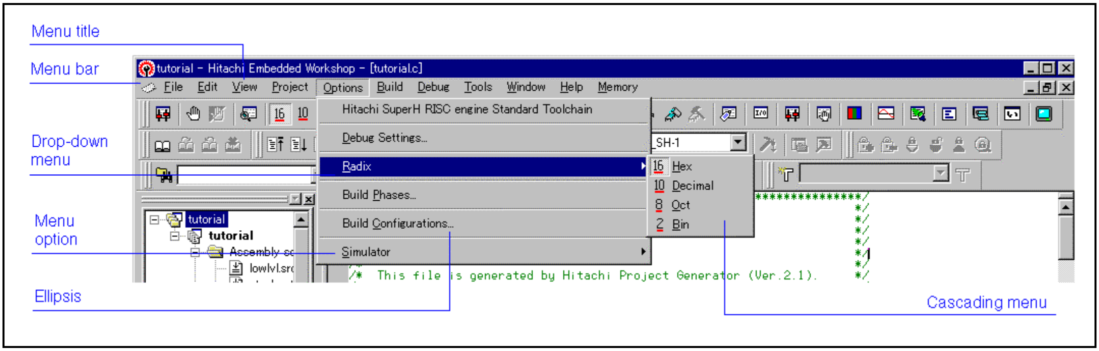


Figure 4.1 Menus

Check marks indicate that the feature provided by the menu option is selected.

Ellipsis indicates that selecting the menu option will open a dialog box that requires extra information to be entered.

Refer to your Windows® user manual for details on how to use the Windows® menu system.

4.1 View

The View menu is used to select and open new child windows. If the menu option is grayed, then the features provided by the window are not available with the current debugging platform.

4.1.1 Workspace

Opens the **Workspace** window displaying a list of source files.

4.1.2 Output

Opens the **Output** window displaying a message when using the debugger.

4.1.3 Breakpoints

Opens the **Breakpoints** window allowing the user to view and edit current breakpoints.

4.1.4 Command Line

Opens the **Command Line** window allowing the user to enter text-based commands to control the debugging platform. These commands can be piped in from a batch file, and the results piped out to a log file, allowing automatic tests to be performed.

4.1.5 Disassembly

Launches the **Set Address** dialog box allowing the user to enter the address that you wish to view.

4.1.6 IO

Opens the **IO** window allowing the user to view and modify the control register.

4.1.7 Labels

Launches the **Labels** window allowing the user to manipulate the current program's symbols (labels).

4.1.8 Locals

Opens the **Locals** window allowing the user to view and edit the values of the variables defined in the current function. The contents are blank unless the PC is within a C/C++ source-level function.

4.1.9 Memory...

Launches the **Set Address** dialog box allowing the user to specify a memory block and view format to display within a **Memory** window.

4.1.10 Performance Analysis

Launches the **Performance Analysis** window allowing the user to set up and view the number of times that particular sections of the user program have been called.

4.1.11 Profile

Opens the **Profile** window allowing the user to view the address and size of a function or a global variable, the number of times the function is called, and profile data.

4.1.12 Registers

Opens the **Registers** window allowing the user to view all the current CPU registers and their contents.

4.1.13 Status

Opens the **System Status** window allowing the user to view the debugging platform's current status and the current session and program names.

4.1.14 Trace

Opens the **Trace** window allowing the user to see the current trace information.

4.1.15 Watch

Opens the **Watch** window allowing the user to enter C/C++-source level variables and view and modify their contents.

4.1.16 TLB...

Opens the **Open TLB** dialog box allowing the user to enter the type of TLB that you wish to view.

4.1.17 Cache...

Opens the **Open Cache** dialog box allowing the user to enter the type of cache that you wish to view.

4.1.18 Simulated I/O

Opens the **Simulated I/O** window enabling the standard I/O and file I/O.

4.1.19 Stack Trace

Opens the **Stack Trace** window displaying the current stack trace information.

4.1.20 Coverage…

Opens the **Coverage** window allowing the user to view the coverage information.

4.1.21 Image...

Opens the **Image** window displaying the memory contents as images.

4.1.22 Waveform...

Opens the **Waveform** window displaying the memory contents as waveforms.

4.1.23 Trigger

Opens the **Trigger** window displaying trigger buttons to generate manual interrupts during simulation.

4.2 Options

The Options menu is used to change the settings for the debugging interface of the HEW and make the settings for the debugging platform.

4.2.1 Debug Settings…

Launches the **Debug Settings** dialog box allowing the user to modify the settings for the debugging interface of the HEW (not debugging platform dependent settings).

4.2.2 Radix

16 10 8 2 Cascades a menu displaying a list of radix in which the numeric values will be displayed and entered by default (without entering the radix prefix). The current radix has a toolbar button to its left is locked down.

For example, if the current radix is decimal then the number ten will be displayed as "`10`" and may be entered as "`10`", "`H’A`", "`0x0a`", etc.; if the current radix is hexadecimal then the number ten will be displayed as "`0A`" and entered as "`A`", "`D’10`", etc.

4.2.3 Simulator

System…:

Launches a **Simulator System** dialog box allowing the user to modify the debugging platform settings. Refer to section 5.24, Simulator System Dialog Box for more details.

Memory Resource…:

Opens the **Simulator Memory Resource** window allowing the user to view and edit the debugging platform's current memory map.

4.3 Debug

The Debug menu controls the execution of the user program in the debugging platform.

4.3.1 Reset CPU

Resets the user system hardware and sets the PC to the reset vector address.

4.3.2 Go

Starts executing the user program at the current PC.

4.3.3 Reset Go

Executes the user program from the reset vector address.

4.3.4 Go To Cursor

Starts executing the user program at the current PC and continues until the PC equals the address indicated by the current text cursor (not mouse cursor) position.

4.3.5 Set PC To Cursor

Changes the value of the Program Counter (PC) to the address at the row of the text cursor (not mouse cursor). Disabled if no address is available for the current row.

4.3.6 Run...

Launches the **Run Program** dialog box allowing the user to enter temporary breakpoints before executing the user program.

4.3.7 Step In

Executes a block of user program before breaking. The size of this block is normally a single instruction but may be set by the user to more than one instruction or a C/C++-source line (see also section 4.3.10, Step...). If a subroutine call is reached, then the subroutine will be entered and the view is updated to include its code.

4.3.8 Step Over

Executes a block of user program before breaking. The size of this block is normally a single instruction but can be set by the user to more than one instruction or a C/C++-source line (see also section 4.3.10, Step...). If a subroutine call is reached, then the subroutine will not be entered and sufficient user program will be executed to set the current PC position to the next line in the current view.

4.3.9 Step Out

Executes sufficient user program to reach the end of the current function and set the PC to the next line in the calling function before breaking.

4.3.10 Step...

Launches the **Step Program** dialog box allowing the user to modify the settings for stepping.

4.3.11 Step Mode

Specifies the **Step Mode** allowing the user to select a unit of stepping from **Auto** (automatic selection), **Assembly** (assembly instruction units), or **Source** (C/C++ source level).

4.3.12 Halt Program

Stops the execution of the user program.

4.3.13 Initialize

Disconnects the debugging platform and connects it again.

4.3.14 Disconnect

Disconnects the debugging platform.

4.3.15 Download Modules

Downloads the object program.

4.3.16 Unload Modules

Unloads the object program.

4.4 Memory

The Memory menu is used for aspects of the user program that access memory.

4.4.1 Search…

Launches the **Search Memory** dialog box allowing the user to specify the start and end addresses and the data value to be searched and to perform the search. Search conditions (match/unmatch and search direction) can also be specified.

4.4.2 Copy...

Launches the **Copy Memory** dialog box allowing the user to copy a block of the debugging platform's memory to an address within the same memory area. The blocks may overlap, in which case any data within the overlapped region of the source block will be overwritten. If a block of memory is highlighted in a **Memory** window, these will be automatically entered as the start and end addresses when the dialog box is displayed. Data in the source block can be compared with that in the destination while being copied.

4.4.3 Compare...

Launches the **Compare Memory** dialog box, allowing the user to select a start and an end address in the memory area, to check against another area in memory. If a block of memory is highlighted in a **Memory** window, these will be automatically entered as the start and end addresses when the dialog box is displayed.

4.4.4 Fill...

Launches the **Fill Memory** dialog box allowing the user to fill a block of the debugging platform's memory with a value. If a block of memory is highlighted in a **Memory** window, these will be automatically entered as the start and end addresses when the dialog box is displayed.

4.4.5 Refresh

Forces a manual update of the contents of all open **Memory** windows.

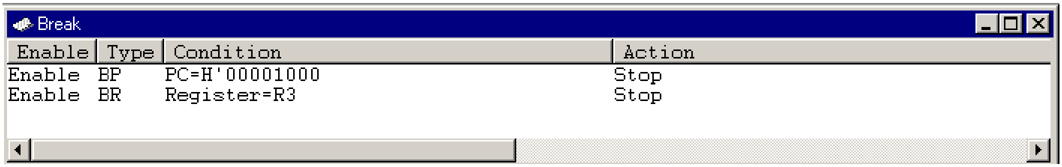
4.4.6 Configure Overlay...

Launches the **Overlay** dialog box. When the overlay function is used, the target section group can be selected in the dialog box.

Section 5 Windows and Dialog Boxes

This section describes types of windows and dialog boxes, the features that they support and the options available through their associated popup menu.

5.1 Break Window



Break

Enable	Type	Condition	Action
Enable	BP	PC=H'00001000	Stop
Enable	BR	Register=R3	Stop

Figure 5.1 Break Window

This window displays all of the specified breakpoints. Items that can be displayed are listed below.

[Enable] Displays whether the breakpoint is enabled or disabled.
Enable: Valid
Disable: Invalid

[Type] Displays break types
BP: PC break
BA: Break access
BD: Break data
BR: Break register (Register name)
BS: Break sequence
BCY: Break cycle

[Condition] Displays the conditions that satisfies a break condition. The contents displayed differ from the type of the break. When the type of the break is BR, the register name is displayed, and when the type of the break is BCY, the number of cycles is displayed.
BP: PC = Program counter (Corresponding file name, line, and symbol name)
BA: Address = Address (Symbol name)
BD: Address = Address (Symbol name)
BR: Register = Register name
BS: PC = Program counter (Corresponding file name, line, and symbol name)
BCY: Cycle = Number of cycles (displayed in hexadecimal)

[Action] Displays the operation of the simulator/debugger when a break condition is satisfied.
Stop: Execution halts
File Input (file name) [File state: Memory data is read from file]
File Output (file name) [File state: Memory data is written to file]

Interrupt (Interrupt type/priority): Interrupt processing
Only for SH3-DSP (Interrupt type 1, interrupt type 2/priority) is displayed.

When a breakpoint is double-clicked in this window, the **Set Break** 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.

5.1.1 Add...

Sets breakpoints. Clicking this item will open the **Set Break** dialog box and break conditions can be specified.

5.1.2 Edit...

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

5.1.3 Enable

Enables the selected breakpoint(s).

5.1.4 Disable

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

5.1.5 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 5.1.4, Disable).

5.1.6 Delete All

Removes all breakpoints.

5.1.7 Go to Source

Only enabled when one breakpoint is selected. Opens **Source** or **Disassembly** window at address of breakpoint.

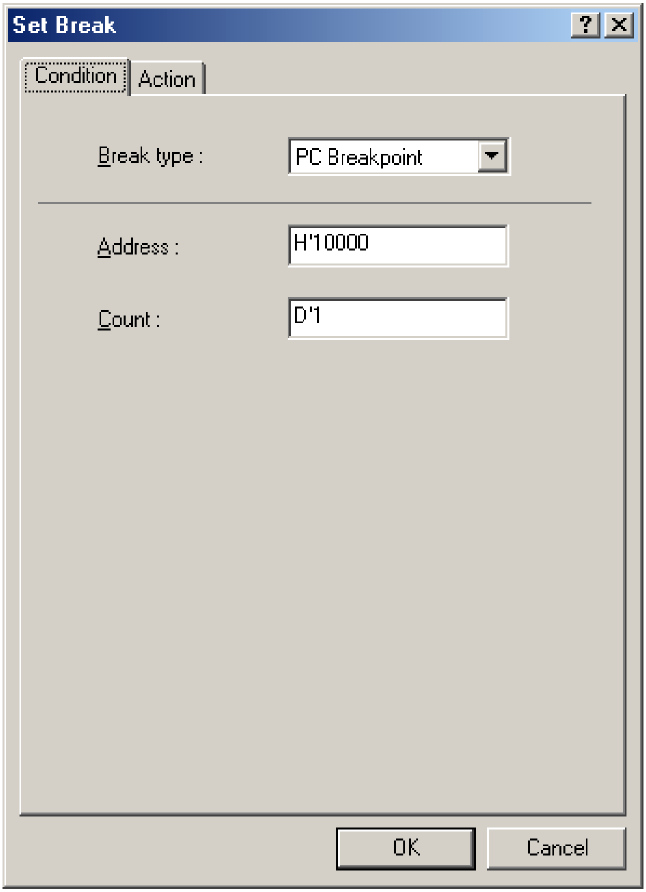
5.1.8 Close File

Closes the selected File Input or File Output data file and resets the address to read the file.

5.1.9 Close All Files

Closes all the selected File Input and File Output files and resets the address to read the file.

5.2 Set Break Dialog Box (Condition Sheet)



Break type : PC Breakpoint

Address : H'10000

Count : D'1

Figure 5.2 Set Break Dialog Box (Condition Sheet)

This dialog box specifies break conditions.

A break type to be set is specified using the radio buttons in the **[Break type]** box. Items that can be specified are listed below.

[PC Breakpoint] Up to 255 breakpoints can be specified

[Address] Address where a break occurs

[Count] Number of times that a specified instruction is fetched (when the prefix is omitted, values must be input and are displayed in decimal) (1 to 16383, default: 1)

[Break Access] Up to two addresses can be specified

[Start address] Start address of memory where a break occurs if the memory is accessed

[End address] End address of memory where a break occurs if the memory is accessed (If no data is input, only the start address is break range)

[Access type] Read, Write, or Read/Write

[Break Data] Up to eight values of data can be specified

[Start address] Address of memory where a break occurs

[Data] Data value that causes a break

[Size] Data size

[Option] Data match/mismatch

[Break Register] Up to eight register names can be specified

[Register] Register name where break conditions are specified

[Size] Data size

[Data] Data value that causes a break (If no data is input, a break occurs whenever data is written to the register)

[Option] Data match/mismatch

[Break Sequence]Only one address can be specified

[Address1] to Pass addresses that are the conditions to generate a break.
[Address8] (All eight breakpoints do not have to be set.)

[Break Cycle] Up to 255 cycles can be specified

[Cycle] Number of cycles to determine a break (H’1 to H’FFFFFFFF). A condition will be satisfied by a number of cycles of [Cycle] x n. However, the specified number of cycles and the number of cycles that satisfied the condition may be different.

[Count] Number of times breaks will occur

[ALL] A break will occur whenever a condition is satisfied.

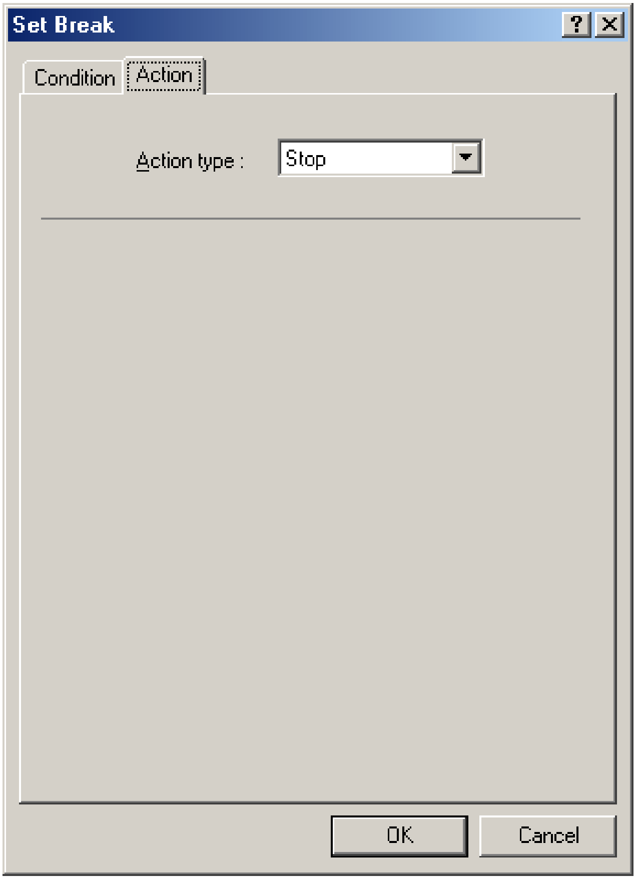
[Times] (when the prefix is omitted, values must be input and are displayed in hexadecimal) (1 to 65535)
A break will occur only when the number of times the conditions is satisfied is equal to or below the value specified for [Times].

When **[PC Breakpoint]** and **[Break Sequence]** is selected, if an overloaded function or class name including a member function is specified in address, the **Select Function** dialog box opens. In the dialog box, select a function. For details, refer to the HEW Debugger User’s Manual.

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

Note: For the SH3-DSP series, specify values within the range H'A5000000 to H'A501FFFF (X and Y memory virtual addresses, corresponding to physical addresses H'05000000 to H'0501FFFF) as the [Start address] and [End address] in [Break Access] and as the [Start address] in [Break Data] for X or Y memory accesses by the MOVX or MOVY instruction.

5.3 Set Break Dialog Box (Action Sheet)



Action type : Stop

Figure 5.3 Set Break Dialog Box (Action Sheet)

This dialog box specifies the operation when a break condition is satisfied. First set the operation in [Action type]. The following show the items that can be set in each operation.

[Stop] Stops the operation of user program when a condition is satisfied. There is no item to be set.

[File Input] The data of a specified file is referred to and its contents are written to the specified memory when a condition has been satisfied

- [Input file] Data file to refer to.
  When the simulator/debugger has reached referring to the end of a file, it will repeat referring to the beginning of the same file.
- [Address] Memory address where data should be written to.
- [Data size] Data size to be written to (1/2/4/8).
- [Count] Number of data to be written to (when the prefix is omitted, values must be input and are displayed in decimal) (H’1 to H’FFFFFFFF).

[File Output] The contents of a specified memory is written to the specified file when a condition has been satisfied.

- [Output file] Data file to be written to.
- [Append] When a existing file is specified for the [Output File],
- [Address] Memory address to read data to.
- [Data size] Size of one data to refer to (1/2/4/8).
- [Count] Number of data to refer to (when the prefix is omitted, values must be input and are displayed in decimal) (H’1 to H’FFFFFFFF).
- [Option] Data match/mismatch

[Interrupt] Interrupts the program when a condition has been satisfied. For details, refer to section 3.13, Pseudo-Interrupts.

- [Interrupt type1] Specifies the following values for each CPU (when the prefix is omitted, values must be input and are displayed in hexadecimal)
  - SH-1, SH-2, and SH2-DSP series
    Interrupt vector number (H’0 to H’FF)
  - SH-3, SH-4, and SH3-DSP series
    INTEVT (H’0 to H’FFF)
- [Interrupt type2] Only the SH3-DSP series can specify the following (when the prefix is omitted, values must be input and are displayed in hexadecimal):
  INTEVT2 (H’0 to H’FFF)
- [Priority] Interrupt priority (when the prefix is omitted, values must be input and are displayed in hexadecimal) (H’0 to H’11)
  When H'10 is specified, the interrupt is always accepted regardless of the I bit in SR, but is masked by the BL bit in

SR.
When H'11 is specified, the interrupt is always accepted regardless of the I and BL bits in SR.

Note: When the same file is specified for multiple File Inputs, the simulator/debugger will read data from the file in the order conditions are satisfied. When the same file is specified for multiple File Outputs, the simulator/debugger will write data to the file in the order conditions are satisfied. However, when File Input and File Output specify the same file, only the operation for the first condition satisfied is valid.

5.4 Command Line Window

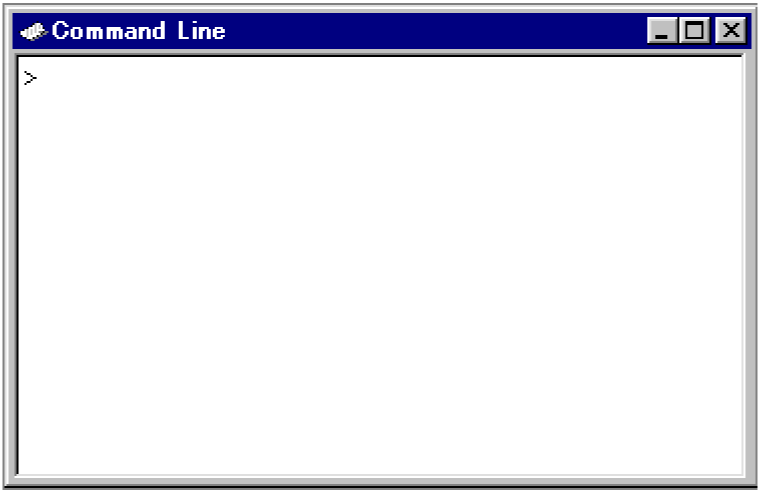


Figure 5.4 Command Line Window

Allows the user to control the debugging platform by sending text-based commands instead of the window menus and commands. It is useful if a series of predefined commands need to be sent to the debugging platform by calling them from a batch file and, optionally, recording the output in a log file. The command can be executed by pressing 'Enter' after the command is input to the last line (or, the **Enter** button in the right of the text box is clicked). For information about the available commands, refer to the on-line help.

If available, the window title displays the current batch and log file names separated by colons.

Pressing the Ctrl + ↑ or Ctrl + ↓ keys on the last line displays the previously executed command line.

The functionality of the toolbar buttons is identical to the popup menu options shown below.

Clicking the right mouse button on the **Command Line** window displays the popup menus. The menus include the following options.

5.4.1 Set Batch File...

Launches the **Set Batch File** dialog box, allowing the user to enter the name of a command file (*.hdc). Clicking the **[Play]** button closes the dialog box and the specified command file runs. Clicking the **[OK]** button displays the specified command file name as the window title. Clicking the **[Cancel]** button closes the dialog box without modifying the setting.

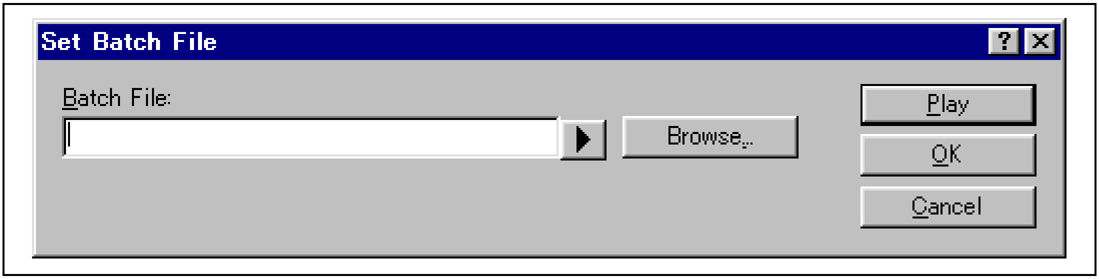


Figure 5.5 Set Batch File Dialog Box

5.4.2 Play

Runs the command file (*.hdc) selected in the **Set Batch File** dialog box. It is displayed in a recessed state while the batch file is running and can be used to stop an executing batch file and return control to the user.

5.4.3 Stop

Stops the execution of a command. The button becomes valid during the execution of the command.

5.4.4 Set Log File...

Launches the **Open Log File** dialog box, allowing the user to enter the name of a log file (*.log). The logging option is automatically set and the name of the file shown on the window title bar.

Opening a previous log file will ask the user if they wish to append or over-write the current log.

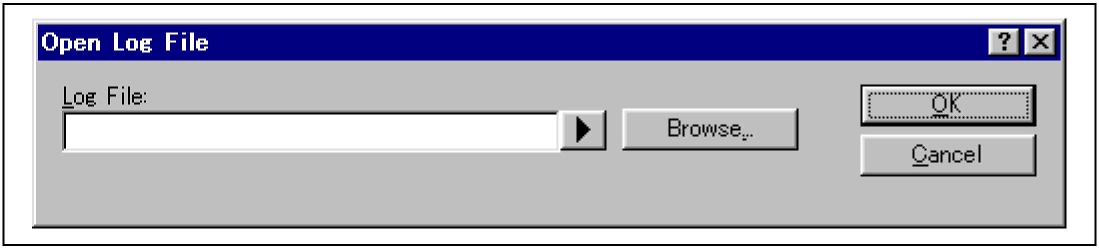


Figure 5.6 Open Log File Dialog Box

5.4.5 Logging

Toggles logging to file on and off. When logging is active, the button becomes effective. Note that the contents of the log file cannot be viewed until logging is completed, or temporarily disabled by clearing the check box. Re-enabling logging will append to the log file.

5.4.6 Browse…

Displays the **Browse** dialog box. This dialog box pastes the full path of the selected file to the cursor location. This option can only be used when the cursor is at the last line.

5.4.7 Placeholder

Pastes the selected placeholder to the cursor location. This function is only available when the cursor is located on the last line.

5.4.8 Select All

Selects all contents output in the **Command Line** window.

5.4.9 Copy

Only available if a block of text is highlighted. This copies the highlighted text into the Windows® clipboard, allowing it to be pasted into other applications.

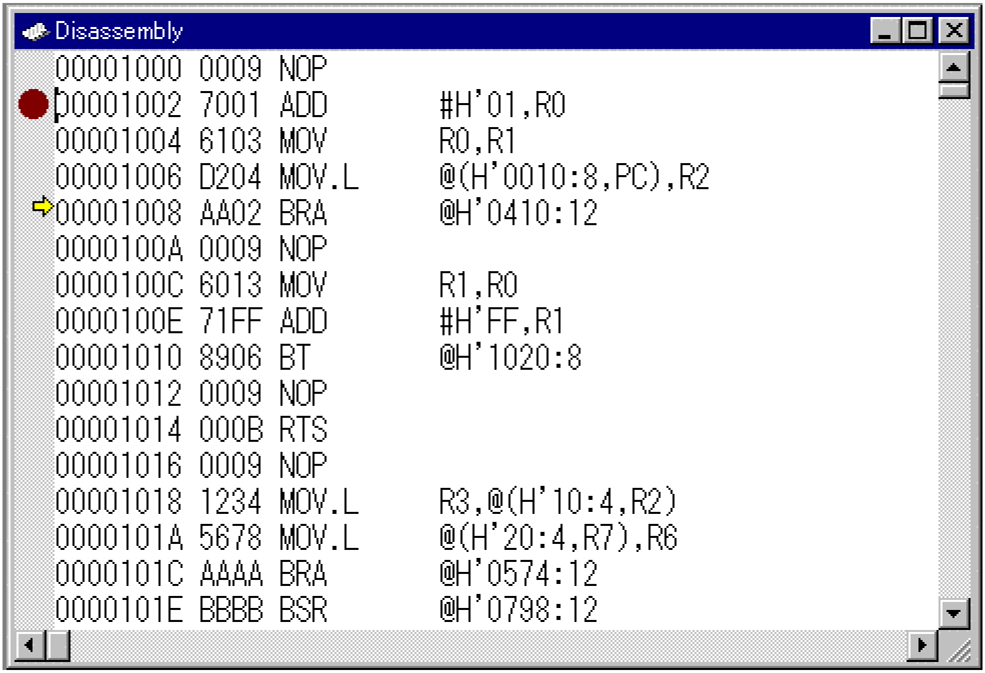
5.4.10 Paste

This option pastes the contents of the Windows® clipboard to the current cursor location. This option can only be used when the cursor is at the last line.

5.5 Disassembly Window

This window is used to display code at the assembly-language level.

Assembly-language information is obtained by disassembling the memory contents, and may be edited or viewed directly from memory without requiring debug information from the object file.



```
 00001000 0009 NOP
●00001002 7001 ADD      #H'01,R0
 00001004 6103 MOV      R0,R1
 00001006 D204 MOV.L    @(H'0010:8,PC),R2
⇨00001008 AA02 BRA      @H'0410:12
 0000100A 0009 NOP
 0000100C 6013 MOV      R1,R0
 0000100E 71FF ADD      #H'FF,R1
 00001010 8906 BT       @H'1020:8
 00001012 0009 NOP
 00001014 000B RTS
 00001016 0009 NOP
 00001018 1234 MOV.L    R3,@(H'10:4,R2)
 0000101A 5678 MOV.L    @(H'20:4,R7),R6
 0000101C AAAA BRA      @H'0574:12
 0000101E BBBB BSR      @H'0798:12
```

Figure 5.7 Disassembly Window

This window displays address information, addresses, instruction code, and instruction mnemonic. The following are the address information items:

🟦: A bookmark is set.

●: A PC Break is set.

⇨: PC location

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

5.5.1 View Source

Opens the **Source** window including the source program corresponding to the text cursor (not the mouse cursor) position. Only available when the selected source line is valid.

5.5.2 Go to cursor

Commences to execute the user program starting from the current PC address. The program will continue to run until the PC reaches the address indicated by the text cursor (not the mouse cursor) or another break condition is satisfied. PC breakpoint is used for this function. The function is not available when 255 PC breakpoints have already been specified.

5.5.3 Set Address...

Displays the **Set Address** dialog box. Specify the address from which the display should start.

5.5.4 Set PC Here

Changes the value of the PC to the address indicated by the text cursor (not the mouse cursor).

5.5.5 Edit...

Launches the **Assembler** dialog box allowing the user to modify the instruction at that address. Note that changes to the machine code do not modify the source file, and any changes will be lost at the end of the session.

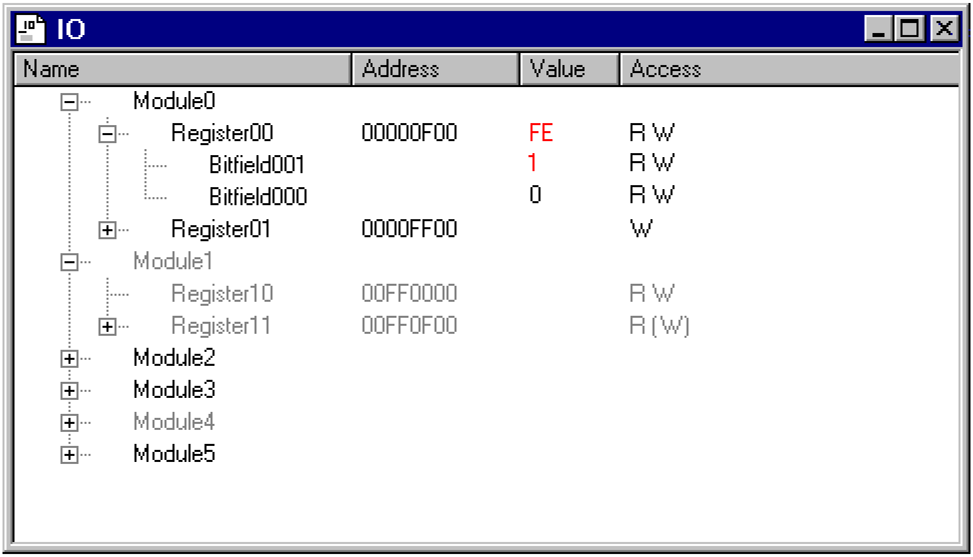
5.5.6 Code Bytes

Selects whether or not the instruction code is displayed.

5.5.7 Toggle Breakpoint

Enables or disables PC breakpoints.

5.6 IO Window



Name	Address	Value	Access
Module0			
Register00	00000F00	FE	R W
Bitfield001		1	R W
Bitfield000		0	R W
Register01	0000FF00		W
Module1			
Register10	00FF0000		R W
Register11	00FF0F00		R (W)
Module2			
Module3			
Module4			
Module5			

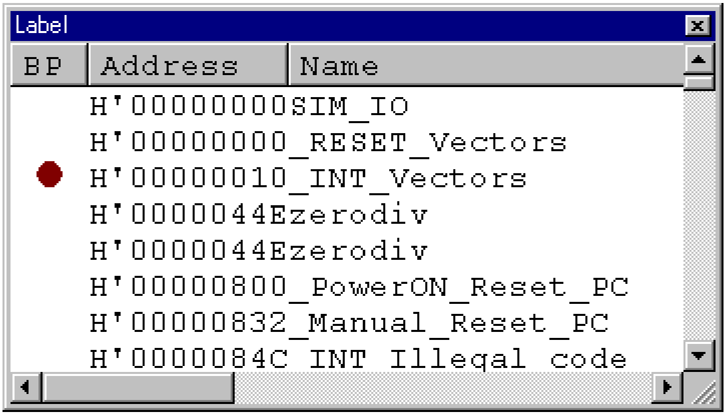
Figure 5.8 IO Window

Refers to and sets the values of control registers.

Double-clicking the plus mark (+) or minus mark (-), or entering the plus key (+) or minus key (-) will expand or compress the information of control registers, and then display it.

Double-clicking the **Name** column displays the Edit Register dialog box. The value of the control registers can be modified.

5.7 Label Window



Label

BP	Address	Name
	H'00000000	SIM_IO
	H'00000000	_RESET_Vectors
●	H'00000010	_INT_Vectors
	H'0000044E	zerodiv
	H'0000044E	zerodiv
	H'00000800	_PowerON_Reset_PC
	H'00000832	_Manual_Reset_PC
	H'0000084C	_INT_Illegal_code

Figure 5.9 Label Window

You can view symbols sorted either alphabetically (by ASCII code) or by address value by clicking on the respective column heading.

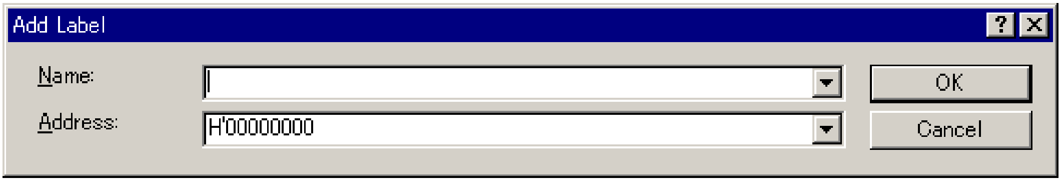
It supports column-specific double-click actions:

- BP - Sets or cancels a PC breakpoint at that address.

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

5.7.1 Add…

Launches the **Add Label** dialog box:



Add Label

Name:

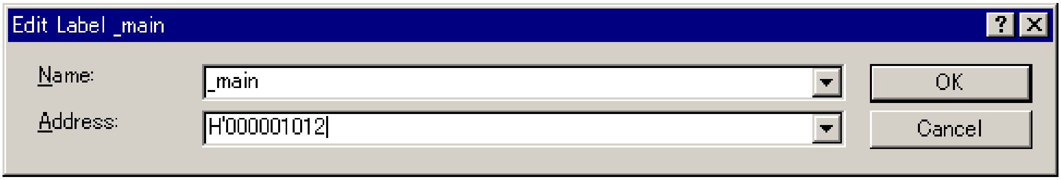
Address: H'00000000

Figure 5.10 Add Label Dialog Box

Enter the new label name into the **Name** field and the corresponding value into the **Address** field and press **[OK]**. The **Add Label** dialog box closes and the label list is updated to show the new label. When an overloaded function or a class name is entered in the **Address** field, the **Select Function** dialog box opens for you to select a function. For details, refer to the HEW Debugger User’s Manual.

5.7.2 Edit…

Launches the **Edit Label** dialog box:



Edit Label _main

Name: _main

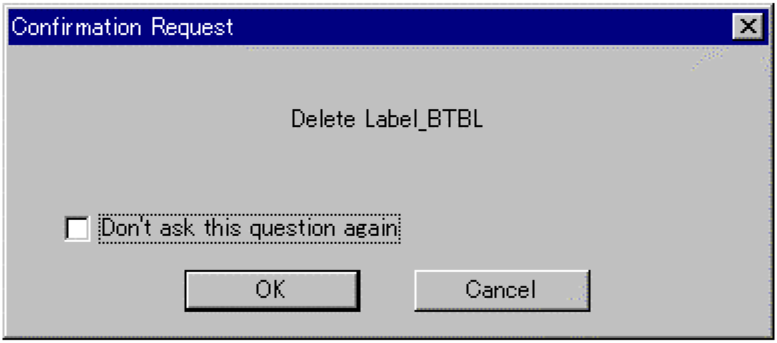
Address: H'000001012

Figure 5.11 Edit Label Dialog Box

Edit the label name and value as required and then press **[OK]** to save the modified version in the label list. The list display is updated to show the new label details. When an overloaded function or a class name is entered in the **Address** field, the **Select Function** dialog box opens for you to select a function. For details, refer to the HEW Debugger User's Manual.

5.7.3 Delete

Deletes the currently selected label from the symbol list. Alternatively use the **Delete** accelerator key. A confirmation message box appears:



Confirmation Request

Delete Label_BTBL

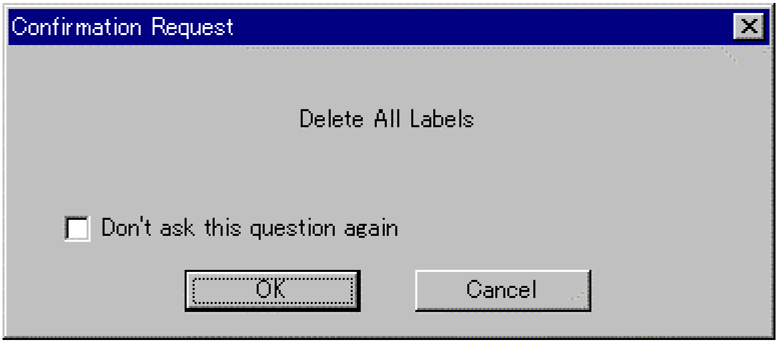
□ Don't ask this question again

Figure 5.12 Message Box for Confirming Label Deletion

If you click **[OK]**, the label is removed from label list and the window display is updated. If the message box is not required then do not select the **Delete Label** option of the **Confirmation** sheet in the **HEW Options** dialog box.

5.7.4 Delete All

Deletes all the labels from the list. A confirmation message box appears:



Confirmation Request

Delete All Labels

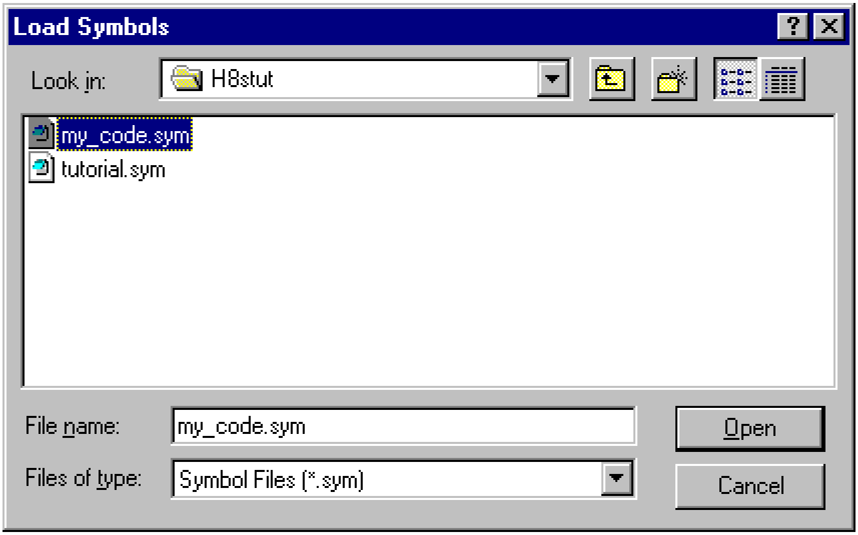
□ Don't ask this question again

Figure 5.13 Message Box for Confirming All Label Deletion

If you click **[OK]**, all the labels are removed from the HEW system’s symbol table and the list display will be cleared. If the message box is not required then do not select the **Delete All Labels** option of the **Confirmation** sheet in the **HEW Options** dialog box.

5.7.5 Load…

Merges a symbol file into HEW's current symbol table. The **Load Symbols** dialog box opens:



Load Symbols

Look in: H8stut

my_code.sym
tutorial.sym

File name: my_code.sym

Files of type: Symbol Files (*.sym)

Figure 5.14 Load Symbols Dialog Box

The dialog box operates like a standard Windows® **Open File** dialog box; select the file and click **[Open]** to start loading. The standard file extension for symbol files is ".sym". When the symbol loading is complete a confirmation message box may be displayed showing how many symbols have been loaded (this can be switched off in the **Confirmation** sheet on the **HEW Options** dialog box).

5.7.6 Save

Saves HEW's current symbol table to a symbol file.

5.7.7 Save As…

The **Save Symbols** dialog box operates like a standard Windows® **Save File As** dialog box. Enter the name for the file in the **File name** field and click **[Open]** to save HEW's current label list to a symbol file. The standard file extension for symbol files is ".sym".

See appendix B of the HEW Debugger User's Manual for symbol file format.

5.7.8 Find…

Launches the **Find Label** dialog box:

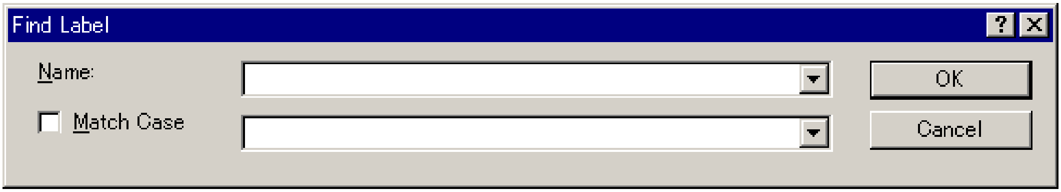


Figure 5.15 Find Label Dialog Box

Enter all or part of the label name that you wish to find into the edit box and click **[OK]** or press **ENTER**. The dialog box closes and HEW searches the label list for a label name containing the text that you entered.

Note: Only the label is stored by 1024 characters of the start, therefore the label name must not overlap mutually in 1024 characters or less. Labels are case sensitive.

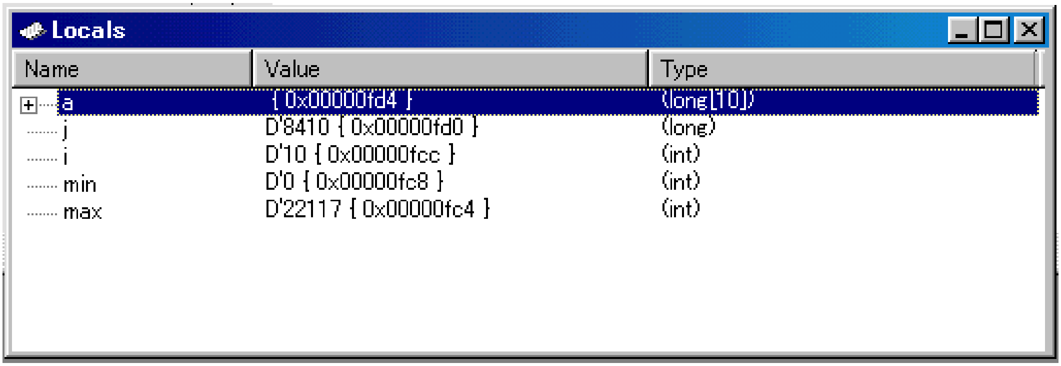
5.7.9 Find Next

Finds the next occurrence of the label containing the text that you entered.

5.7.10 View Source

Opens the **Source** or **Disassembly** window containing the address corresponding to the label.

5.8 Locals Window



Name	Value	Type
⊞ a	{ 0x00000fd4 }	(long[10])
j	D'8410 { 0x00000fd0 }	(long)
i	D'10 { 0x00000fcc }	(int)
min	D'0 { 0x00000fc8 }	(int)
max	D'22117 { 0x00000fc4 }	(int)

Figure 5.16 Locals Window

Allows the user to view and modify the values of all the local variables. The contents of this window are blank unless the current PC can be associated to a function containing local variables in the source files *via* the debugging information available in the object file.

The following items are displayed.

[Name] Name of the variable

[Value] Value, assigned location.
The assigned location is enclosed by { }.

[Type] Displays the type of variable

The variables are listed with a plus indicating that the information may be expanded by double-clicking on the variable name, and a minus indicating that the information may be collapsed. Alternatively, the plus and minus keys may be used. For more information on the display of information, refer to the HEW Debugger User’s Manual.

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

5.8.1 Edit Value...

Launches a dialog box to modify the selected variable’s value.

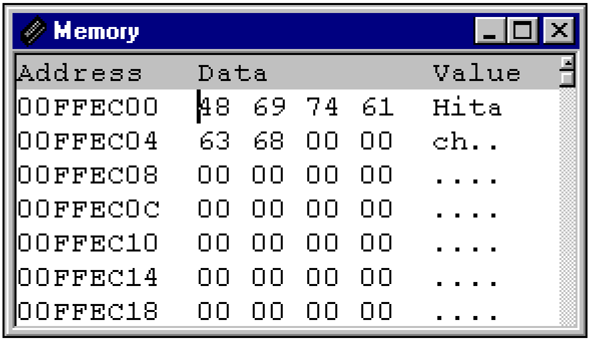
5.8.2 Radix

Changes the radix for the selected local variable display.

5.8.3 Copy

Only available if a block of text is highlighted. This copies the highlighted text into the Windows® clipboard, allowing it to be pasted into other applications.

5.9 Memory Window



```
Address   Data          Value
00FFEC00  48 69 74 61   Hita
00FFEC04  63 68 00 00   ch..
00FFEC08  00 00 00 00   ....
00FFEC0C  00 00 00 00   ....
00FFEC10  00 00 00 00   ....
00FFEC14  00 00 00 00   ....
00FFEC18  00 00 00 00   ....
```

Figure 5.17 Memory Window

Allows the user to view and modify the contents of the debugging platform's memory. Memory may be viewed in ASCII, byte, word, longword, single-precision floating-point, and double-precision floating-point formats, and the title bar indicates the current view style and the address shown as the offset from the previous label (symbol).

The contents of memory can be edited by double-clicking on a data item. The latter will launch the **Edit** dialog box, allowing the user to enter a new value using a complex expression. If the data at that address cannot be modified (i.e. within ROM or guarded memory) then the message "Invalid address value" is displayed.

Double-clicking within the Address column will launch the **Set Address** dialog box, allowing the user to enter an address. Clicking the **[OK]** button will update the window so that the address entered in the **Set Address** dialog box is the first address displayed in the top-left corner.

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

5.9.1 Lock Refresh

Controls the **Memory** window so that it is not automatically updated when user program execution stops.

5.9.2 Refresh

Forcibly updates the **Memory** window contents.

5.9.3 Start Address…

Launches the **Set Address** dialog box, allowing the user to enter new start and end addresses. The window will be updated so that this is the first address displayed in the top-left corner. When an overloaded function or a class name including a member function is entered, the **Select Function** dialog box opens for you to select a function. For details, refer to section 7.3, Supporting Duplicate Labels in the HEW Debugger User's Manual. The size to display data can be specified.

5.9.4 Format…

Displays the **Format Memory Display** dialog box. The size to display data, the format to display data, and the font used to display data can be specified.

5.9.5 Search…

Launches the **Search Memory** dialog box, allowing the user to search a block of the debugging platform's memory for a specified data value. If a block of memory is highlighted, the start and end fields in the dialog box will be set automatically with the start and end addresses corresponding to the highlighted block, respectively. Search conditions (match/unmatch and search direction) can also be specified. The **Memory** window is updated to start with the address which holds the data that satisfies the search conditions.

5.9.6 Search Next

Only available when data has been found with the **Search...** option. This option starts search from the address following the one found with the last search operation.

5.9.7 Copy…

Launches the **Copy Memory** dialog box, allowing the user to copy a block of memory within the debugging platform to another location within the same memory space. The blocks may overlap. The start and end fields may be set similarly to the **Search** option (see section 5.9.5, Search...). Data in the source block can be compared with that in the destination while being copied.

5.9.8 Compare...

Launches the **Compare Memory** dialog box, allowing the user to select a start and an end address in the memory area, to check against another area in memory. If a block of memory is highlighted in a **Memory** window, these will be automatically set as the start and end addresses when the dialog box is displayed.

Similar to Verify memory, but compares two blocks in memory.

5.9.9 Fill…

Launches the **Fill Memory** dialog box, allowing the user to fill a block of the debugging platform's memory with a specified value. The start and end fields may be set similarly to the **Search** option (see section 5.9.5, Search...).

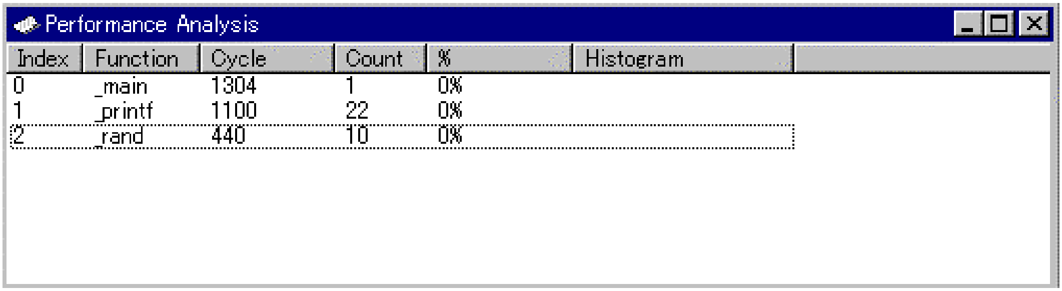
5.9.10 Save…

Launches the **Save Memory As** dialog box, allowing the user to save a block of the debugging platform's memory in an S-Record file (*.mot). The start and end fields may be set similarly to the **Search** option (see section 5.9.5, Search...).

5.9.11 Load…

Launches the **Load Memory** dialog box, allowing the user to load to the debugging platform's memory from an S-Record file (*.mot) without deleting the current debug information. The offset field may be used to move the address values specified in the file to a different set of addresses. The optional verify flag can be used to check that the information has been downloaded correctly.

5.10 Performance Analysis Window



Index	Function	Cycle	Count	%	Histogram
0	_main	1304	1	0%	
1	_printf	1100	22	0%	
2	_rand	440	10	0%	

Figure 5.18 Performance Analysis Window

This window displays the number of execution cycles required for the specified functions.

The number of execution cycles can be obtained from the difference between the total number of execution when the target function is called and that when execution returns from the function.

The following items are displayed:

[Index] Index number of the set condition

[Function] Name of the function to be measured (or the start address of the function)

[Cycle] Total number of instruction execution cycles

[Count] Total number of calls for the function

[%] Ratio of execution cycle count required for the function to the execution cycle count required for the whole program

[Histogram] Histogram display of the above ratio

Double-clicking a function to be evaluated displays the **Performance Option** dialog box. In this dialog box, functions can be modified.

A popup menu containing the following options is available by right-clicking within the view area:

5.10.1 Add Range...

Adds a new function to be evaluated. Clicking this option launches the **Performance Option** dialog box, allowing the user to add a function. The **Performance Option** dialog box can also be opened by the Insert key.

5.10.2 Edit Range...

Only enabled when the highlighting bar is on a user-defined range. Launches the **Performance Option** dialog box, allowing the user to modify the range's settings. The **Performance Option** dialog box can also be opened by the Enter key.

5.10.3 Reset Counts/Times

Clears the current performance analysis data.

5.10.4 Enable Analysis

Toggles the collection of performance analysis data. When performance analysis is active, a check mark is shown to the left of the text.

5.10.5 Delete Range

Only enabled when the highlighting bar is on a user-defined range. Deletes the range and immediately recalculates the data for the other ranges. The range can also be deleted by the Delete key.

5.10.6 Delete All Ranges

Deletes all the current user-defined ranges, and clears the performance analysis data.

5.11 Performance Option Dialog Box

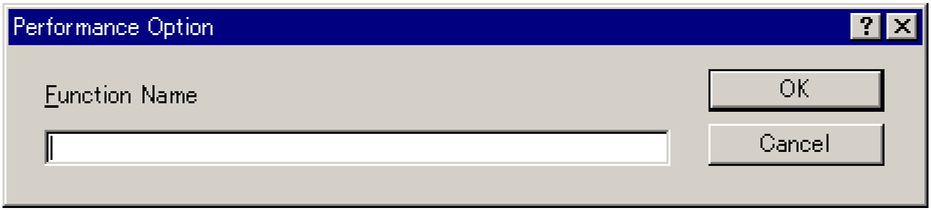


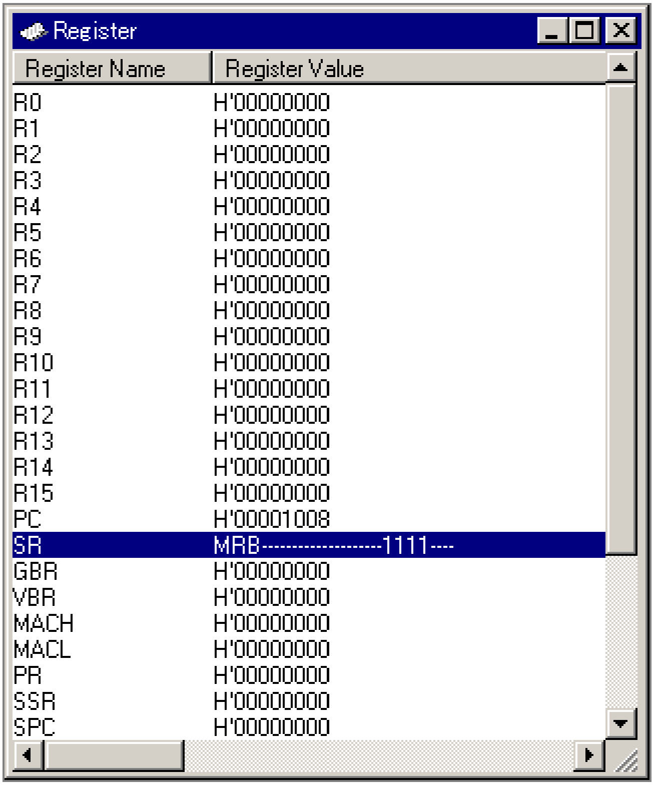
Figure 5.19 Performance Option Dialog Box

This dialog box specifies functions (including labels) to be evaluated. Evaluation results are displayed in the **Performance Analysis** window.

Note that when an overloaded function or a class name including a member function is specified, the **Select Function** dialog box opens. In the dialog box, select a function. For details, refer to section 7.3, Supporting Duplicate Labels in the HEW Debugger User’s Manual.

Clicking the **[OK]** button stores the setting. Clicking the **[Cancel]** button closes this dialog box without setting the function to be evaluated.

5.12 Register Window



Register Name	Register Value
R0	H'00000000
R1	H'00000000
R2	H'00000000
R3	H'00000000
R4	H'00000000
R5	H'00000000
R6	H'00000000
R7	H'00000000
R8	H'00000000
R9	H'00000000
R10	H'00000000
R11	H'00000000
R12	H'00000000
R13	H'00000000
R14	H'00000000
R15	H'00000000
PC	H'00001008
SR	MRB--------------------1111----
GBR	H'00000000
VBR	H'00000000
MACH	H'00000000
MACL	H'00000000
PR	H'00000000
SSR	H'00000000
SPC	H'00000000

Figure 5.20 Register Window

Allows the user to view and modify the current register values.

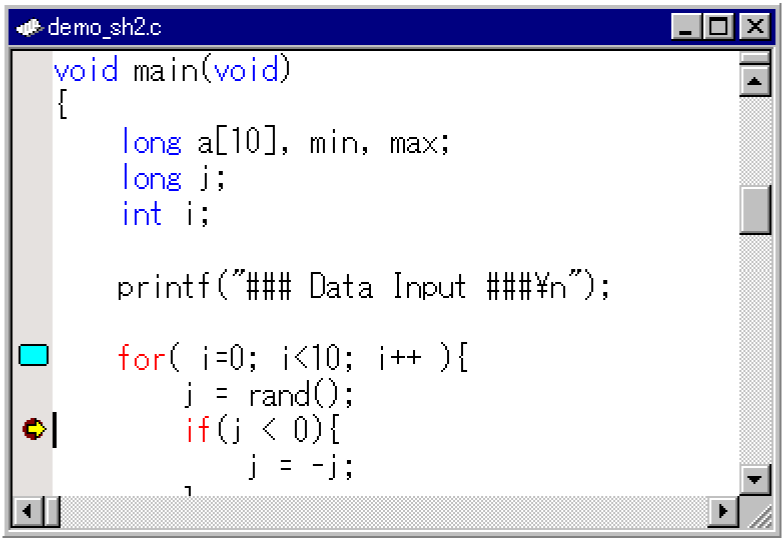
A popup menu containing the following options is available by right-clicking within the window:

5.12.1 Edit…

Launches the **Edit Register** dialog box, allowing the user to set the value of the register indicated by the text cursor (not mouse cursor).

5.13 Source Window

The **Source** window can be used to view any source file that was included within the object file’s debug information - this may be C/C++ and assembly language. This window also displays the coverage information.



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

    printf("### Data Input ###¥n");

    for( i=0; i<10; i++ ){
        j = rand();
        if(j < 0){
            j = -j;
        }
```

Figure 5.21 Source Window

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

🟦: A bookmark is set.

●: A PC Break is set.

⇨: PC location

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

5.13.1 Toggle Breakpoint

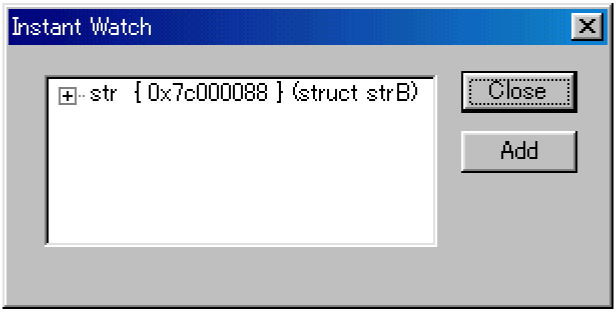
Sets or removes PC breakpoints.

5.13.2 Enable/Disable Breakpoint

Enables or disables PC breakpoints.

5.13.3 Instant Watch...

Opens the **Instant Watch** dialog box and displays the variable at the cursor location.



⊞ str { 0x7c000088 } (struct strB)

Figure 5.22 Instant Watch Dialog Box

"+" shown to the left of the variable name indicates that the information may be expanded by clicking on the variable name, and "-" indicates that the information may be collapsed. Clicking **[Add]** registers the variable in the **Watch** window. Clicking **[Close]** closes the window without registering the variable in the **Watch** window.

5.13.4 Go To Cursor

Commences to execute the user program starting from the current PC address. The program will continue to run until the PC reaches the address indicated by the text cursor (not the mouse cursor) or another break condition is satisfied. PC breakpoint is used for this function. The function is not available when 255 PC breakpoints have already been specified.

5.13.5 Set PC Here

Changes the value of the PC to the address indicated by the text cursor (not the mouse cursor).

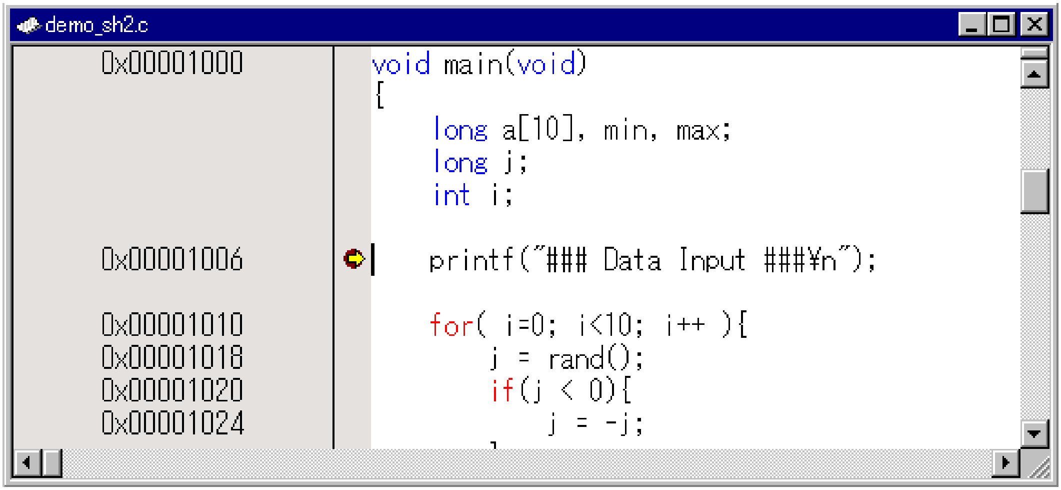
5.13.6 Go to Disassembly

Opens a **Disassembly** window showing code disassembled from the address that corresponds to the current line of source code.

5.14 Source Address Column

When a program is downloaded, the **Source** window display is updated to the current source file addresses. The source file addresses are displayed in the left part of the **Source** window.

These addresses are helpful when setting the PC value or a breakpoint. Figure 5.23 shows the **Source** window and the **address** column.



```
0x00001000   void main(void)
             {
                 long a[10], min, max;
                 long j;
                 int i;

0x00001006      printf("### Data Input ###¥n");

0x00001010      for( i=0; i<10; i++ ){
0x00001018          j = rand();
0x00001020          if(j < 0){
0x00001024              j = -j;
```

Figure 5.23 Source Window and Address Column

Use the following procedure to display or close the address column for all files:

1. Right-click the mouse in the editor window.
2. Click the [Define Column Format...] menu item.
3. The Global Editor Column States dialog box will appear.
4. Each check box specifies whether the corresponding column is displayed. If the check box is selected, the column is displayed. If the check box is shaded, the column is displayed for some files and not displayed in the other files.
5. Click the [OK] button to update the display with the new settings.

For a single file, use the following procedure to display or close the address column:

1. Right-click the mouse in the editor window of the column that needs to be changed, and an editor popup menu will appear.
2. Click the [Column] menu item to display a list of the column names. When a check mark is shown to the left of the column name, the column is displayed. Clicking on a column name can display or close the column.

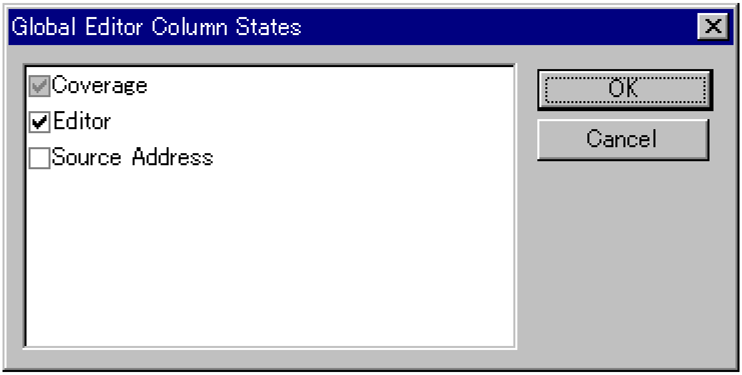


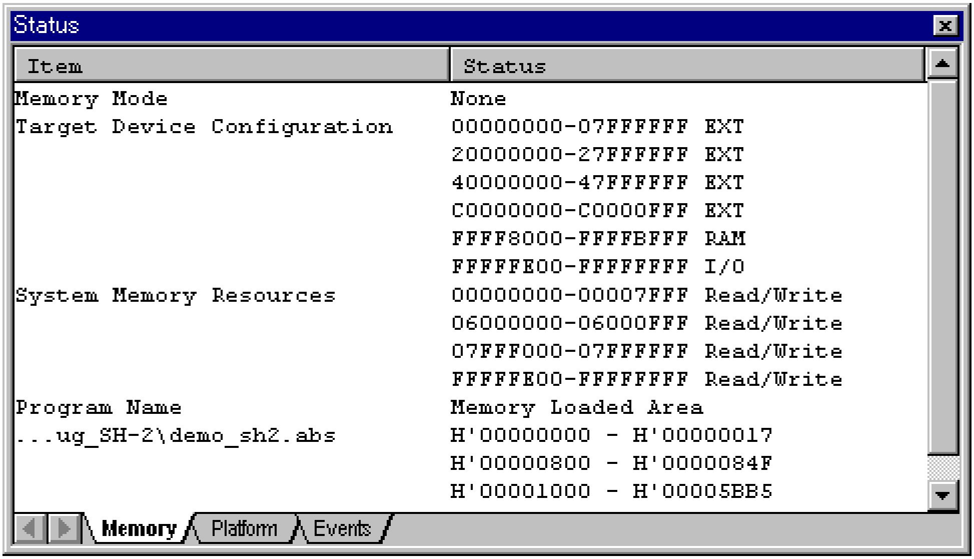
Figure 5.24 Global Editor Column States Dialog Box

5.15 Debugger Column

Each component can add a specific column in the **Disassembly** and **Source** windows. A typical example of a debugger column is the coverage column that displays the code coverage in a graphical form during debugging. Another example is the target component column that shows the hardware breakpoints set in the user system.

Right-clicking on a column displays a popup menu specifically for that column. The menu items differ between columns. The operation when a column is double-clicked depends on the column. For example, double-clicking on the target column will specify a hardware breakpoint.

5.16 Status Window



Item	Status
Memory Mode	None
Target Device Configuration	00000000-07FFFFFF EXT
	20000000-27FFFFFF EXT
	40000000-47FFFFFF EXT
	C0000000-C0000FFF EXT
	FFFF8000-FFFFBFFF RAM
	FFFFFE00-FFFFFFFF I/O
System Memory Resources	00000000-00007FFF Read/Write
	06000000-06000FFF Read/Write
	07FFF000-07FFFFFF Read/Write
	FFFFFE00-FFFFFFFF Read/Write
Program Name	Memory Loaded Area
...ug_SH-2\demo_sh2.abs	H'00000000 - H'00000017
	H'00000800 - H'0000084F
	H'00001000 - H'00005BB5

Figure 5.25 Status Window

Allows the user to view the current status of the debugging platform.

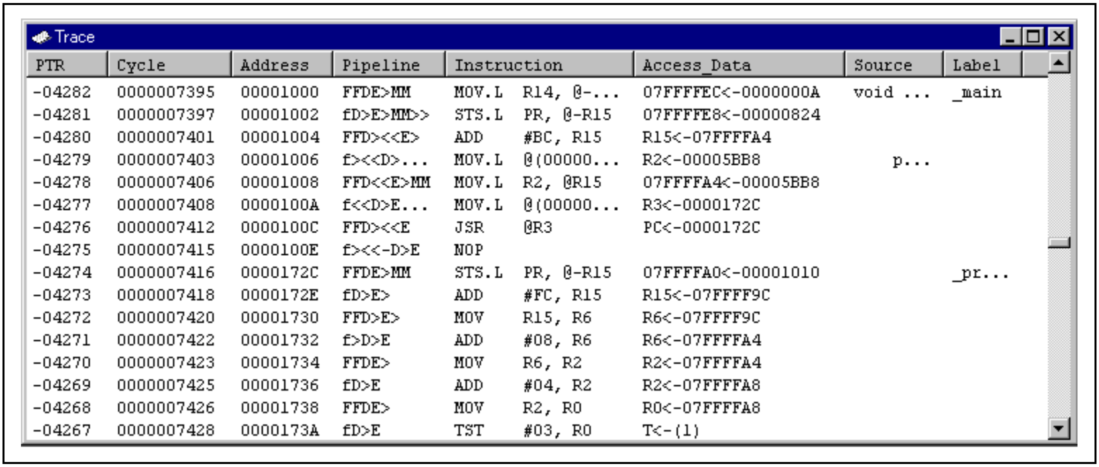
The **Status** window is split into three sheets:

1. Memory - contains information about the current memory status including the memory mapping resources and the areas used by the currently loaded object file.
2. Platform - contains information about the current status of the debugging platform, typically including CPU type and mode; and run status.
3. Events - contains information about the current event (breakpoint) status, including resource information.

5.17 Trace Window

This window displays trace information. The displayed information items depend on the target CPU. The trace acquisition conditions can be specified in the **Trace Acquisition** dialog box.

SH-1, SH-2, SH-2E, and SH2-DSP Series:



PTR	Cycle	Address	Pipeline	Instruction	Access_Data	Source	Label
-04282	0000007395	00001000	FFDE>MM	MOV.L R14, @-...	07FFFFEC<-0000000A	void ...	_main
-04281	0000007397	00001002	fD>E>MM>>	STS.L PR, @-R15	07FFFFE8<-00000824		
-04280	0000007401	00001004	FFD><<E>	ADD #BC, R15	R15<-07FFFFA4		
-04279	0000007403	00001006	f><<D>...	MOV.L @(00000...	R2<-00005BB8	p...	
-04278	0000007406	00001008	FFD<<E>MM	MOV.L R2, @R15	07FFFFA4<-00005BB8		
-04277	0000007408	0000100A	f<<D>E...	MOV.L @(00000...	R3<-0000172C		
-04276	0000007412	0000100C	FFD><<E	JSR @R3	PC<-0000172C		
-04275	0000007415	0000100E	f><<-D>E	NOP			
-04274	0000007416	0000172C	FFDE>MM	STS.L PR, @-R15	07FFFFA0<-00001010		_pr...
-04273	0000007418	0000172E	fD>E>	ADD #FC, R15	R15<-07FFFF9C		
-04272	0000007420	00001730	FFD>E>	MOV R15, R6	R6<-07FFFF9C		
-04271	0000007422	00001732	f>D>E	ADD #08, R6	R6<-07FFFFA4		
-04270	0000007423	00001734	FFDE>	MOV R6, R2	R2<-07FFFFA4		
-04269	0000007425	00001736	fD>E	ADD #04, R2	R2<-07FFFFA8		
-04268	0000007426	00001738	FFDE>	MOV R2, R0	R0<-07FFFFA8		
-04267	0000007428	0000173A	fD>E	TST #03, R0	T<-(1)		

Figure 5.26 Trace Window (for SH-1, SH-2, SH-2E, and SH2-DSP Series)

This window displays the following trace information items:

[PTR] Pointer in the trace buffer (0 for the last executed instruction)

[Cycle] Total number of instruction execution cycles (cleared by pipeline reset)

[Address] Instruction address

[Pipeline] Pipeline execution status
Each symbol has the following meaning:
F: Instruction fetch (with memory access)
f: Instruction fetch (without memory access)
D: Instruction decode
E: Instruction execution
M: Memory access
W: Write back
P: DSP
m: Multiplier execution
–: Stall inherent in the instruction
>: Split
<: Stall due to conflict

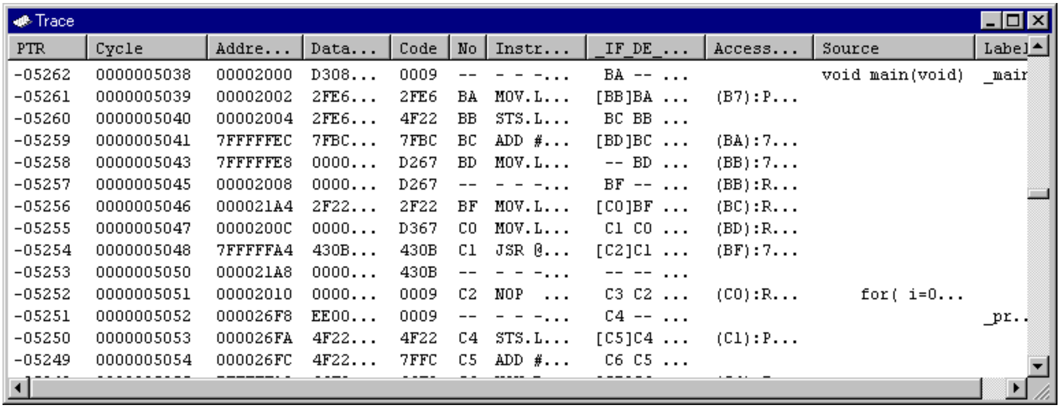
Refer to the programming manual of each device for more information on pipeline operation.

[Instruction] Instruction mnemonic

[Access Data] Data access information (display format: destination <- accessed data)

[Source] C/C++ or assembly-language source programs

SH-3 and SH-3E Series:



PTR	Cycle	Addre...	Data...	Code	No	Instr...	_IF_DE_...	Access...	Source	Label
-05262	0000005038	00002000	D308...	0009	--	- - -...	BA -- ...		void main(void)	_mair
-05261	0000005039	00002002	2FE6...	2FE6	BA	MOV.L...	[BB]BA ...	(B7):P...		
-05260	0000005040	00002004	2FE6...	4F22	BB	STS.L...	BC BB ...			
-05259	0000005041	7FFFFFEC	7FBC...	7FBC	BC	ADD #...	[BD]BC ...	(BA):7...		
-05258	0000005043	7FFFFFE8	0000...	D267	BD	MOV.L...	-- BD ...	(BB):7...		
-05257	0000005045	00002008	0000...	D267	--	- - -...	BF -- ...	(BB):R...		
-05256	0000005046	000021A4	2F22...	2F22	BF	MOV.L...	[C0]BF ...	(BC):R...		
-05255	0000005047	0000200C	0000...	D367	C0	MOV.L...	C1 C0 ...	(BD):R...		
-05254	0000005048	7FFFFFA4	430B...	430B	C1	JSR @...	[C2]C1 ...	(BF):7...		
-05253	0000005050	000021A8	0000...	430B	--	- - -...	-- -- ...			
-05252	0000005051	00002010	0000...	0009	C2	NOP ...	C3 C2 ...	(C0):R...	for(i=0...	
-05251	0000005052	000026F8	EE00...	0009	--	- - -...	C4 -- ...			_pr..
-05250	0000005053	000026FA	4F22...	4F22	C4	STS.L...	[C5]C4 ...	(C1):P...		
-05249	0000005054	000026FC	4F22...	7FFC	C5	ADD #...	C6 C5 ...			

Figure 5.27 Trace Window (for SH-3 and SH-3E Series)

This window displays the following trace information items:

[PTR] Pointer in the trace buffer (0 for the last executed instruction)

[Cycle] Total number of instruction execution cycles (cleared by pipeline reset)

[Address Bus] Data on the address bus

[Data Bus] Data on the data bus

[Code] Instruction code

[No] Instruction number (corresponds to execution number in each stage)

[Instruction] Instruction mnemonic

[IF] Instruction number that was fetched (enclosed by [ ] when the instruction did not access memory)

[DE] Instruction number that was decoded

[EX] Instruction number that was executed

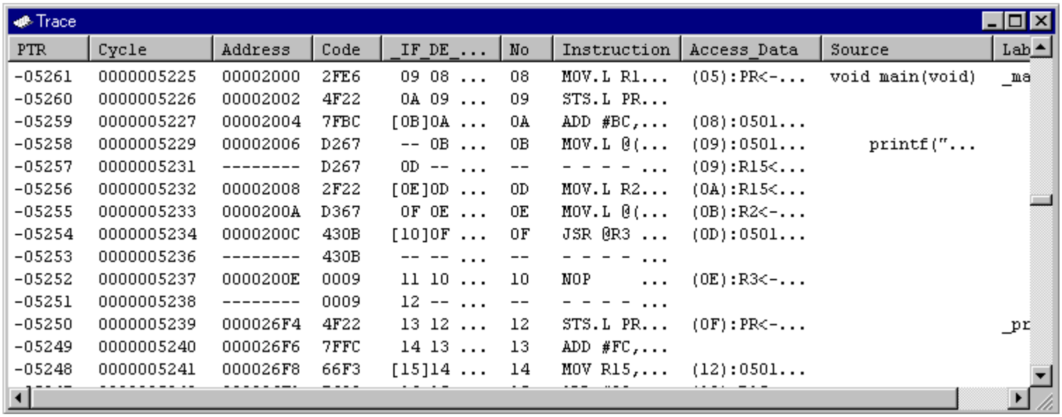
[MA] Instruction number that accessed memory

[SW] Instruction number that wrote back data

[Access Data] Data access information (display format: destination <- accessed data)

[Source] C/C++ or assembly-language source programs

SH3-DSP Series:



```
PTR     Cycle       Address   Code  _IF_DE_...  No  Instruction  Access_Data   Source             Lab
-05261  0000005225  00002000  2FE6   09 08 ...  08  MOV.L R1...  (05):PR<-...  void main(void)    _ma
-05260  0000005226  00002002  4F22   0A 09 ...  09  STS.L PR...
-05259  0000005227  00002004  7FBC  [0B]0A ...  0A  ADD #BC,...  (08):0501...
-05258  0000005229  00002006  D267   -- 0B ...  0B  MOV.L @(...  (09):0501...      printf("...
-05257  0000005231  --------  D267   0D -- ...  --  - - - - ...  (09):R15<...
-05256  0000005232  00002008  2F22  [0E]0D ...  0D  MOV.L R2...  (0A):R15<...
-05255  0000005233  0000200A  D367   0F 0E ...  0E  MOV.L @(...  (0B):R2<-...
-05254  0000005234  0000200C  430B  [10]0F ...  0F  JSR @R3 ...  (0D):0501...
-05253  0000005236  --------  430B   -- -- ...  --  - - - - ...
-05252  0000005237  0000200E  0009   11 10 ...  10  NOP     ...  (0E):R3<-...
-05251  0000005238  --------  0009   12 -- ...  --  - - - - ...
-05250  0000005239  000026F4  4F22   13 12 ...  12  STS.L PR...  (0F):PR<-...                     _pr
-05249  0000005240  000026F6  7FFC   14 13 ...  13  ADD #FC,...
-05248  0000005241  000026F8  66F3  [15]14 ...  14  MOV R15,...  (12):0501...
```

Figure 5.28 Trace Window (for SH3-DSP Series)

This window displays the following trace information items:

[PTR] Pointer in the trace buffer (0 for the last executed instruction)

[Cycle] Total number of instruction execution cycles (cleared by pipeline reset)

[Address] Program counter value

[Code] Instruction code

[IF] Instruction number that was fetched (enclosed by [ ] when the instruction did not access memory)

[DE] Instruction number that was decoded

[EX] Instruction number that was executed

[MA] Instruction number that accessed memory

[SW] Instruction number that wrote back data

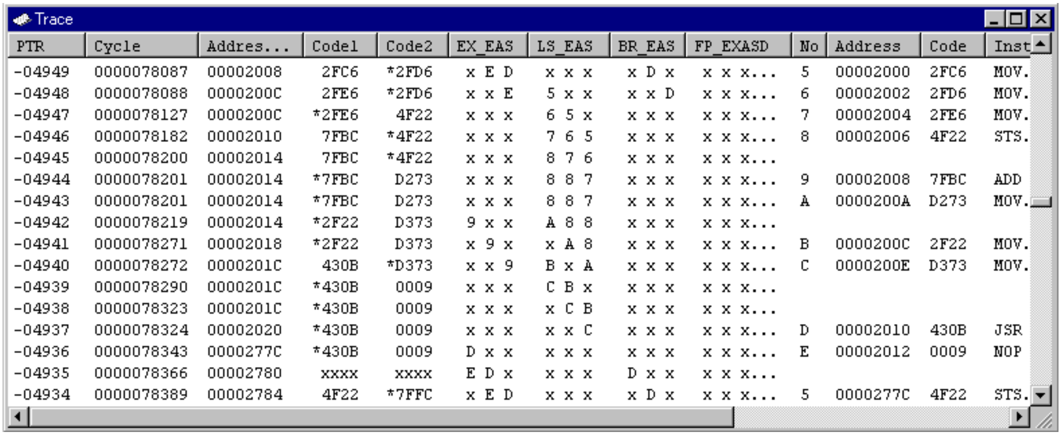
[No] Instruction number (corresponds to execution number in each stage)

[Instruction] Instruction mnemonic

[Access Data] Data access information (display format: destination <- accessed data)

[Source] C/C++ or assembly-language source programs

SH-4 Series:



PTR	Cycle	Addres...	Code1	Code2	EX_EAS	LS_EAS	BR_EAS	FP_EXASD	No	Address	Code	Inst
-04949	0000078087	00002008	2FC6	*2FD6	x E D	x x x	x D x	x x x...	5	00002000	2FC6	MOV.
-04948	0000078088	0000200C	2FE6	*2FD6	x x E	5 x x	x x D	x x x...	6	00002002	2FD6	MOV.
-04947	0000078127	0000200C	*2FE6	4F22	x x x	6 5 x	x x x	x x x...	7	00002004	2FE6	MOV.
-04946	0000078182	00002010	7FBC	*4F22	x x x	7 6 5	x x x	x x x...	8	00002006	4F22	STS.
-04945	0000078200	00002014	7FBC	*4F22	x x x	8 7 6	x x x	x x x...				
-04944	0000078201	00002014	*7FBC	D273	x x x	8 8 7	x x x	x x x...	9	00002008	7FBC	ADD
-04943	0000078201	00002014	*7FBC	D273	x x x	8 8 7	x x x	x x x...	A	0000200A	D273	MOV.
-04942	0000078219	00002014	*2F22	D373	9 x x	A 8 8	x x x	x x x...				
-04941	0000078271	00002018	*2F22	D373	x 9 x	x A 8	x x x	x x x...	B	0000200C	2F22	MOV.
-04940	0000078272	0000201C	430B	*D373	x x 9	B x A	x x x	x x x...	C	0000200E	D373	MOV.
-04939	0000078290	0000201C	*430B	0009	x x x	C B x	x x x	x x x...				
-04938	0000078323	0000201C	*430B	0009	x x x	x C B	x x x	x x x...				
-04937	0000078324	00002020	*430B	0009	x x x	x x C	x x x	x x x...	D	00002010	430B	JSR
-04936	0000078343	0000277C	*430B	0009	D x x	x x x	x x x	x x x...	E	00002012	0009	NOP
-04935	0000078366	00002780	xxxx	xxxx	E D x	x x x	D x x	x x x...				
-04934	0000078389	00002784	4F22	*7FFC	x E D	x x x	x D x	x x x...	5	0000277C	4F22	STS.

Figure 5.29 Trace Window (for SH-4 Series)

This window displays the following trace information items:

[PTR] Pointer within the trace buffer (The latest instruction is 0)

[Cycle] Total number of instruction execution cycles (cleared by pipeline reset). The CPU internal clock cycles are counted as execution cycles. The rates of the external and internal clocks can be specified using the **Clock Rate** command. For the SH-4BSC, one execution cycle is equivalent to three external cycles.

[Address Bus] Program counter value

[Code1] Fetched code 1

[Code2] Fetched code 2

The code currently decoded is marked with *. If the two codes are executed in parallel, the code fetched at the smaller address is marked with *.

[EX-EAS] Instruction number that was executed (in the E stage), accessed memory (in the A stage), or wrote back data (in the S stage) in the EX pipeline

[LS-EAS] Instruction number that was executed, accessed memory, or wrote back data in the LS pipeline

[BR-EAS] Instruction number that was executed, accessed memory, or wrote back data in the BR pipeline

[FP-EXASD] Instruction number that was executed, accessed memory, or wrote back data in the FP pipeline (only for FSCA, FSRRA, FIPR, and FTRV instructions in the X stage, and for FDIV and FSQRT instructions in the D stage)

[No] Instruction number (corresponds to execution number in each stage)

[Address] Executed instruction address

[Code] Executed instruction code

[Instruction] Executed instruction mnemonic

[Access Data] Data access information (display format: destination <- transfer data)

[Source] C/C++ or assembly-language source programs

Double-clicking a line in the **Trace** window opens the **Source** window or **Disassembly** window. In the window, the source code is displayed and the selected line is indicated by the cursor.

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

5.17.1 Find...

Launches the **Trace Search** dialog box, allowing the user to search the current trace buffer for a specific trace record.

5.17.2 Find Next

If a find operation is successful, and the item found is non-unique, then this will move to the next similar item.

5.17.3 Acquisition

Launches the **Trace Acquisition** dialog box, allowing the user to define the area of user program to be traced.

5.17.4 Clear

Empties the trace buffer in the debugging platform. If more than one trace window is open, all **Trace** windows will be cleared as they all access the same buffer.

5.17.5 Save...

Launches the **Save As** file dialog box, allowing the user to save the contents of the trace buffer as a text file. A range can be defined based on the PTE number (saving the complete buffer may take several minutes). Note that this file cannot be reloaded into the trace buffer.

5.17.6 View Source

Displays the source program corresponding to the address.

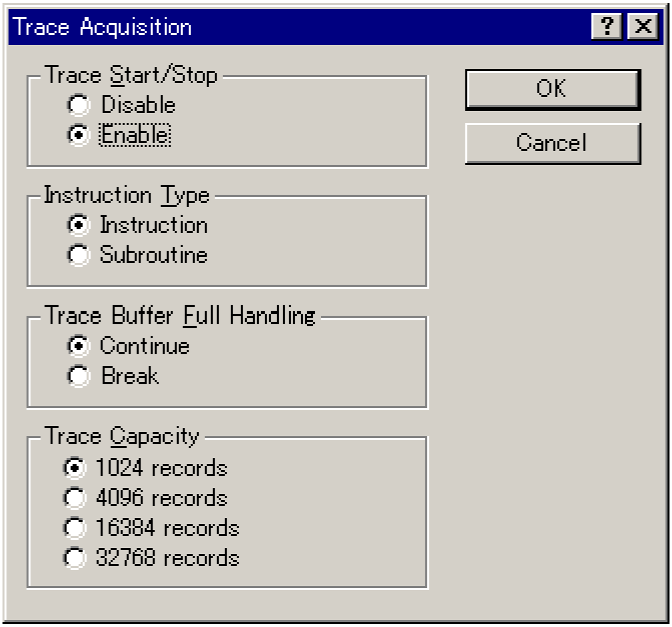
5.17.7 Trim Source

Removes white space from the left side of the source.

5.17.8 Statistic

Analyzes statistical information under the specified conditions.

5.18 Trace Acquisition Dialog Box



Trace Acquisition

Trace Start/Stop
○ Disable
● Enable

Instruction Type
● Instruction
○ Subroutine

Trace Buffer Full Handling
● Continue
○ Break

Trace Capacity
● 1024 records
○ 4096 records
○ 16384 records
○ 32768 records

Figure 5.30 Trace Acquisition Dialog Box

This dialog box specifies the conditions for trace information acquisition.

[Trace Start/Stop]

[Disable]	Disables trace information acquisition.
[Enable]	Enables trace information acquisition.

[Instruction Type]

[Instruction]	Acquires trace information for all instructions.
[Subroutine]	Acquires trace information for the subroutine instructions only.

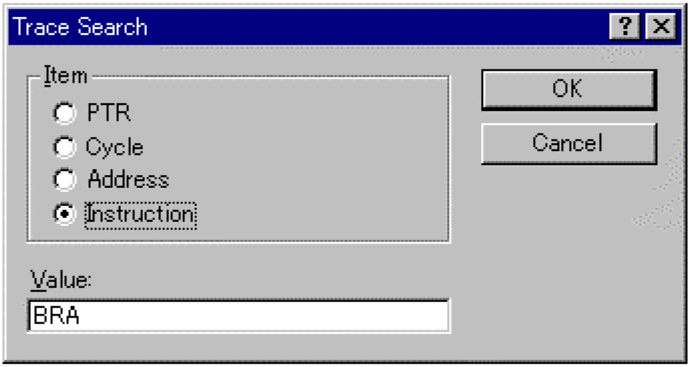
[Trace Buffer Full Handling]

[Continue]	Continues acquiring trace information even if the trace information acquisition buffer becomes full.
[Break]	Stops execution when the trace information acquisition buffer becomes full.

The trace buffer capacity can be selected from 1024 records, 4096 records, 16384 records, and 32768 records in the **[Trace Capacity]**.

Clicking the **[OK]** button stores the settings. Clicking the **[Cancel]** button closes this dialog box without modifying the settings.

5.19 Trace Search Dialog Box



○ PTR
○ Cycle
○ Address
● Instruction

Value: BRA

Figure 5.31 Trace Search Dialog Box

This dialog box specifies the conditions for searching trace information. Specify a search item in **[Item]** and search for the specified contents in **[Value]**.

[PTR] Pointer in the trace buffer (0 for the last executed instruction, specify in the form of -nnn)

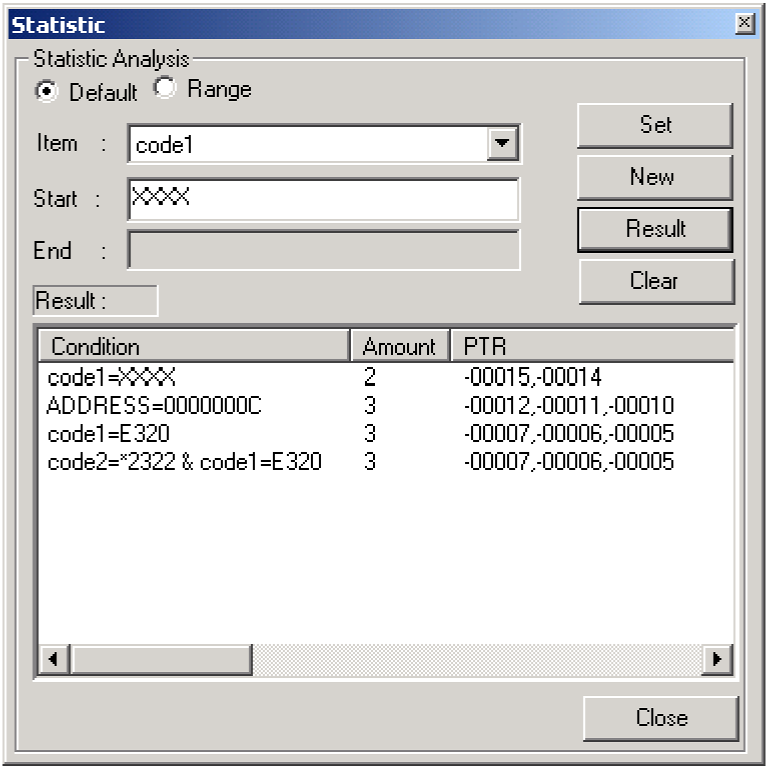
[Cycle] Total number of instruction execution cycles

[Address] Instruction address

[Instruction] Instruction mnemonic

Clicking the **[OK]** button stores the settings. Clicking the **[Cancel]** button closes this dialog box without searching.

5.20 Trace Statistic Dialog Box



Statistic Analysis

● Default ○ Range

Item : code1

Start : XXXX

End :

Result :

Condition	Amount	PTR
code1=XXXX	2	-00015,-00014
ADDRESS=0000000C	3	-00012,-00011,-00010
code1=E320	3	-00007,-00006,-00005
code2=*2322 & code1=E320	3	-00007,-00006,-00005

Figure 5.32 Trace Statistic Dialog Box

Allows the user to analyze statistical information concerning the trace data. The target of analysis is specified in **[Item]** and the input value or character string is specified by **[Start]** and **[End]**.

When **[Default]** is selected, the input value or character string cannot be specified as a range. To specify a range, select **[Range]**.

[Set]	Adds a new condition to the current one
[New]	Creates a new condition
[Result]	Obtains the result of statistical information analysis
[Clear]	Clears all condition and results of statistical information analysis

Clicking the **[Close]** button closes this dialog box.

5.21 Trigger Window

Displays trigger buttons to manually generate interrupts. The details of the interrupt to be generated by pressing each trigger button can be specified in the **Trigger Setting** dialog box.

Up to 16 trigger buttons can be used.

For details on the interrupt processing in the simulator/debugger, refer to section 3.17, Pseudo-Interrupts.

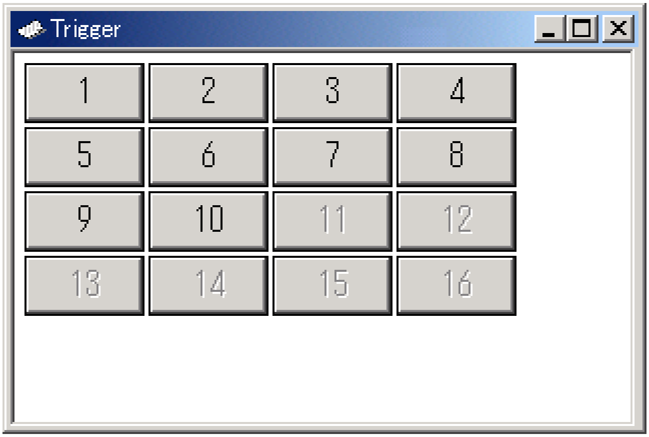


Figure 5.33 Trigger Window

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

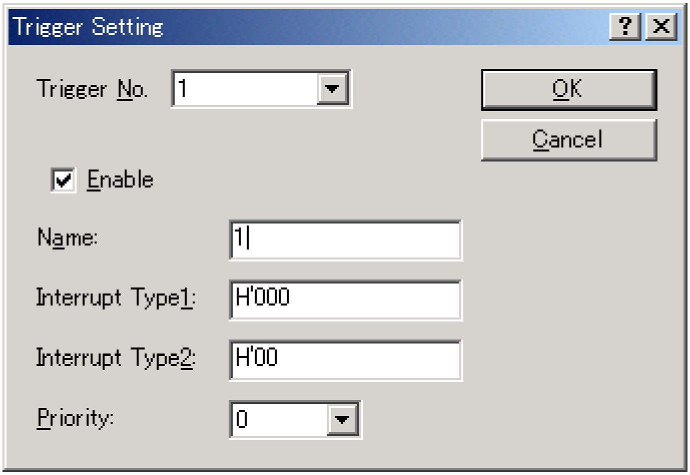
5.21.1 Setting...

Launches the **Trigger Setting** dialog box, allowing the user to specify the details of the interrupt to be generated by pressing each trigger button.

5.21.2 Size

Specifies the size of trigger buttons displayed in the **Trigger** window. Large, Normal, or Small can be selected from the submenu.

5.22 Trigger Setting Dialog Box


Trigger Setting

Trigger No. 1

☑ Enable

Name: 1

Interrupt Type1: H'000

Interrupt Type2: H'00

Priority: 0

Figure 5.34 Trigger Setting Dialog Box

Allows the user to specify the details of the interrupt to be generated by pressing each trigger button.

[Trigger No.] Selects a trigger button to specify details

[Name] Specifies the name of the selected trigger button, which will be displayed in the **Trigger** window

[Enable] Enables the trigger button when this box is checked.

[Interrupt Type1] Specifies the following values for each CPU
SH-1, SH-2, and SH2-DSP series:
Interrupt vector number (H’0 to H’FF)
SH-3, SH-4, and SH3-DSP series:
INTEVT (H’0 to H’FFF)

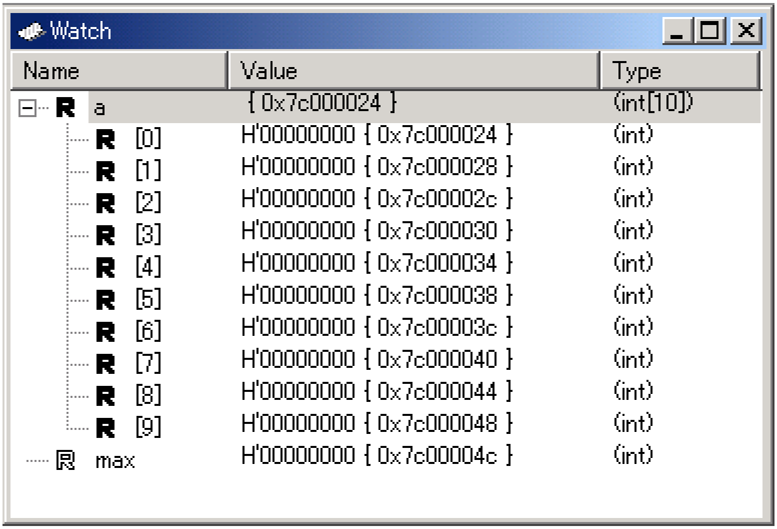
[Interrupt Type2] Only the SH3-DSP series can specify the following:
INTEVT2 (H’0 to H’FFF)

[Priority] Interrupt priority (when the prefix is omitted, values must be input and are displayed in hexadecimal) (H’0 to H’11)
When H'10 is specified, the interrupt is always accepted regardless of the I bit in SR, but is masked by the BL bit in SR.
When H'11 is specified, the interrupt is always accepted regardless of the I and BL bits in SR.

Clicking the **[OK]** button stores the setting. Clicking the **[Cancel]** button closes this dialog box without setting the details of the interrupt.

Note: If the [Cancel] button is clicked after multiple trigger button settings are modified, the modifications of all those buttons are canceled.

5.23 Watch Window



Name	Value	Type
⊟ R a	{ 0x7c000024 }	(int[10])
R [0]	H'00000000 { 0x7c000024 }	(int)
R [1]	H'00000000 { 0x7c000028 }	(int)
R [2]	H'00000000 { 0x7c00002c }	(int)
R [3]	H'00000000 { 0x7c000030 }	(int)
R [4]	H'00000000 { 0x7c000034 }	(int)
R [5]	H'00000000 { 0x7c000038 }	(int)
R [6]	H'00000000 { 0x7c00003c }	(int)
R [7]	H'00000000 { 0x7c000040 }	(int)
R [8]	H'00000000 { 0x7c000044 }	(int)
R [9]	H'00000000 { 0x7c000048 }	(int)
R max	H'00000000 { 0x7c00004c }	(int)

Figure 5.35 Watch Window

Allows the user to view and modify C/C++-source level variables. The contents of this window are displayed only when the debugging information available in the absolute file (*.abs) includes the information on the C/C++ source program. The variable information is not displayed if the source program information is excluded from the debugging information during optimization by the compiler. In addition, the variables that are declared as macro cannot be displayed.

The following items are displayed.

[Name] Name of the variable

[Value] Value, assigned location, and type.
The assigned location is enclosed by { }.

[Type] Displays the type of variable

"+" shown to the left of the variable name indicates that the information may be expanded by double-clicking on the variable name, and "-" indicates that the information may be collapsed. Alternatively, the "+" and "-" keys may be used.

“R” specifies whether the corresponding variable is updated in real time. When a “R” is bold, the value of the corresponding variable will be updated in real time during user program execution.

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

5.23.1 Auto Update

“R” mark of the selected variable becomes bold-faced type and updates the variable in real time.

5.23.2 Auto Update All

All “R” marks become bold-faced type and update all the displayed variables in real time.

5.23.3 Delete Auto Update

“R” mark of the selected variable becomes not bold-typed face and cancels real time update.

5.23.4 Delete Auto Update All

All “R” marks become not bold-faced type and cancel real time update.

5.23.5 Add Watch...

Launches the **Add Watch** dialog box, allowing the user to enter a variable to be watched.

5.23.6 Edit Value...

Launches the **Edit Value** dialog box, allowing the user to change the variable's value. Particular care should be taken when the value of a pointer is changed as it may no longer point to valid data.

5.23.7 Delete

Removes the selected variable from the **Watch** window.

5.23.8 Delete All

Removes all the variables from the **Watch** window.

5.23.9 Radix

Modifies the radix for the selected variable display.

5.23.10 Copy

Only available if a block of text is highlighted. This copies the highlighted text into the Windows® clipboard, allowing it to be pasted into other applications.

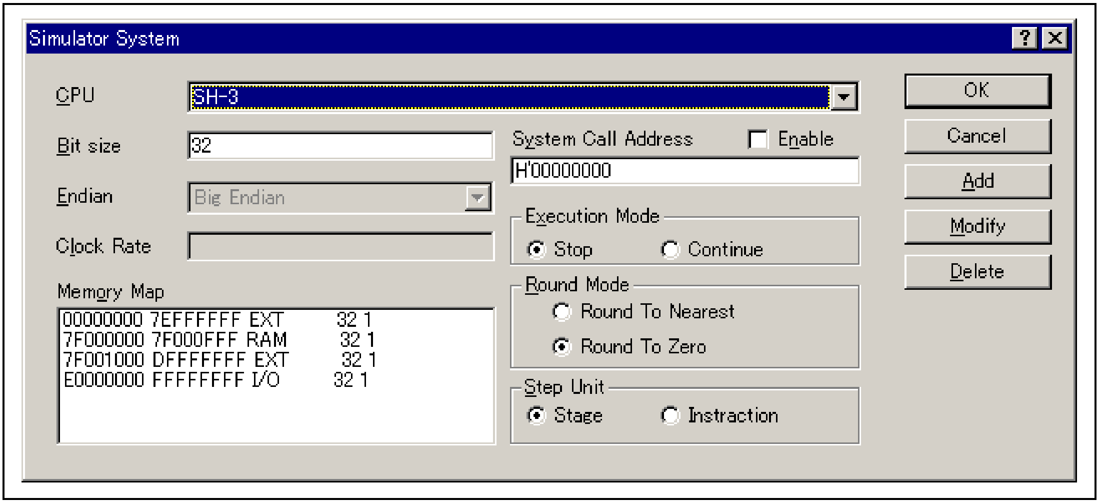
5.23.11 Save As...

Launches the **Save As** dialog box, allowing the user to specify the name of a file and saves the contents of the **Watch** window in the file. If the **Append** check box is selected, the window contents are appended to the existing file, and if it is not selected, the existing file is overwritten.

5.23.12 Go To Memory...

Displays the memory contents to which the selected variable is assigned in the **Memory** window.

5.24 Simulator System Dialog Box



Simulator System

CPU: SH-3

Bit size: 32

Endian: Big Endian

Clock Rate:

Memory Map

```
00000000 7EFFFFFF EXT    32 1
7F000000 7F000FFF RAM    32 1
7F001000 DFFFFFFF EXT    32 1
E0000000 FFFFFFFF I/O    32 1
```

System Call Address: H'00000000 □ Enable

Execution Mode: ● Stop ○ Continue

Round Mode: ○ Round To Nearest ● Round To Zero

Step Unit: ● Stage ○ Instraction

Figure 5.36 Simulator System Dialog Box

This dialog box specifies the endian, system call start location, execution mode, floating-point rounding mode, and memory map.

[CPU] Displays the current CPU. (The CPU must be specified in the **Debug Settings** dialog box.)
For the SH2-DSP (SH7410), **Internal Mode** or **External Mode** is selected.
For SH2-DSP (SH7065), one of the following is selected:
[ROM Disable/Fast Mode] Specifies internal ROM disabled/fast access mode.
[ROM Disable/Slow Mode] Specifies internal ROM disabled/slow access mode.

[ROM Enable/Fast Mode] Specifies internal ROM enabled/fast access mode.
[ROM Enable/Slow Mode] Specifies internal ROM enabled/slow access mode.

For SH2-DSP (Core), one of the following is selected:

[ROM Disable] Specifies internal ROM disabled mode.
[ROM Enable] Specifies internal ROM enabled mode.

[Bit size] Displays the address bus width. It is fixed to 32 bits.

[Endian] Displays the currently specified endian.

[Clock Rate] This can be specified only for SH-4 series.
Specifies the internal clock ratio when the external clock is assumed 1. (H1 to H'10)

[System Call Address] Specifies the start address of a system call that performs standard input/output or file input/output processing from the user system.
[Enable] Specifies whether the system call is enabled or disabled.

[Execution Mode] Specifies whether the simulator/debugger stops or continues operating when a simulation error occurs.
[Stop] Stops the simulation.
[Continue] Continues the simulation.

[Round Mode] Specifies the rounding mode for floating-point decimal-to-binary conversion.
[Round to nearest] Rounds to the nearest value.
[Round to zero] Rounds toward zero.

[Step Unit] This can be specified only for SH-3, SH3-DSP series, and SH-4 series.
Specifies the status of the instruction executed when STEP command is executed and a PC break is satisfied.
[Stage] Stops before instruction is executed.
Pipeline operation is not affected.
[Instruction] Stops after instruction is executed.
Pipeline operation is affected, and as a result, the number of effective cycles become incorrect.

In the **[Memory Map]**, the start address, end address, memory type, data bus width, and access cycles are displayed in that order. The memory types are as follows:

- SH-1, SH-2, SH-2E, SH-3, and SH-3E
  ROM (internal ROM), RAM (internal RAM), EXT (external memory), I/O (internal I/O)
- SH2-DSP (SH7410)/SH3-DSP
  XROM (internal XROM), YROM (internal YROM), XRAM (internal XRAM), YRAM (internal YRAM), EXT (external memory), I/O (internal I/O)
- SH2-DSP (SH7612)

XRAM (internal XRAM), YRAM (internal YRAM), INTRAM (internal RAM), EXT (external memory), I/O (internal I/O)

- SH2-DSP (SH7065)

  XRAM (internal XRAM), YRAM (internal YRAM), INTROM (internal ROM), EXT (external memory), I/O (internal I/O)

- SH2-DSP (Core)

  XRAM (internal XRAM), YRAM (internal YRAM), INTROM (internal ROM), INTRAM (internal RAM), EXT (external memory), I/O (internal I/O)

- SH3-DSP (Core)

  XRAM (internal XRAM), YRAM (internal YRAM), URAM (user RAM), EXT (external memory), I/O (internal I/O)

- SH-4/SH-4 (SH7750R)

  NORMAL (normal memory), INTRAM (internal RAM), I/O (internal I/O)

- SH-4BSC

  NORMAL (normal memory), MPX (Multiplex), BCSRAM (byte-control SRAM), BSTROM (burst ROM and burst count), DRAM (DRAM), SDRAM (synchronous DRAM), INTRAM (internal RAM), I/O (internal I/O)

[Memory Map] can be specified, modified, or deleted using the following buttons:

[Add] Specifies **[Memory Map]** items. Clicking this button opens the **Memory Map Modify** dialog box, and memory map items can be specified.

[Modify] Modifies **[Memory Map]** items. Select an item to be modified in the list box and click the **[Modify]** button. The **Memory Map Modify** dialog box opens and memory map items can be modified.

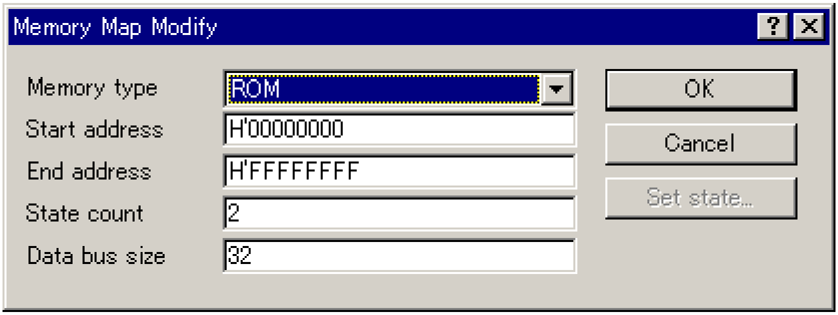
[Delete] Deletes **[Memory Map]** items. Select an item to be deleted in the list box and click the **Delete** button.

Notes: For the SH-4BSC, the following must be noted:

1. **The access cycle count is displayed as --.**
2. **Memory map entries cannot be newly created or canceled. Therefore, only the [Modify] button can be used for [Memory Map].**
3. **For the internal RAM and I/O, the memory map entries cannot be modified. To enable or disable the internal RAM, use the ORA bit of the CCR control register.**
4. **The memory map contents depend on the BSC settings. If the memory map contents cannot be determined due to BSC incorrect settings, this dialog box shows ?????? for memory types and ?? for the bus width.**

Clicking the **[OK]** button stores the modified settings. Clicking the **[Cancel]** button closes this dialog box without modifying the settings.

5.25 Memory Map Modify Dialog Box



Memory type: ROM
Start address: H'00000000
End address: H'FFFFFFFF
State count: 2
Data bus size: 32

Figure 5.37 Memory Map Modify Dialog Box

This dialog box specifies the memory map of the target CPU of the simulator/debugger.

The contents displayed in this dialog box depend on the target CPU. The specified data is used to calculate the number of cycles for memory access.

[Memory type] Memory type

[Start address] Start address of the memory corresponding to a memory type

[End address] End address of the memory corresponding to a memory type

[State count] Number of memory access cycles

[Data bus size] Memory data bus width

For the SH-4 series, **[Start address]** and **[End address]** cannot be modified. Specify **[Memory type]** and **[Data bus size]**.

In addition, for the SH-4 series, clicking the **[Set state...]** button opens the **Set State** dialog box, and the number of wait states to be inserted in areas 0 to 7 can be specified. The specified values correspond to the values in the WCR1 and WCR2 control registers.

Clicking the **[OK]** button stores the settings. Clicking the **[Cancel]** button closes this dialog box without modifying the settings.

Note: The memory map setting for the area allocated to a system memory resource cannot be deleted or modified. First delete the system memory resource allocation with the Simulator Memory Resource dialog box, then delete or modify the memory map setting.

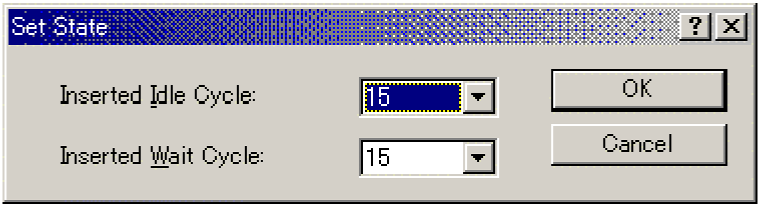
5.26 Set State Dialog Box

This dialog box specifies the wait state to be inserted in each area. This dialog box is provided only for the SH-4 series.

Note that only the normal memory is supported for the SH-4/SH-4 (SH7750R).

The **Set State** dialog box contents depend on the memory type allocated to the target area. The values set in this dialog box are also set to the WCR1 and WCR2 control registers.

Normal Memory, Burst ROM, and Byte-Control SRAM:



Inserted Idle Cycle: 15

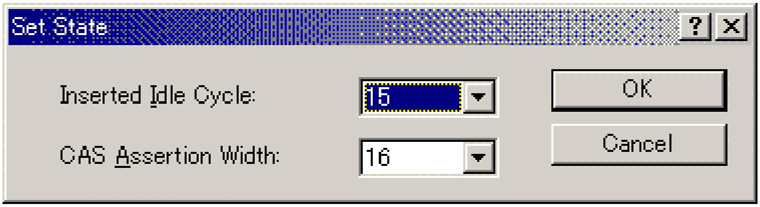
Inserted Wait Cycle: 15

Figure 5.38 Set State Dialog Box (Normal Memory)

[Inserted Idle Cycle] Specifies the number of idle cycles to be inserted when the access type changes from read to write, or when the access area changes (corresponds to the AnIW in WCR1).

[Inserted Wait Cycle] Specifies the number of wait cycles to be inserted in all accesses (corresponds to the AnW in WCR2).

DRAM:



Inserted Idle Cycle: 15

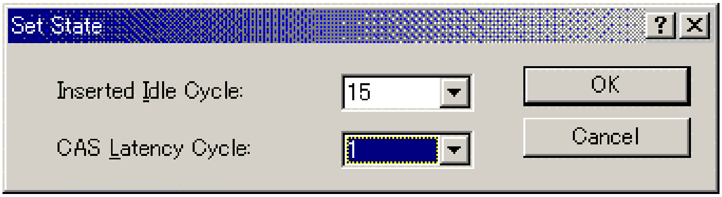
CAS Assertion Width: 16

Figure 5.39 Set State Dialog Box (DRAM)

[Inserted Idle Cycle] Specifies the number of idle cycles to be inserted when the access type changes from read to write, or when the access area changes (corresponds to the AnIW in WCR1).

[CAS Assertion Width] Specifies the $\overline{\text{CAS}}$ assertion width (corresponds to the AnW in WCR2).

SDRAM:



Inserted Idle Cycle: 15

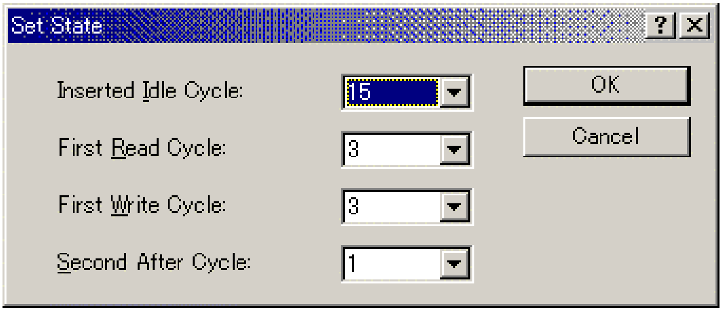
CAS Latency Cycle: 1

Figure 5.40 Set State Dialog Box (SDRAM)

[Inserted Idle Cycle] Specifies the number of idle cycles to be inserted when the access type changes from read to write, or when the access area changes (corresponds to the AnIW in WCR1).

[CAS Latency Cycle] Specifies the number of $\overline{\text{CAS}}$ latency cycles (corresponds to AnW in WCR2).

MPX:



Inserted Idle Cycle: 15

First Read Cycle: 3

First Write Cycle: 3

Second After Cycle: 1

Figure 5.41 Set State Dialog Box (MPX)

[Inserted Idle Cycle] Specifies the number of idle cycles to be inserted when the access type changes from read to write, or when the access area changes (corresponds to the AnIW in WCR1).

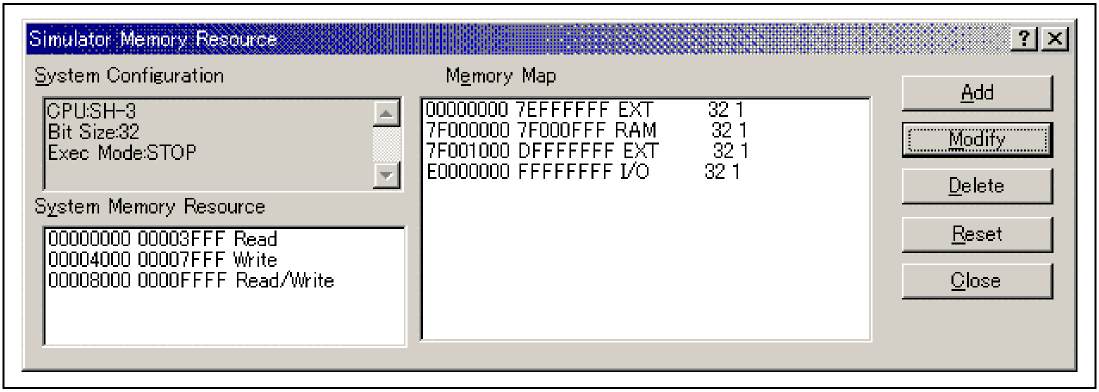
[First Read Cycle]* Specifies the number of cycles to be inserted in the first cycle for the read access.

[First Write Cycle]* Specifies the number of cycles to be inserted in the first cycle for the write access.

[Second After Cycle]* Specifies the number of cycles to be inserted in the second and the following cycles for the burst transfer.

The items marked with * correspond to the AnW in WCR2.

5.27 Simulator Memory Resource Dialog Box



```
CPU:SH-3
Bit Size:32
Exec Mode:STOP
```

```
00000000 00003FFF Read
00004000 00007FFF Write
00008000 0000FFFF Read/Write
```

```
00000000 7EFFFFFF EXT      32 1
7F000000 7F000FFF RAM      32 1
7F001000 DFFFFFFF EXT      32 1
E0000000 FFFFFFFF I/O      32 1
```

Figure 5.42 Simulator Memory Resource Dialog Box

This dialog box sets and modifies a memory map and information on the target CPU.

[System Configuration] Displays the target CPU, address bus width, and execution mode of the simulator/debugger.

[System memory resource] Displays the access type, start address, and end address of the current memory resources.

[Memory map] Displays the start address, end address, memory type, data bus width, and access cycles.

[System memory resource] can be specified, modified, and deleted using the following buttons:

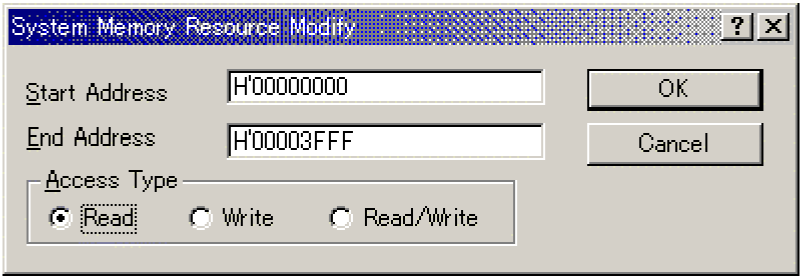
[Add] Specifies **[System memory resource]** items. Clicking this button opens the **System Memory Resource Modify** dialog box, and **[System memory resource]** items can be specified.

[Modify] Modifies **[System memory resource]** items. Select an item to be modified in the list box and click this button. The **System Memory Resource Modify** dialog box opens and **[System memory resource]** items can be modified.

[Delete] Deletes **[System memory resource]** items. Select an item to be deleted in the list box and click this button.

Note that the **[Reset]** button can reset the **[Memory map]** and **[System memory resource]** to the default value. Clicking the **[Close]** button closes this dialog box. **[System memory resource]** contains the same setting information as **[Memory resource]** on the **Simulator** sheet in the **Hitachi SuperH RISC engine Standard Toolchain** dialog box. For details on the **Hitachi SuperH RISC engine Standard Toolchain** dialog box, refer to section 2.3.1, Memory Mapping in the HEW Debugger User’s Manual.

5.28 System Memory Resource Modify Dialog Box



Start Address: H'00000000

End Address: H'00003FFF

● Read ○ Write ○ Read/Write

Figure 5.43 System Memory Resource Modify Dialog Box

This dialog box specifies or modifies system memory settings.

[Start address] Start address of the memory area to be allocated

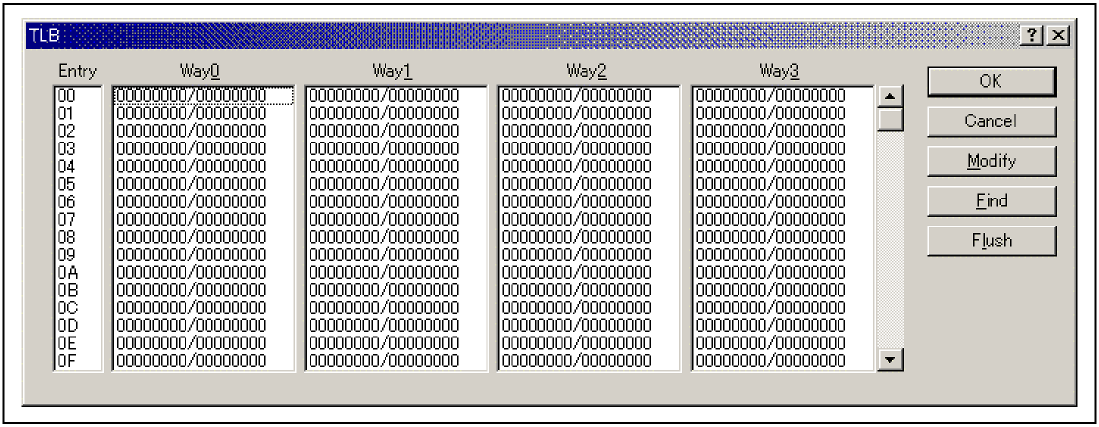
[End address] End address of the memory area to be allocated

[Access type] Access type

Read:	Read only
Write:	Write only
Read/Write:	Read and write

Click the **[OK]** button after specifying the **[Start address]**, **[End address]**, and **[Access type]**. Clicking the **[Cancel]** button closes this dialog box without modifying the setting.

5.29 TLB Dialog Box



Entry	Way0	Way1	Way2	Way3
00	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
01	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
02	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
03	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
04	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
05	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
06	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
07	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
08	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
09	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
0A	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
0B	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
0C	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
0D	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
0E	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000
0F	00000000/00000000	00000000/00000000	00000000/00000000	00000000/00000000

Figure 5.44 TLB Dialog Box

This dialog box displays the TLB contents. This dialog box is provided only for the SH-3, SH-3E, and SH3-DSP series.

The following items are displayed.

[Entry] Entry number in the TLB (H’00 to H’1F)

[Way0]-[Way3] Address array and data array in each way

The TLB contents can be modified, searched, and flushed using the following buttons.

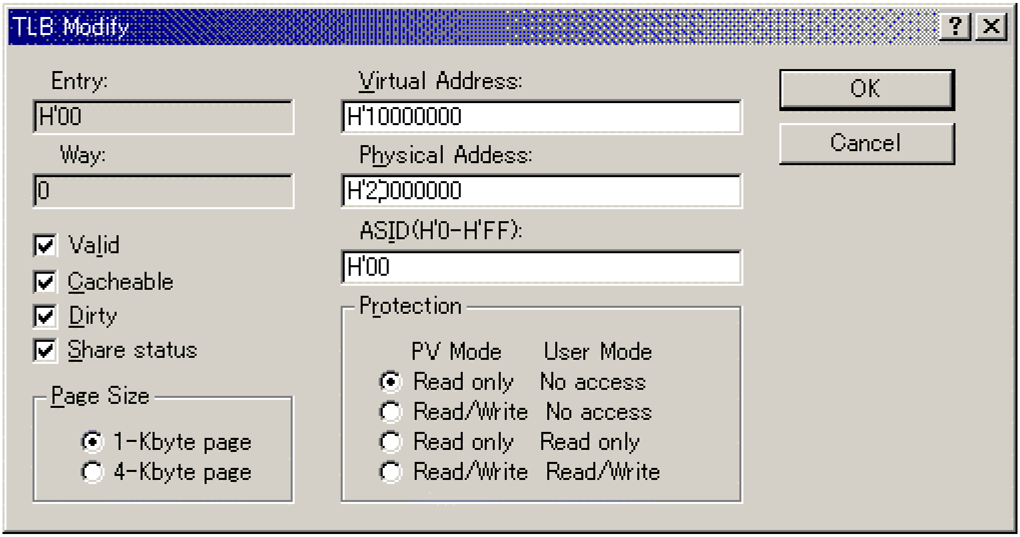
[Modify] Modifies the TLB contents. After selecting the entry to be modified in the list box, click the button. The **TLB Modify** dialog box will open and the TLB contents can be modified.

[Find] Searches the TLB contents. Clicking the button will open the **TLB Find** dialog box and the search condition can be specified.

[Flush] Flushes all TLB contents. Clicking the button clears the valid bits of all address arrays and data arrays, and invalidates all TLB entries.

Clicking the **[OK]** button stores the modified contents in the memory. Clicking the **[Cancel]** button closes the dialog box without storing the modified contents.

5.30 TLB Modify Dialog Box



Entry: H'00

Way: 0

☑ Valid
☑ Cacheable
☑ Dirty
☑ Share status

Page Size
● 1-Kbyte page
○ 4-Kbyte page

Virtual Address: H'10000000

Physical Address: H'20000000

ASID(H'0-H'FF): H'00

Protection

	PV Mode	User Mode
●	Read only	No access
○	Read/Write	No access
○	Read only	Read only
○	Read/Write	Read/Write

Figure 5.45 TLB Modify Dialog Box

This dialog box specifies the TLB contents of the entry and way selected in the **TLB** dialog box. This dialog box is provided only for the SH-3, SH-3E, and SH3-DSP series.

The following items can be specified.

[Entry] Entry number selected by the **TLB** dialog box.

[Way] Way number selected by the **TLB** dialog box.

[Virtual Address] Virtual address in longword size. Bits 16 to 12 are set to 0 regardless of the input value. Bits 31 to 10 are used as the virtual address.

[Physical Address] Physical address in longword size. Bits 31 to 10 are valid.

[ASID] Address space ID, which specifies the process that can access the virtual page.

[Valid] Specifies whether or not the entry is valid. Selecting this box makes the entry valid.

[Cacheable] Enables or disables caching. Selecting this box enables caching of the page.

[Dirty] Specifies whether or not the page has been written to. Selecting this box assumes that the page has been written to.

[Share status] Specifies whether or not to share the page with multiple processes. Selecting this box specifies that the page is shared.

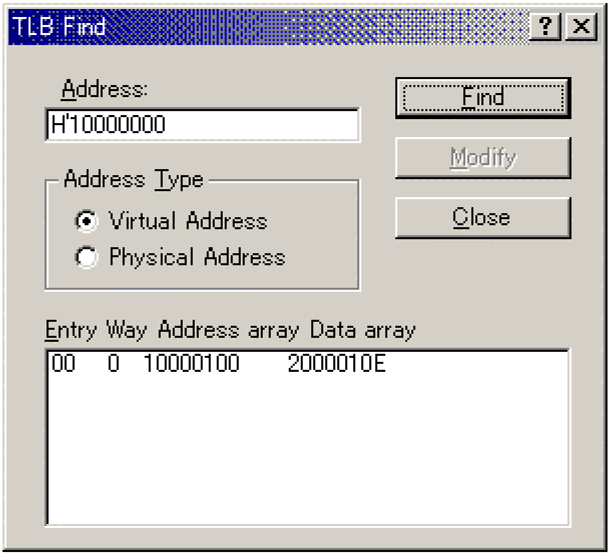
[Page Size] Specifies the page size.

The page protection status can be selected by **[Protection]**.

PV Mode	User Mode	
Read only	No access	Enables read in privileged mode.
Read/Write	No access	Enables read and write in privileged mode.
Read only	Read only	Enables read in privileged or user mode.
Read/Write	Read/Write	Enables read and write in privileged or user mode.

Clicking the **[OK]** button displays the modified contents in the **TLB** dialog box. Clicking the **[Cancel]** button closes the dialog box without modifying the TLB contents.

5.31 TLB Find Dialog Box



Address: H'10000000

Address Type
● Virtual Address
○ Physical Address

```
Entry Way Address array Data array
00    0   10000100      2000010E
```

Figure 5.46 TLB Find Dialog Box

This dialog box searches the TLB contents. This dialog box is provided only for the SH-3, SH-3E, and SH3-DSP series.

The following search conditions can be specified.

[Address] Specifies the address to be searched for.

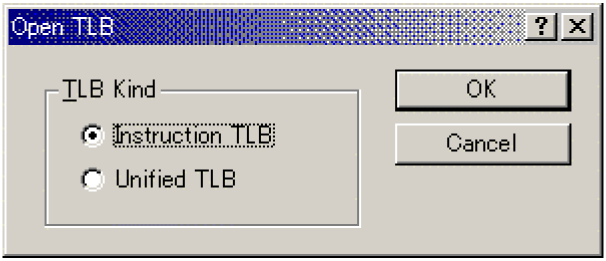
[Address Type] Specifies whether the address to be searched for is virtual or physical.

After specifying the search condition, clicking the **[Find]** button starts search. The search results are displayed in the list box at the bottom of the dialog box, in the order of TLB entry, way, address array, and data array.

To modify the displayed TLB contents, select the TLB entry in the list box, and click the **[Modify]** button. The **TLB Modify** dialog box will open and the TLB contents can be modified.

This dialog box is closed by clicking the **[Close]** button.

5.32 Open TLB Dialog Box



TLB Kind

● Instruction TLB

○ Unified TLB

Figure 5.47 Open TLB Dialog Box

This dialog box selects the TLB to be displayed. This dialog box is provided only for the SH-4 series.

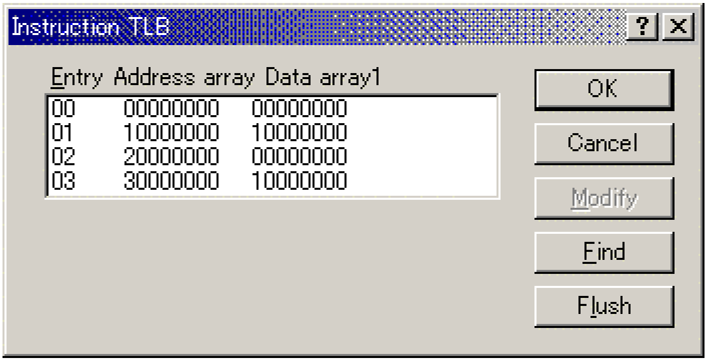
In this dialog box, select one of the following TLBs:

[Instruction TLB] Selects the instruction TLB (ITLB).

[Unified TLB] Selects the unified TLB (UTLB).

Clicking the **[OK]** button displays the selected **TLB** dialog box. Clicking the **[Cancel]** button closes the **Open TLB** dialog box.

5.33 Instruction TLB Dialog Box



Entry	Address array	Data array1
00	00000000	00000000
01	10000000	10000000
02	20000000	00000000
03	30000000	10000000

Figure 5.48 Instruction TLB Dialog Box

This dialog box displays the ITLB contents. This dialog box is provided only for the SH-4 series.

The following items are displayed.

[Entry] Entry number in the ITLB (H'00 to H'03)

[Address array] Address array in each entry of the ITLB

[Data array1] Data array 1 in each entry of the ITLB

The ITLB contents can be modified, flushed, and searched using the following buttons.

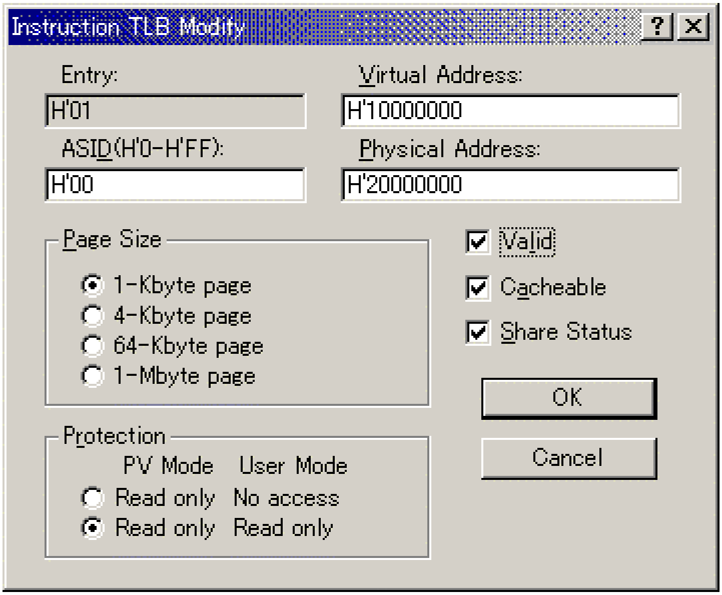
[Modify] Modifies the ITLB contents. After selecting the entry to be modified in the list box, click the button. The **Instruction TLB Modify** dialog box will open and the ITLB contents can be modified.

[Flush] Flushes all ITLB contents. Clicking the button clears the V bits of all address arrays and data arrays 1 to zero, and invalidates all ITLB entries.

[Find] Searches the ITLB contents. Clicking the button will open the **Instruction TLB Find** dialog box and the search condition can be specified.

Clicking the **[OK]** button stores the modified contents in the memory. Clicking the **[Cancel]** button closes the dialog box without storing the modified contents.

5.34 Instruction TLB Modify Dialog Box



Instruction TLB Modify

Entry: H'01

Virtual Address: H'10000000

ASID(H'0-H'FF): H'00

Physical Address: H'20000000

Page Size

● 1-Kbyte page
○ 4-Kbyte page
○ 64-Kbyte page
○ 1-Mbyte page

Protection

	PV Mode	User Mode
○	Read only	No access
●	Read only	Read only

☑ Valid
☑ Cacheable
☑ Share Status

Figure 5.49 Instruction TLB Modify Dialog Box

This dialog box modifies the ITLB contents of the entry selected in the **Instruction TLB** dialog box. This dialog box is provided only for the SH-4 series.

The following items can be specified.

[Entry] Entry number selected by the **Instruction TLB** dialog box.

[Virtual Address] Virtual address in longword size. Bits 31 to 10 are valid.

[Physical Address] Physical address in longword size. Bits 31 to 10 are valid.

[Protection] Specifies the page protection status.

PV Mode	User Mode	
Read only	No access	Enables read in privileged mode.
Read only	Read only	Enables read in privileged or user mode.

[ASID] Address space ID, which specifies the process that can access the virtual page.

[Valid] Specifies whether or not the entry is valid. Selecting this box makes the entry valid.

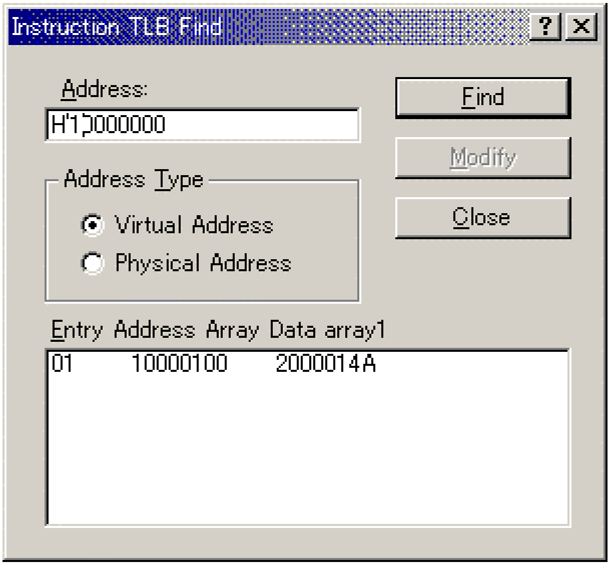
[Cacheable] Enables or disables caching. Selecting this box enables caching of the page.

[Share status] Specifies whether or not to share the page with multiple processes. Selecting this box specifies that the page is shared.

[Page Size] Specifies the page size.

Clicking the **[OK]** button displays the modified contents in the **Instruction TLB** dialog box. Clicking the **[Cancel]** button closes the dialog box without modifying the ITLB contents.

5.35 Instruction TLB Find Dialog Box



Address: H'10000000

Address Type
● Virtual Address
○ Physical Address

```
Entry Address Array Data array1
01        10000100      2000014A
```

Figure 5.50 Instruction TLB Find Dialog Box

This dialog box searches the ITLB contents. This dialog box is provided only for the SH-4 series.

The following search conditions can be specified.

[Address] Specifies the address to be searched for.

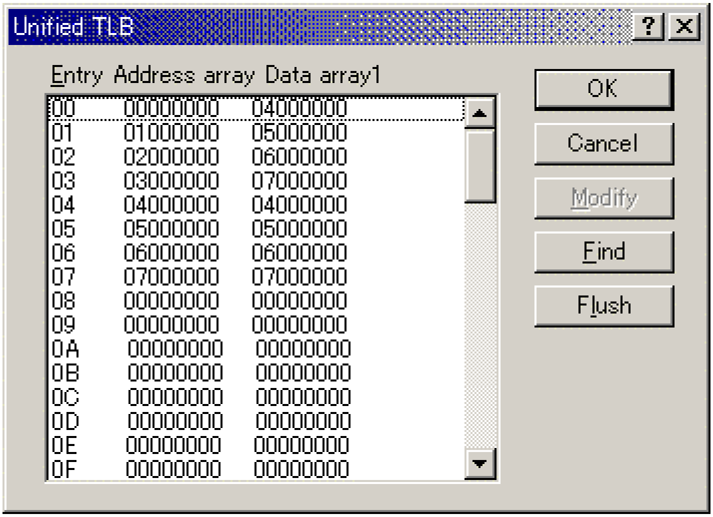
[Address type] Specifies whether the address to be searched for is virtual or physical.

After specifying the search condition, clicking the **[Find]** button starts search. The search results are displayed in the list box at the bottom of the dialog box, in the order of ITLB Entry, Address array, and Data array 1.

To modify the displayed ITLB contents, select the ITLB entry in the list box, and click the **[Modify]** button. The **Instruction TLB Modify** dialog box will open and the ITLB contents can be modified.

Clicking the **[Close]** button closes this dialog box.

5.36 Unified TLB Dialog Box



Entry	Address array	Data array1
00	00000000	04000000
01	01000000	05000000
02	02000000	06000000
03	03000000	07000000
04	04000000	04000000
05	05000000	05000000
06	06000000	06000000
07	07000000	07000000
08	00000000	00000000
09	00000000	00000000
0A	00000000	00000000
0B	00000000	00000000
0C	00000000	00000000
0D	00000000	00000000
0E	00000000	00000000
0F	00000000	00000000

Figure 5.51 Unified TLB Dialog Box

This dialog box displays the UTLB contents. This dialog box is provided only for the SH-4 series.

The following items are displayed.

[Entry] Entry number in the UTLB (H'00 to H'3F)

[Address array] Address array in each entry of the UTLB

[Data array1] Data array 1 in each entry of the UTLB

The UTLB contents can be modified, flushed, and searched using the following buttons.

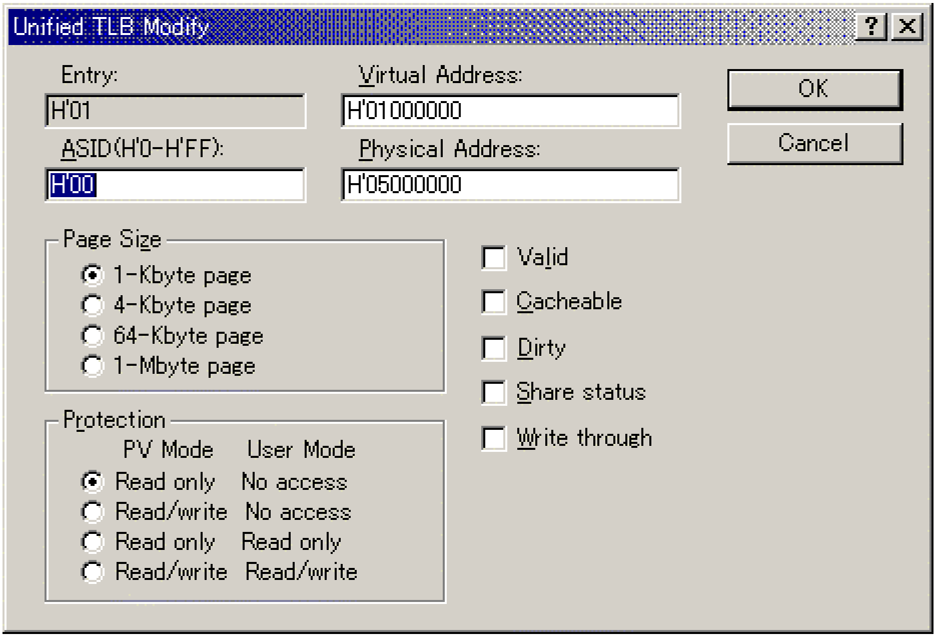
[Modify] Modifies the UTLB contents. After selecting the entry to be modified in the list box, click the button. The **Unified TLB Modify** dialog box will open and the UTLB contents can be modified.

[Flush] Flushes all UTLB contents. Clicking the button clears the V bits of all address arrays and data arrays 1 to zero, and invalidates all UTLB entries.

[Find] Searches the UTLB contents. Clicking the button will open the **Unified TLB Find** dialog box and the search condition can be specified.

Clicking the **[OK]** button stores the modified contents in the memory. Clicking the **[Cancel]** button closes the dialog box without storing the modified contents.

5.37 Unified TLB Modify Dialog Box



Entry: H'01

Virtual Address: H'01000000

ASID(H'0-H'FF): H'00

Physical Address: H'05000000

Page Size

- ● 1-Kbyte page
- ○ 4-Kbyte page
- ○ 64-Kbyte page
- ○ 1-Mbyte page

Protection

	PV Mode	User Mode
●	Read only	No access
○	Read/write	No access
○	Read only	Read only
○	Read/write	Read/write

- □ Valid
- □ Cacheable
- □ Dirty
- □ Share status
- □ Write through

Figure 5.52 Unified TLB Modify Dialog Box

This dialog box specifies the UTLB contents of the entry selected in the **Unified TLB** dialog box. This dialog box is provided only for the SH-4 series.

The following items can be specified.

[Entry] Entry number selected by the **Unified TLB** dialog box.

[Virtual Address] Virtual address in longword size. Bits 31 to 10 are valid.

[Physical Address] Physical address in longword size. Bits 31 to 10 are valid.

[ASID] Address space ID, which specifies the process that can access the virtual page.

[Valid] Specifies whether or not the entry is valid. Selecting this box makes the entry valid.

[Cacheable] Enables or disables caching. Selecting this box enables caching of the page.

[Dirty] Specifies whether or not the page has been written to. Selecting this box makes the simulator/debugger assume that the page has been written to.

[Share status] Specifies whether or not to share the page with multiple processes. Selecting this box specifies that the page is shared.

[Write through] Write through bit. Specifies the cache writing mode.

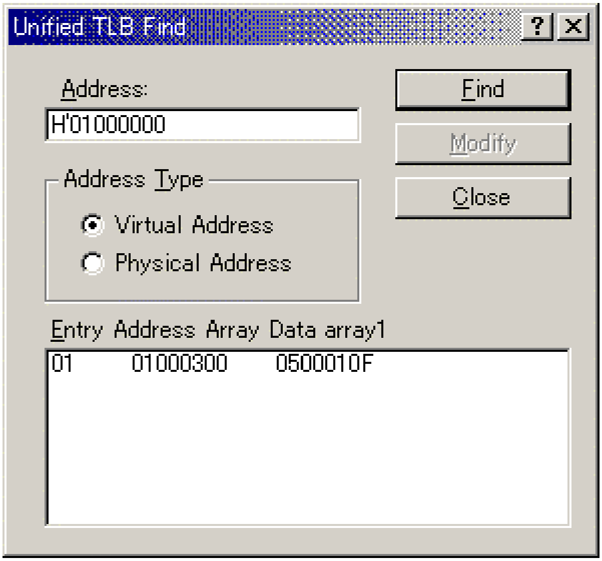
[Page Size] Specifies the page size.

[Protection] Specifies the page protection status.

PV Mode	User Mode	
Read only	No access	Enables read in privileged mode.
Read only	Read only	Enables read in privileged or user mode.
Read/Write	No access	Enables read and write in privileged mode.
Read/Write	Read/Write	Enables read and write in privileged or user mode.

Clicking the **[OK]** button displays the modified contents in the **Unified TLB** dialog box. Clicking the **[Cancel]** button closes the dialog box without modifying the UTLB contents.

5.38 Unified TLB Find Dialog Box



Address:
H'01000000

Address Type
● Virtual Address
○ Physical Address

Entry Address Array Data array1

```
01      01000300      0500010F
```

Figure 5.53 Unified TLB Find Dialog Box

This dialog box searches the UTLB contents. This dialog box is provided only for the SH-4 series.

The following search conditions can be specified.

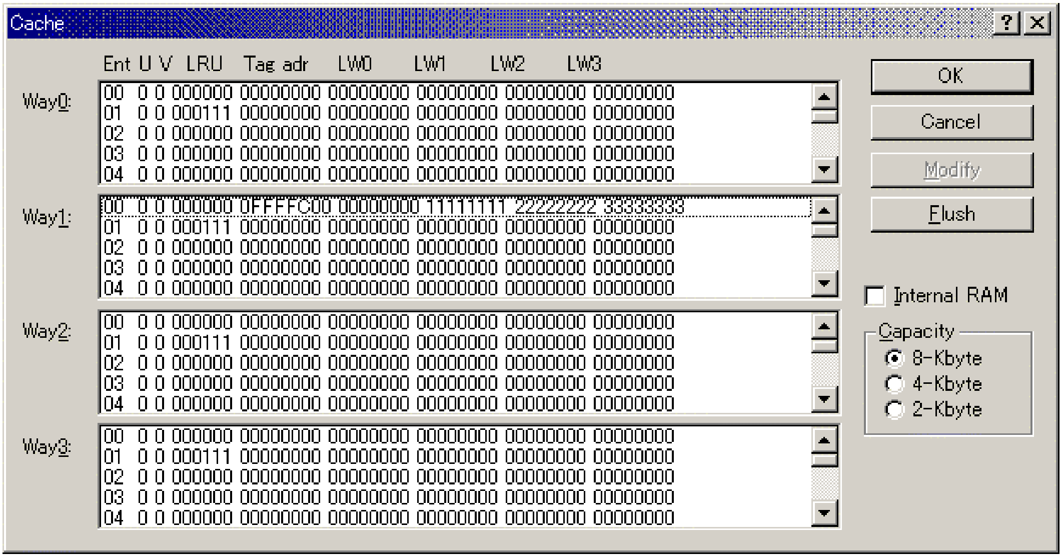
[Address]	Specifies the address to be searched for.
[Address type]	Specifies whether the address to be searched for is virtual or physical.

After specifying the search condition, clicking the **[Find]** button starts search. The search results are displayed in the list box at the bottom of the dialog box, in the order of UTLB entry, address array, and data array 1.

To modify the displayed UTLB contents, select the UTLB entry in the list box, and click the **[Modify]** button. The **Unified TLB Modify** dialog box will open and the UTLB contents can be modified.

Clicking the **[Close]** button closes this dialog box.

5.39 Cache Dialog Box



```
     Ent U V  LRU   Tag adr   LW0      LW1      LW2      LW3
Way0: 00  0 0 000000 00000000 00000000 00000000 00000000 00000000
      01  0 0 000111 00000000 00000000 00000000 00000000 00000000
      02  0 0 000000 00000000 00000000 00000000 00000000 00000000
      03  0 0 000000 00000000 00000000 00000000 00000000 00000000
      04  0 0 000000 00000000 00000000 00000000 00000000 00000000
Way1: 00  0 0 000000 0FFFFC00 00000000 11111111 22222222 33333333
      01  0 0 000111 00000000 00000000 00000000 00000000 00000000
      02  0 0 000000 00000000 00000000 00000000 00000000 00000000
      03  0 0 000000 00000000 00000000 00000000 00000000 00000000
      04  0 0 000000 00000000 00000000 00000000 00000000 00000000
Way2: 00  0 0 000000 00000000 00000000 00000000 00000000 00000000
      01  0 0 000111 00000000 00000000 00000000 00000000 00000000
      02  0 0 000000 00000000 00000000 00000000 00000000 00000000
      03  0 0 000000 00000000 00000000 00000000 00000000 00000000
      04  0 0 000000 00000000 00000000 00000000 00000000 00000000
Way3: 00  0 0 000000 00000000 00000000 00000000 00000000 00000000
      01  0 0 000111 00000000 00000000 00000000 00000000 00000000
      02  0 0 000000 00000000 00000000 00000000 00000000 00000000
      03  0 0 000000 00000000 00000000 00000000 00000000 00000000
      04  0 0 000000 00000000 00000000 00000000 00000000 00000000
```

□ Internal RAM

Capacity
● 8-Kbyte
○ 4-Kbyte
○ 2-Kbyte

Figure 5.54 Cache Dialog Box (for SH-3 and SH-3E Series)

This dialog box displays the cache contents in [Way0] to [Way3]. This dialog box is provided only for the SH-3, SH-3E, SH3-DSP series, and SH2-DSP (SH7612).

The following items are displayed.

[Ent] Entry number in the cache. The specifiable value depends on the CPU.
SH-3 and SH-3E series: H'00 to H'7F
SH3-DSP series: H'00 to H'FF
SH2-DSP (SH7612): H'00 to H'3F

[U] Update bit. When this bit is 1, the entry has been written to.

[V] Validity bit. When this bit is 1, the entry is valid.

[LRU] Numerical string that determines which way's entry should be replaced when a cache miss occurs. (The same LRU value is assigned to the same entry in all ways.)

[Tag adr] Tag address.

[LW0] to [LW3] Longword data 0 to 3 stored in cache.

[Internal RAM]* Internal RAM mode. Selecting this box enables a half of the cache to be used as the internal RAM when 8-Kbyte or 4-Kbyte is selected in **[Capacity]**.

[Capacity]* Cache capacity.

Note: The items marked with * are displayed only for the SH-3 and SH-3E series.

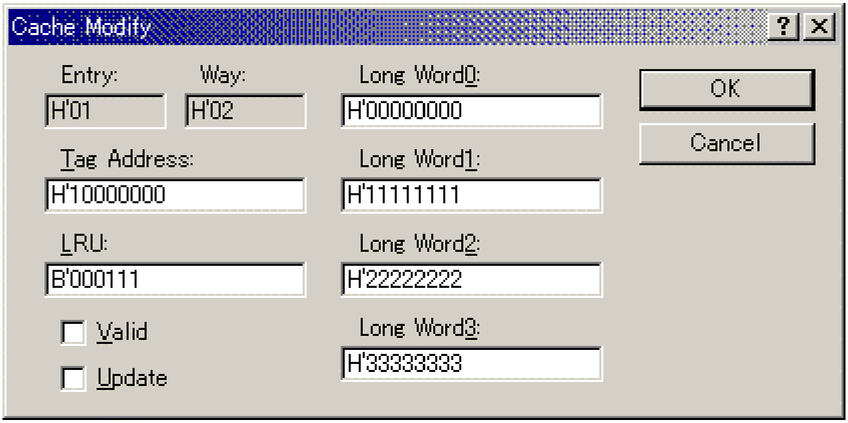
The cache contents can be modified and flushed using the following buttons.

[Modify] Modifies the cache contents. After selecting the entry to be modified in the list box, click the button. The **Cache Modify** dialog box will open and the cache contents can be modified.

[Flush] Flushes all cache contents. Clicking the button clears the V, U, and LRU bits of all entries to zero, and invalidates all cache entries.

Clicking the **[OK]** button stores the modified contents in the memory. Clicking the **[Cancel]** button closes the dialog box without storing the modified contents.

5.40 Cache Modify Dialog Box



Entry: H'01

Way: H'02

Long Word0: H'00000000

Tag Address: H'10000000

Long Word1: H'11111111

LRU: B'000111

Long Word2: H'22222222

□ Valid

□ Update

Long Word3: H'33333333

Figure 5.55 Cache Modify Dialog Box

This dialog box modifies the cache contents of the way and entry selected in the **Cache** dialog box. This dialog box is provided only for the SH-3, SH-3E, SH3-DSP series, and the SH2-DSP (SH7612).

The following items can be specified.

[Entry] Entry number selected by the **Cache** dialog box.

[Way] Way number selected by the **Cache** dialog box.

[Tag Address] Tag address. A longword physical address must be specified. Bits 31 to 10 are valid.

[LRU] Numerical string that determines which way's entry should be replaced when a cache miss occurs. The LRU values for the same entries in the other ways are also modified to the specified value.

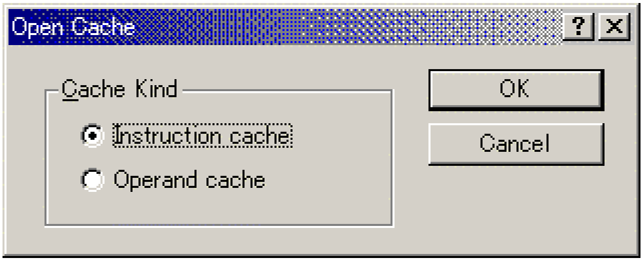
[Valid] Specifies whether or not the entry is valid. Selecting this box makes the entry valid.

[Update] Indicates whether or not the entry has been written to. Selecting this box makes the simulator/debugger assume that the entry has been written to.

[Long Word0] to
[Long Word3] Longword data 0 to 3 to be set to cache entries.

Clicking the **[OK]** button displays the modified contents in the **Cache** dialog box. Clicking the **[Cancel]** button closes the dialog box without displaying the modified contents in the **Cache** dialog box.

5.41 Open Cache Dialog Box



● Instruction cache
○ Operand cache

Figure 5.56 Open Cache Dialog Box

This dialog box selects the cache to be displayed. This dialog box is provided only for the SH-4 series.

In this dialog box, select one of the following caches:

[Instruction cache] Selects the instruction cache (IC).

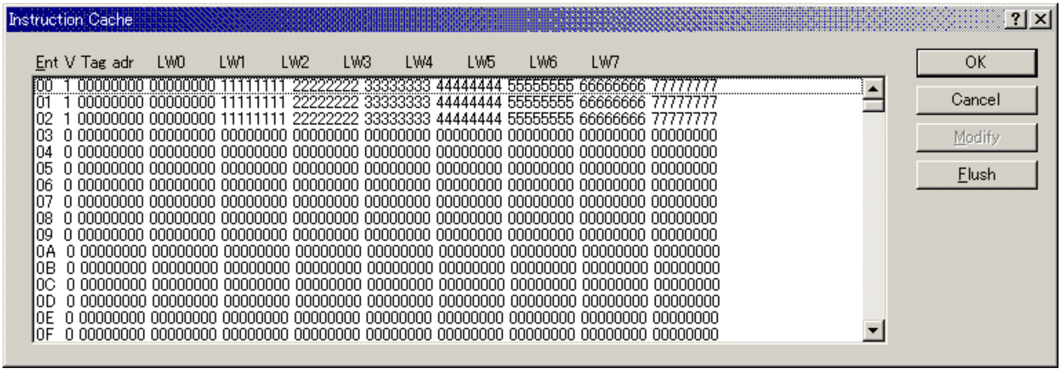
[Operand cache] Selects the operand cache (OC).

Clicking the **[OK]** button displays the selected cache dialog box. Clicking the **[Cancel]** button closes the **Open Cache** dialog box.

5.42 Instruction Cache Dialog Box

This dialog box displays the contents of the IC. This dialog box is provided only for the SH-4 series, and the displayed contents differ according to the target CPU.

SH-4/SH-4BSC:



```
Ent V Tag adr  LW0      LW1      LW2      LW3      LW4      LW5      LW6      LW7
00  1 00000000 00000000 11111111 22222222 33333333 44444444 55555555 66666666 77777777
01  1 00000000 00000000 11111111 22222222 33333333 44444444 55555555 66666666 77777777
02  1 00000000 00000000 11111111 22222222 33333333 44444444 55555555 66666666 77777777
03  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
04  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
05  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
06  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
07  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
08  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
09  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0A  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0B  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0C  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0D  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0E  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0F  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Figure 5.57 Instruction Cache Dialog Box (for SH-4/SH-4BSC)

The following items are displayed.

[Ent] Entry number in the IC (H'00 to H'FF).

[V] Validity bit. When this bit is 1, the entry is valid.

[Tag adr] Tag address.

[LW0] to [LW7] Longword data 0 to 7 stored in IC entries.

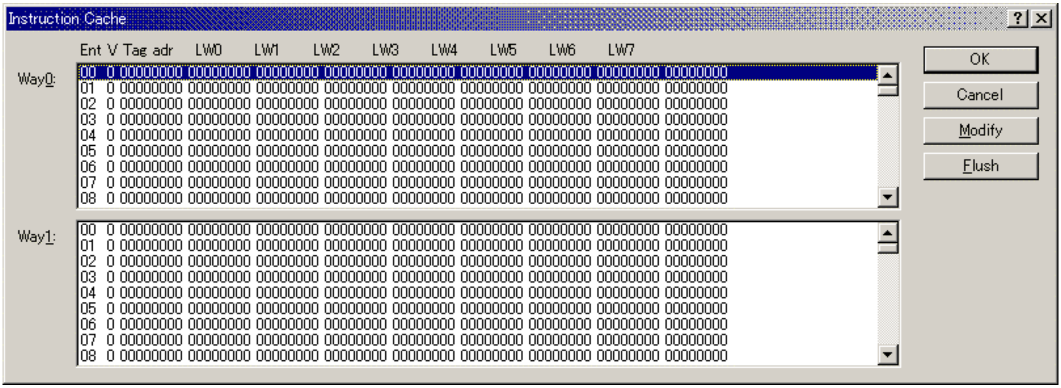
The IC contents can be modified and flushed using the following buttons.

[Modify] Modifies the IC contents. After selecting the entry to be modified in the list box, click the button. The **Instruction Cache Modify** dialog box will open and the IC contents can be modified.

[Flush] Flushes all IC entries. Clicking the button clears the V bits of all entries to zero, and invalidates all IC entries.

Clicking the **[OK]** button stores the modified contents in the memory. Clicking the **[Cancel]** button closes the dialog box without storing the modified contents.

SH-4 (SH7750R):



```
      Ent V Tag adr  LW0      LW1      LW2      LW3      LW4      LW5      LW6      LW7
Way0: 00  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      01  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      02  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      03  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      04  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      05  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      06  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      07  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      08  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Way1: 00  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      01  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      02  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      03  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      04  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      05  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      06  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      07  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      08  0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Figure 5.58 Instruction Cache Dialog Box (for SH-4 (SH7750R))

This dialog box displays the contents of the cache set on Way0 and Way1. The following items are displayed.

[Ent] Entry number in the IC (H'00 to H'FF).

[V] Validity bit. When this bit is 1, the entry is valid.

[Tag adr] Tag address.

[LW0] to [LW7] Longword data 0 to 7 stored in IC entries.

The IC contents can be modified and flushed using the following buttons.

[Modify] Modifies the IC contents. After selecting the entry to be modified in the list box, click the button. The **Instruction Cache Modify** dialog box will open and the IC contents can be modified.

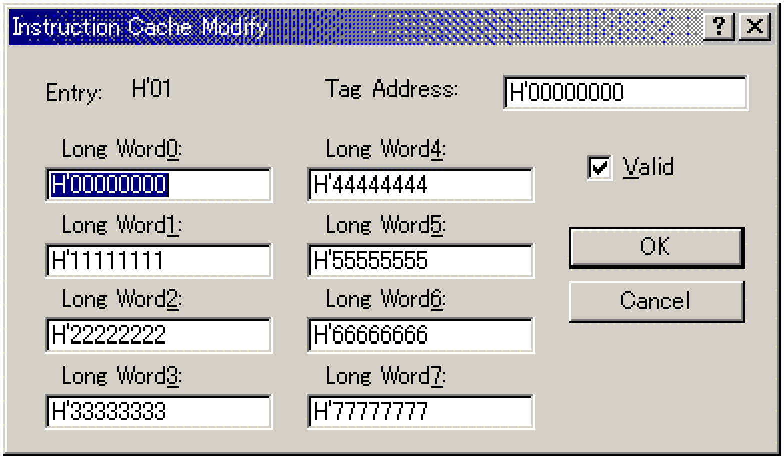
[Flush] Flushes all IC entries. Clicking the button clears the V bits of all entries to zero, and invalidates all IC entries.

Clicking the **[OK]** button stores the modified contents in the memory. Clicking the **[Cancel]** button closes the dialog box without storing the modified contents.

5.43 Instruction Cache Modify Dialog Box

This dialog box modifies the IC contents of the entry selected in the **Instruction Cache** dialog box. This dialog box is provided only for the SH-4 series, and the displayed contents differ according to the target CPU.

SH-4/SH-4BSC:



Entry: H'01

Tag Address: H'00000000

Long Word0: H'00000000

Long Word1: H'11111111

Long Word2: H'22222222

Long Word3: H'33333333

Long Word4: H'44444444

Long Word5: H'55555555

Long Word6: H'66666666

Long Word7: H'77777777

☑ Valid

Figure 5.59 Instruction Cache Modify Dialog Box (for SH-4/SH-4BSC)

The following items can be specified.

[Entry] Displays the entry number selected by the **Instruction Cache** dialog box.

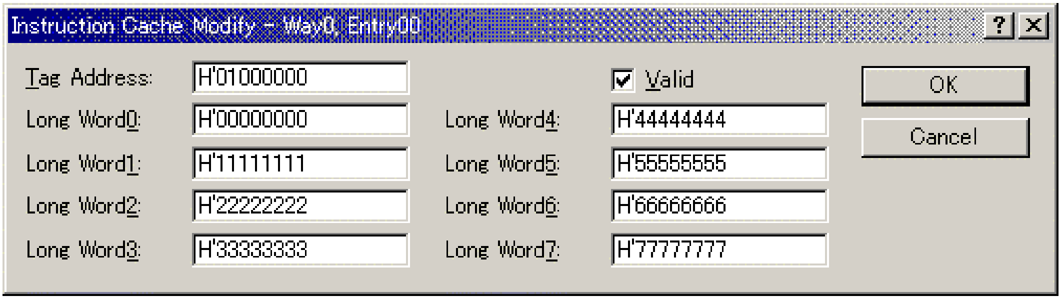
[Tag Address] Tag address. A longword physical address must be specified. Bits 31 to 10 are valid.

[Valid] Indicates whether or not the entry is valid. Selecting this box makes the entry valid.

[Long Word0] to
[Long Word7] Longword data 0 to 7 to be set to IC entries.

Clicking the **[OK]** button displays the modified contents in the **Instruction Cache** dialog box. Clicking the **[Cancel]** button closes the dialog box without displaying the modified contents in the **Instruction Cache** dialog box.

SH-4 (SH7750R):



Instruction Cache Modify - Way0, Entry00

Tag Address: H'01000000

☑ Valid

Long Word0: H'00000000

Long Word1: H'11111111

Long Word2: H'22222222

Long Word3: H'33333333

Long Word4: H'44444444

Long Word5: H'55555555

Long Word6: H'66666666

Long Word7: H'77777777

Figure 5.60 Instruction Cache Modify Dialog Box (for SH-4 (SH7750R))

The following items can be specified.

[Tag Address] Tag address. A longword physical address must be specified. Bits 31 to 10 are valid.

[Valid] Indicates whether or not the entry is valid. Selecting this box makes the entry valid.

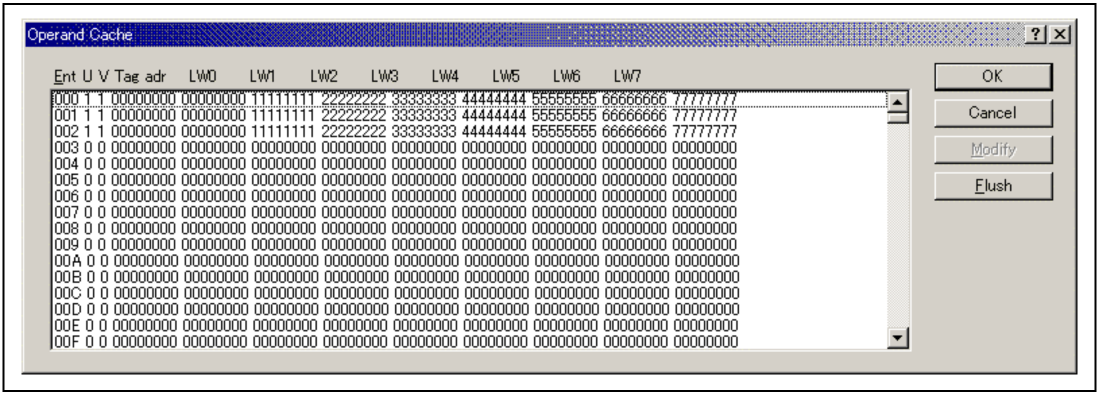
[Long Word0] to
[Long Word7] Longword data 0 to 7 to be set to IC entries.

Clicking the **[OK]** button displays the modified contents in the **Instruction Cache** dialog box. Clicking the **[Cancel]** button closes the dialog box without displaying the modified contents in the **Instruction Cache** dialog box.

5.44 Operand Cache Dialog Box

This dialog box displays the contents of the OC. This dialog box is provided only for the SH-4 series, and the displayed contents differ according to the target CPU.

SH-4/SH-4BSC:



```
Ent U V Tag adr   LW0      LW1      LW2      LW3      LW4      LW5      LW6      LW7
000 1 1 00000000 00000000 11111111 22222222 33333333 44444444 55555555 66666666 77777777
001 1 1 00000000 00000000 11111111 22222222 33333333 44444444 55555555 66666666 77777777
002 1 1 00000000 00000000 11111111 22222222 33333333 44444444 55555555 66666666 77777777
003 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
004 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
005 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
006 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
007 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
008 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
009 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00A 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00B 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00C 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00D 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00E 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00F 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Figure 5.61 Operand Cache Dialog Box (for SH-4/SH-4BSC)

The following items are displayed.

[Ent] Entry number in the OC (H'000 to H'1FF).

[U] Update bit. When this bit is 1, the entry has been written to.

[V] Validity bit. When this bit is 1, the entry is valid.

[Tag adr] Tag address.

[LW0] to [LW7] Longword data 0 to 7 stored in OC entries.

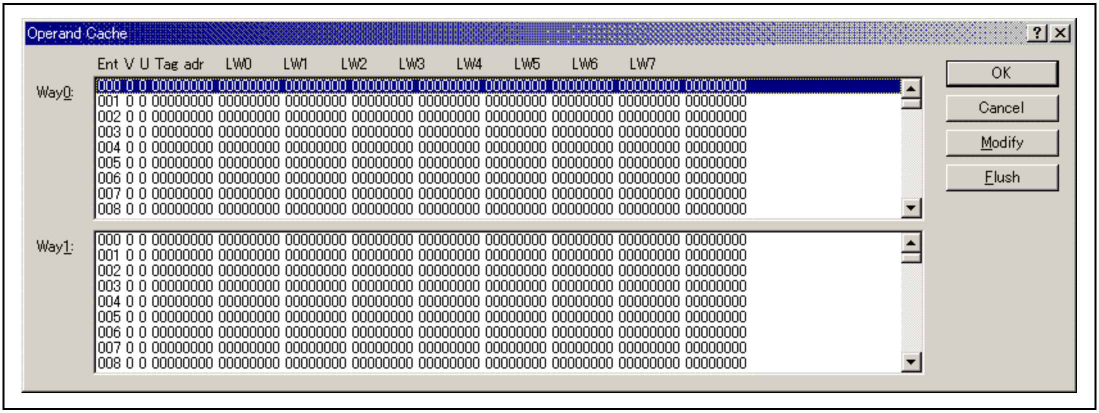
The OC contents can be modified and flushed using the following buttons.

[Modify] Modifies the OC contents. After selecting the entry to be modified in the list box, click the button. The **Operand Cache Modify** dialog box will open and the OC contents can be modified.

[Flush] Flushes all OC entries. Clicking the button clears the U and V bits of all entries to zero, and invalidates all OC entries.

Clicking the **[OK]** button stores the modified contents in the memory. Clicking the **[Cancel]** button closes the dialog box without storing the modified contents.

SH-4 (SH7750R):



```
Ent V U Tag adr  LW0      LW1      LW2      LW3      LW4      LW5      LW6      LW7
Way0:
000 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
001 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
002 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
003 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
004 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
005 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
006 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
007 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
008 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Way1:
000 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
001 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
002 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
003 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
004 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
005 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
006 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
007 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
008 0 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Figure 5.62 Operand Cache Dialog Box (for SH-4 (SH7750R))

This dialog box displays the contents of the cache set on Way0 and Way1. The following items are displayed.

[Ent] Entry number in the OC (H'000 to H'1FF).

[V] Validity bit. When this bit is 1, the entry is valid.

[U] Update bit. When this bit is 1, the entry has been written to.

[Tag adr] Tag address.

[LW0] to [LW7] Longword data 0 to 7 stored in OC entries.

The OC contents can be modified and flushed using the following buttons.

[Modify] Modifies the OC contents. After selecting the entry to be modified in the list box, click the button. The **Operand Cache Modify** dialog box will open and the OC contents can be modified.

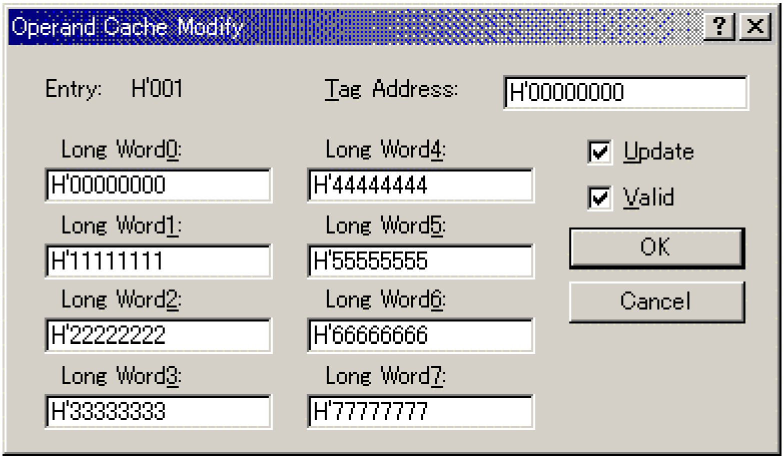
[Flush] Flushes all OC entries. Clicking the button clears the U and V bits of all entries to zero, and invalidates all OC entries.

Clicking the **[OK]** button stores the modified contents in the memory. Clicking the **[Cancel]** button closes the dialog box without storing the modified contents.

5.45 Operand Cache Modify Dialog Box

This dialog box modifies the OC contents of the entry selected in the **Operand Cache** dialog box. This dialog box is provided only for the SH-4 series, and the displayed contents differ according to the target CPU.

SH-4/SH-4BSC:



Operand Cache Modify

Entry: H'001

Tag Address: H'00000000

Long Word0: H'00000000

Long Word1: H'11111111

Long Word2: H'22222222

Long Word3: H'33333333

Long Word4: H'44444444

Long Word5: H'55555555

Long Word6: H'66666666

Long Word7: H'77777777

☑ Update

☑ Valid

Figure 5.63 Operand Cache Modify Dialog Box (for SH-4/SH-4BSC)

The following items can be specified.

[Entry] Entry number selected by the **Operand Cache** dialog box.

[Tag Address] Tag address. A longword physical address must be specified. Bits 31 to 10 are valid.

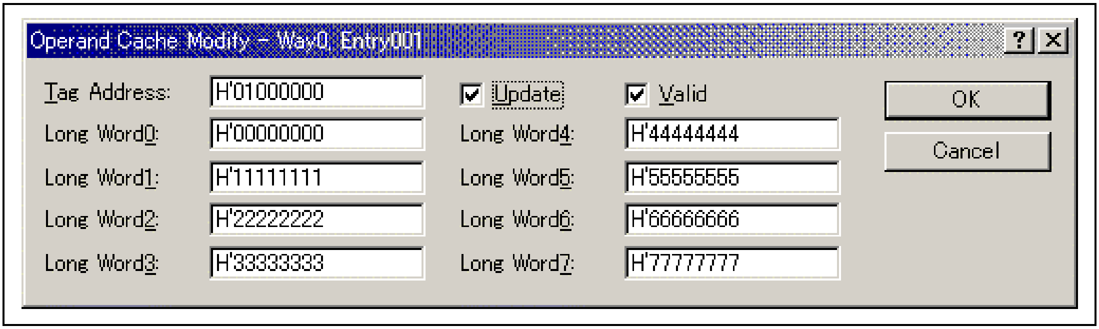
[Update] Indicates whether or not the entry has been written to. Selecting this box makes the simulator/debugger assume that the entry has been written to.

[Valid] Indicates whether or not the entry is valid. Selecting this box makes the entry valid.

[Long Word0] to
[Long Word7] Longword data 0 to 7 to be set to OC entries.

Clicking the **[OK]** button displays the modified contents in the **Operand Cache** dialog box. Clicking the **[Cancel]** button closes the dialog box without displaying the modified contents in the **Operand Cache** dialog box.

SH-4 (SH7750R):



Operand Cache Modify – Way0, Entry001

Tag Address: H'01000000 ☑ Update ☑ Valid

Long Word0: H'00000000 Long Word4: H'44444444

Long Word1: H'11111111 Long Word5: H'55555555

Long Word2: H'22222222 Long Word6: H'66666666

Long Word3: H'33333333 Long Word7: H'77777777

Figure 5.64 Operand Cache Modify Dialog Box (for SH-4 (SH7750R))

The following items can be specified.

[Tag Address] Tag address. A longword physical address must be specified. Bits 31 to 10 are valid.

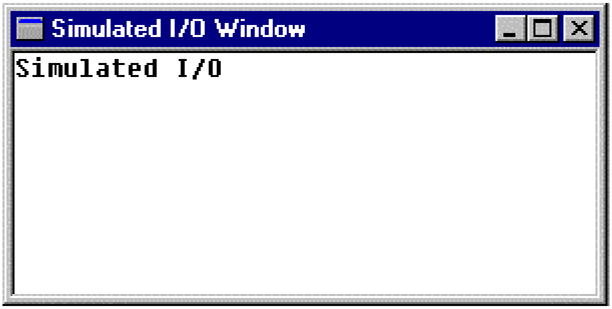
[Update] Indicates whether or not the entry has been written to. Selecting this box makes the simulator/debugger assume that the entry has been written to.

[Valid] Indicates whether or not the entry is valid. Selecting this box makes the entry valid.

[Long Word0] to
[Long Word7] Longword data 0 to 7 to be set to OC entries.

Clicking the **[OK]** button displays the modified contents in the **Operand Cache** dialog box. Clicking the **[Cancel]** button closes the dialog box without displaying the modified contents in the **Operand Cache** dialog box.

5.46 Simulated I/O Window



Simulated I/O

Figure 5.65 Simulated I/O Window

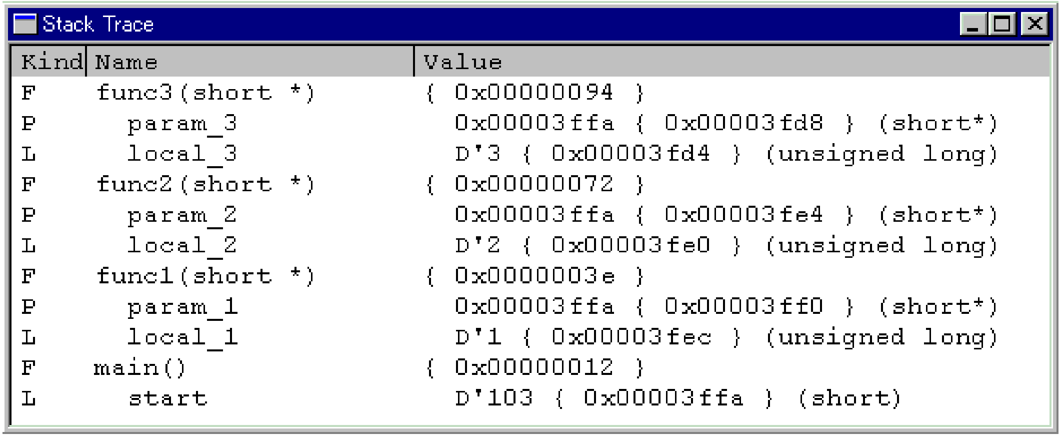
This window is for standard I/O and file I/O system calls from the user program.

Clicking the right mouse button on the **Simulated I/O** window displays the following popup menus.

[Copy]	Copies the highlighted text to the Windows® clipboard so that the text can be pasted to another application.
[Paste]	Pastes the text from the Windows® clipboard to the **Simulated I/O** window.
[Erase All]	Clears the contents of the **Simulated I/O** window.

For the I/O processing, refer to section 3.12, Standard I/O and File I/O Processing.

5.47 Stack Trace Window



Kind	Name	Value
F	func3(short *)	{ 0x00000094 }
P	param_3	0x00003ffa { 0x00003fd8 } (short*)
L	local_3	D'3 { 0x00003fd4 } (unsigned long)
F	func2(short *)	{ 0x00000072 }
P	param_2	0x00003ffa { 0x00003fe4 } (short*)
L	local_2	D'2 { 0x00003fe0 } (unsigned long)
F	func1(short *)	{ 0x0000003e }
P	param_1	0x00003ffa { 0x00003ff0 } (short*)
L	local_1	D'1 { 0x00003fec } (unsigned long)
F	main()	{ 0x00000012 }
L	start	D'103 { 0x00003ffa } (short)

Figure 5.66 Stack Trace Window

This window displays the function call history.

The following items are displayed.

[Kind] Indicates the type of the symbol.
F: Function
P: Function parameter
L: Local variable

[Name] Indicates the symbol name.

[Value] Indicates the value, address, and type of the symbol.

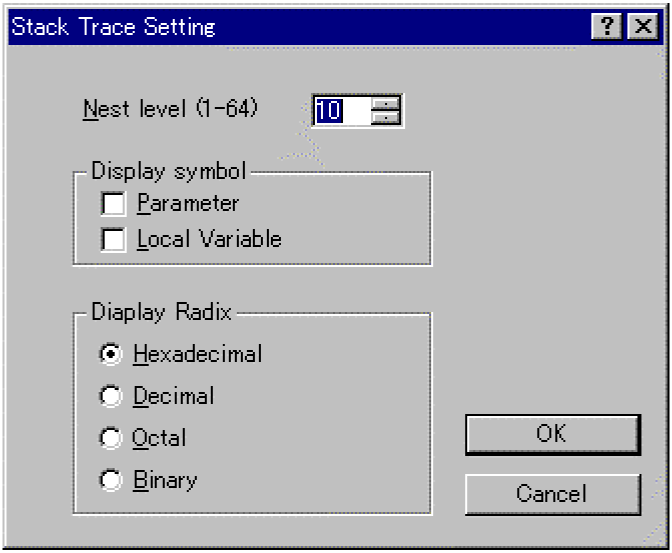
Right-clicking on the mouse within the window displays a popup menu. Supported menu options are described in the following sections:

5.47.1 Go to Source

Displays, in the **Source** window, the source program corresponding to the selected function.

5.47.2 View Setting...

Launches the **Stack Trace Setting** dialog box, allowing the user to specify the **Stack Trace** window settings.



Nest level (1-64) 10

Display symbol
☐ Parameter
☐ Local Variable

Diaplay Radix
● Hexadecimal
○ Decimal
○ Octal
○ Binary

Figure 5.67 Stack Trace Setting Dialog Box

[Nest level] Specifies the level of function call nesting to be displayed in the **Stack Trace** window.

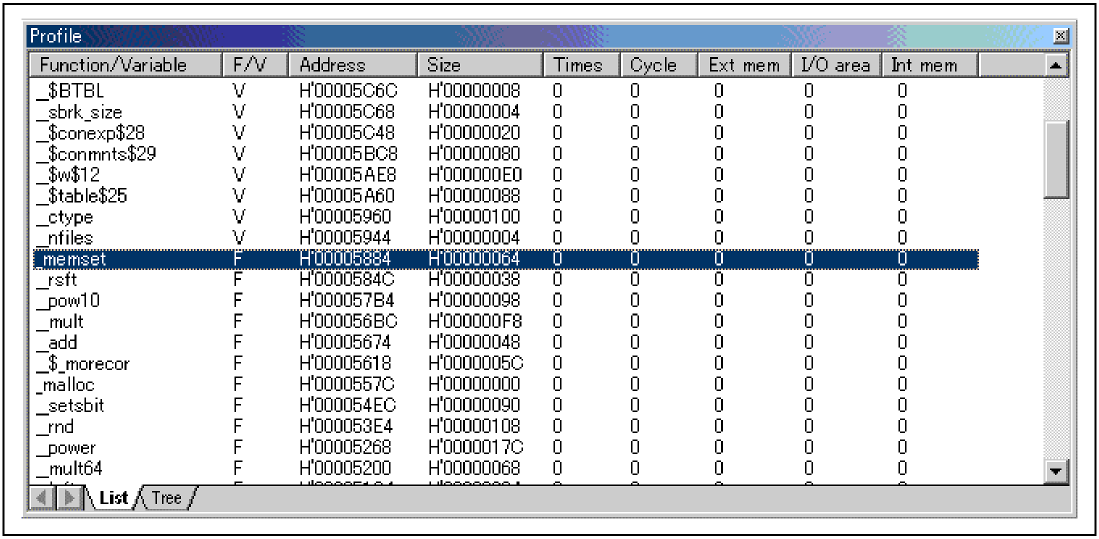
[Display symbol] Specifies the symbol types to be displayed in addition to functions.

[Display Radix] Specifies the radix for displays in the **Stack Trace** window.

5.47.3 Copy

Copies the highlighted text into the Windows® clipboard, allowing it to be pasted into other applications.

5.48 Profile Window (List Sheet)



Profile

Function/Variable	F/V	Address	Size	Times	Cycle	Ext mem	I/O area	Int mem
_$BTBL	V	H'00005C6C	H'00000008	0	0	0	0	0
_sbrk_size	V	H'00005C68	H'00000004	0	0	0	0	0
_$conexp$28	V	H'00005C48	H'00000020	0	0	0	0	0
_$conmnts$29	V	H'00005BC8	H'00000080	0	0	0	0	0
_w12	V	H'00005AE8	H'000000E0	0	0	0	0	0
_$table$25	V	H'00005A60	H'00000088	0	0	0	0	0
_ctype	V	H'00005960	H'00000100	0	0	0	0	0
_nfiles	V	H'00005944	H'00000004	0	0	0	0	0
_memset	F	H'00005884	H'00000064	0	0	0	0	0
_rsft	F	H'0000584C	H'00000038	0	0	0	0	0
_pow10	F	H'000057B4	H'00000098	0	0	0	0	0
_mult	F	H'000056BC	H'000000F8	0	0	0	0	0
_add	F	H'00005674	H'00000048	0	0	0	0	0
_$_morecor	F	H'00005618	H'0000005C	0	0	0	0	0
_malloc	F	H'0000557C	H'00000000	0	0	0	0	0
_setsbit	F	H'000054EC	H'00000090	0	0	0	0	0
_rnd	F	H'000053E4	H'00000108	0	0	0	0	0
_power	F	H'00005268	H'0000017C	0	0	0	0	0
_mult64	F	H'00005200	H'00000068	0	0	0	0	0

Figure 5.68 Profile Window (List Sheet)

This window displays the address and size of a function or a global variable, the number of times the function is called or the global variable is accessed, and profile data. Displayed profile data differs according to the target CPU as follows:

SH-1/SH-2/SH-2E Series, SH2-DSP (SH7410), SH2-DSP (Core), and SH2-DSP (SH7065):
Times (the number of times a function is called or a global variable is accessed)
Cycle (the number of execution cycles)
Ext_mem (the number of external memory accesses)
I/O_area (the number of internal I/O accesses)
Int_mem (the number of internal memory accesses)

SH-3/SH-3E/SH3-DSP Series and SH2-DSP (SH7612):
Times (the number of times a function is called or a global variable is accessed)
Cycle (the number of execution cycles)
Cache miss (the number of cache misses)
Ext_mem (the number of external memory accesses)
I/O_area (the number of internal I/O accesses)
Int_mem (the number of internal memory accesses)

SH-4 Series:
Times (the number of times a function is called or a global variable is accessed)
Cycle (the number of execution cycles)
ICache miss (the number of instruction cache misses)
OCache miss (number of operand cache misses)

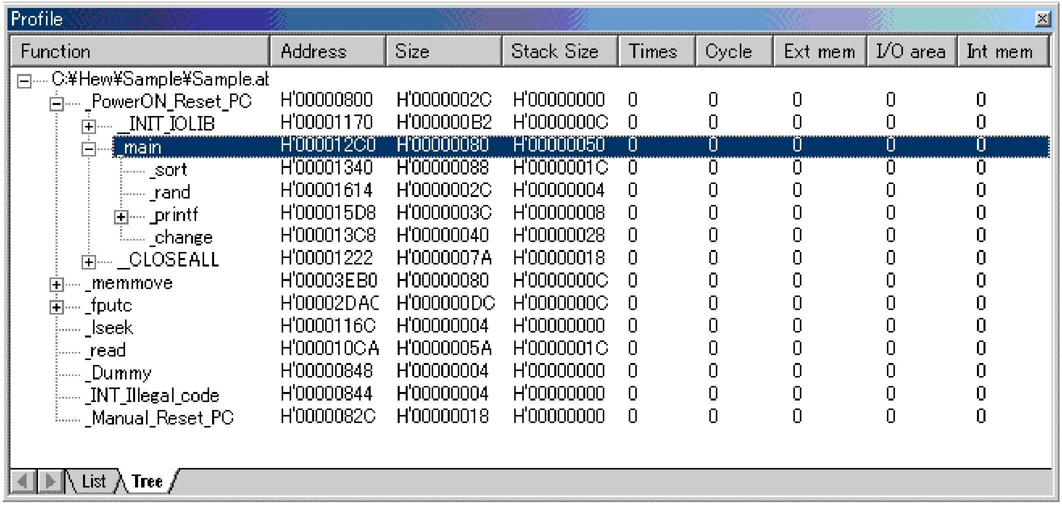
Ext_mem (the number of external memory accesses)
I/O_area (the number of internal I/O accesses)
Int_mem (the number of internal memory accesses)

The number of execution cycles and cache misses are calculated by subtracting the total execution cycles or cache misses at a specific function call instruction execution from the total execution cycles or cache misses at a return instruction execution of a specific function.

When the column header is clicked, data are sorted in alphabetic or numeric ascending/descending order.

Double-clicking the **Function/Variable** or **Address** column displays the source program corresponding to the address in the line. Right-clicking on the mouse within the window displays a popup menu. For details on this popup menu, refer to section 5.49, Profile Window (Tree Sheet).

5.49 Profile Window (Tree Sheet)



Function	Address	Size	Stack Size	Times	Cycle	Ext mem	I/O area	Int mem
C:¥Hew¥Sample¥Sample.at								
_PowerON_Reset_PC	H'00000800	H'0000002C	H'00000000	0	0	0	0	0
__INIT_IOLIB	H'00001170	H'000000B2	H'0000000C	0	0	0	0	0
_main	H'000012C0	H'00000080	H'00000050	0	0	0	0	0
_sort	H'00001340	H'00000088	H'0000001C	0	0	0	0	0
_rand	H'00001614	H'0000002C	H'00000004	0	0	0	0	0
_printf	H'000015D8	H'0000003C	H'00000008	0	0	0	0	0
_change	H'000013C8	H'00000040	H'00000028	0	0	0	0	0
__CLOSEALL	H'00001222	H'0000007A	H'00000018	0	0	0	0	0
_memmove	H'00003EB0	H'00000080	H'0000000C	0	0	0	0	0
_fputc	H'00002DAC	H'000000DC	H'0000000C	0	0	0	0	0
_lseek	H'0000116C	H'00000004	H'00000000	0	0	0	0	0
_read	H'000010CA	H'0000005A	H'0000001C	0	0	0	0	0
_Dummy	H'00000848	H'00000004	H'00000000	0	0	0	0	0
_INT_Illegal_code	H'00000844	H'00000004	H'00000000	0	0	0	0	0
_Manual_Reset_PC	H'0000082C	H'00000018	H'00000000	0	0	0	0	0

Figure 5.69 Profile Window (Tree Sheet)

This window displays the relation of function calls in a tree structure. Displayed contents are the address, size, stack size, number of function calls, and profile data. The stack size, number of function calls, and profile data are values when the function is called.

Displayed profile data differ according to the target CPU as follows:

SH-1/SH-2/SH-2E Series, SH2-DSP (SH7410), SH2-DSP (Core), and SH2-DSP (SH7065):
Times (the number of times a function is called)
Cycle (the number of execution cycles)

Ext_mem (the number of external memory accesses)
I/O_area (the number of input/output area accesses)
Int_mem (the number of internal memory accesses)

SH-3/SH-3E/SH3-DSP Series and SH2-DSP (SH7612):
Times (the number of times a function is called)
Cycle (the number of execution cycles)
Cache miss (the number of cache misses)
Ext_mem (the number of external memory accesses)
I/O_area (the number of input/output area accesses)
Int_mem (the number of internal memory accesses)

SH-4 Series:
Times (the number of times a function is called)
Cycle (the number of execution cycles)
ICache miss (the number of instruction cache misses)
OCache miss (the number of operand cache misses)
Ext_mem (the number of external memory accesses)
I/O_area (the number of input/output area accesses)
Int_mem (the number of internal memory accesses)

The number of execution cycles and cache misses are calculated by subtracting the total execution cycles or cache misses at a specific function call instruction execution from the total execution cycles or cache misses at a return instruction execution of a specific function.

Note: Displayed stack size does not represent the actual size. Use it as a reference value when the function is called. If there is no stack information file (.sni extension) output from the optimizing linkage editor, the stack size is not displayed. For details of the stack information file, refer to the manual of the optimizing linkage editor.

Double-clicking a function in the Function column expands or reduces the tree structure display. The expansion or reduction is also provided by the “+” or “-“ key. Double-clicking the Address column displays the source program corresponding to the specific address.

Right-clicking on the mouse within the window displays a popup menu. Supported menu options are described in the following sections:

5.49.1 View Source

Displays the source program or disassembled memory contents for the address in the selected line.

5.49.2 View Profile-Chart

Displays the **Profile-Chart** window focused on the function in the specified line.

5.49.3 Enable Profiler

Toggles acquisition profile data. When profile data acquisition is active, a check mark is shown to the left of the menu text.

5.49.4 Not trace the function call

Stops tracing function calls while profile data is acquired. This menu is used when acquiring profile data of the program in which functions are called in a special way, such as task switching in the OS.

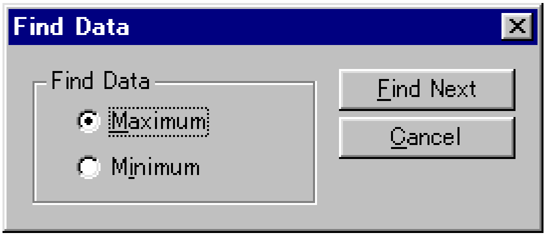
To display the relation of function calls in the **Tree** sheet of the **Profile** window, acquire profile data without selecting this menu. In addition, do not select this menu when optimizing the program by the optimizing linkage editor using the acquired profile information file.

5.49.5 Find…

Displays the **Find Text** dialog box to find a character string in the Function column. Search is started by inputting a character string to be found in the edit box and clicking **[Find Next]** or pressing ENTER.

5.49.6 Find Data…

Displays the **Find Data** dialog box. When the cursor is in the Function column, this menu option is displayed in gray characters.



Find Data

● Maximum
○ Minimum

Figure 5.70 Find Data Dialog Box

By selecting the column to be searched in the **Column** combo box and the search type in the **Find Data** group and entering **[Find Next]** button or **ENTER** key, search is started. If the **[Find Next]** button or the **ENTER** key is input repeatedly, the second larger data (the second smaller data when the Minimum is specified) is searched for.

5.49.7 Clear Data

Clears the number of times functions are called and profile data. Data in the **Profile** window **List** sheet and the **Profile-Chart** window are also cleared.

5.49.8 Output Profile Information Files…

Displays the **Save Profile Information Files** dialog box. Profiling results are saved in a profile information file (.pro extension). The optimizing linkage editor optimizes user programs according to the profile information in this file. For details of the optimization using the profile information, refer to the manual of the optimizing linkage editor.

Note: If profile information has been acquired by selecting the Not trace the function call menu, the program cannot be optimized by the optimizing linkage editor.

5.49.9 Output Text File…

Displays the **Save Text of Profile Data** dialog box. Displayed contents are saved in a text file.

5.49.10 Setting

This menu has the following submenus (the menus available only in the **List** sheet are also included).

1. Show Functions/Variables

Displays both functions and global variables in the Function/Variable column.

2. Show Functions

Displays only functions in the Function/Variable column.

3. Show Variables

Displays only global variables in the Function/Variable column.

4. Only Executed Functions

Only displays the executed functions. If a stack information file (.sni extension) output from the optimizing linkage editor does not exist in the directory where the load module is located, only the executed functions are displayed even if this check box is not checked.

5. Include Data of Child Functions

Sets whether or not to display information for a child function called in the function as profile data.

5.49.11 Properties...

This menu cannot be used in this simulator/debugger.

5.50 Profile Chart Window

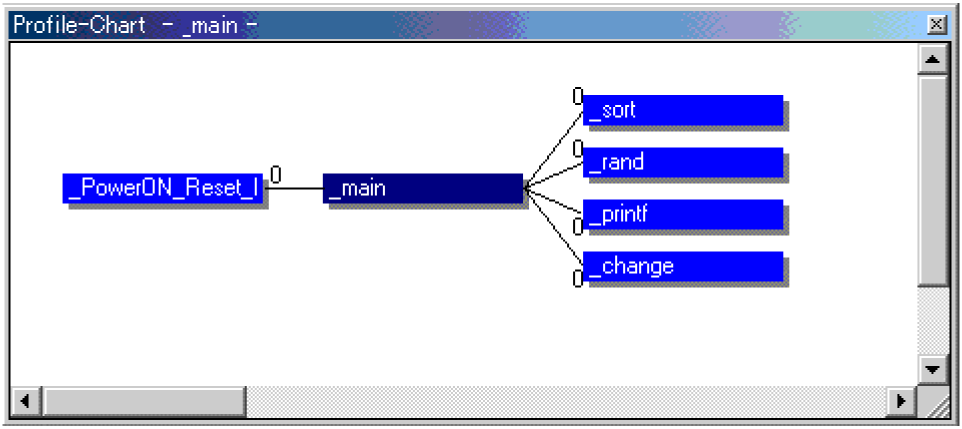


Figure 5.71 Profile-Chart Window

This window displays the relation of calls for a specific function. This window displays the calling relation for the function specified in the List sheet or Tree sheet in the **Profile** window. The specified function is displayed in the middle, the calling function on the left side, and the called function on the right side. Values beside the calling and called functions show the number of times the function has been called.

Right-clicking on the mouse within the window displays a popup menu. Supported menu options are described in the following.

5.50.1 View Source

Displays the source program or disassembled memory contents for the address of the function on which the cursor is placed when the right side button of the mouse is clicked. If the cursor is not placed on a function when the right side button is clicked, this menu option is displayed in gray characters.

5.50.2 View Profile-Chart

Displays the **Profile-Chart** window for the specific function on which the cursor is placed when the right side button of the mouse is clicked. If the cursor is not placed on a function when the right side button is clicked, this menu option is displayed in gray characters.

5.50.3 Enable Profiler

Toggles acquisition of profile data. When profile data acquisition is active, a check mark is shown to the left of the menu text.

5.50.4 Clear Data

Clears the number of times functions are called and profile data. Data in the **List** sheet and **Tree** sheet in the **Profile** window are also cleared.

5.50.5 Multiple View

If the **Profile-Chart** window is going to be opened when it has already been opened, selects whether another window is to be opened or the same window is to be used to display data. When a check mark is shown to the left side of the menu text, another window is opened.

5.50.6 Output Profile Information File…

Displays the **Save Profile Information File** dialog box. Profiling results are saved in a profile information file (.pro extension). The optimizing linkage editor optimizes user programs according to the profile information in this file. For details of the optimization using the profile information, refer to the manual of the optimizing linkage editor.

5.50.7 Expands Size

Expands spaces between each function. The “+” key can also be used to expand spaces.

5.50.8 Reduces Size

Reduces spaces between each function. The “-” key can also be used to reduce spaces.

5.51 Image View Window

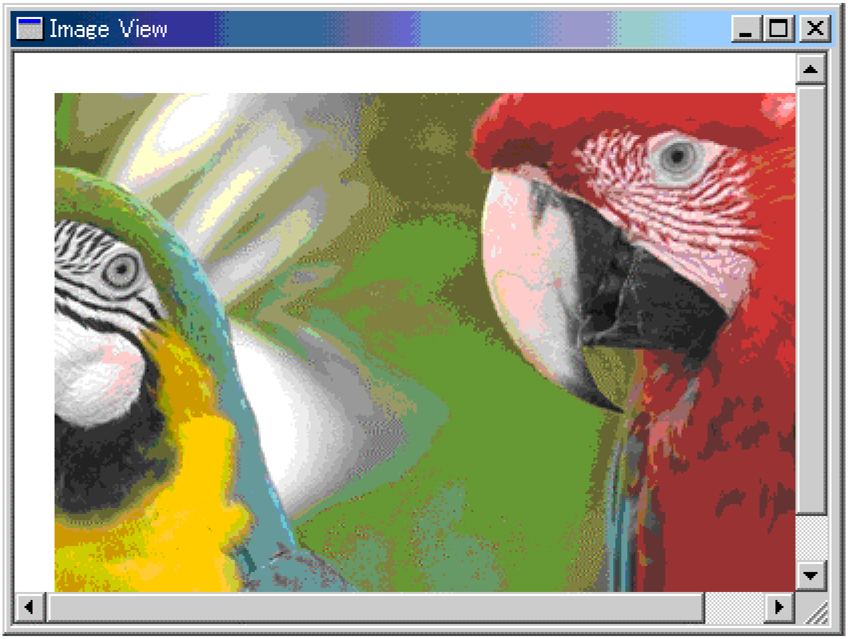


Figure 5.72 Image View Window

Displays the contents of memory in an image. The method used to display the image can be specified in the **Image Properties** dialog box.

Double-clicking within the window displays information, in the Pixel Information dialog box, on the pixel on which the mouse pointer is located.

Right-clicking on the mouse within the window displays a popup menu. Supported menu options are described in the subsequent sections.

5.51.1 Auto Refresh

When this menu is checked, the contents of the window is automatically updated when the user program is executed.

5.51.2 Refresh Now

Updates the contents of the window.

5.51.3 Property…

Displays the **Image Property** dialog box and specifies the format of the displayed image.

5.52 Image Properties Dialog Box

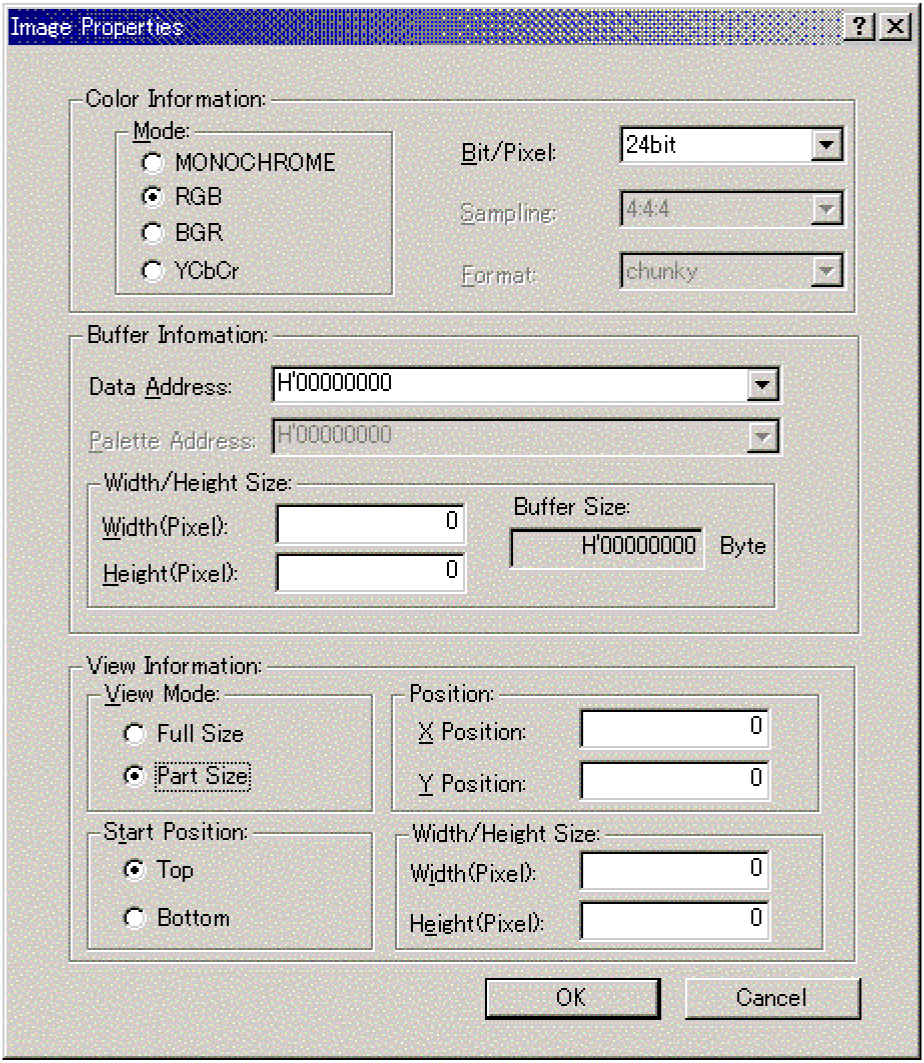


Image Properties

Color Information:

Mode:

○ MONOCHROME
● RGB
○ BGR
○ YCbCr

Bit/Pixel: 24bit

Sampling: 4:4:4

Format: chunky

Buffer Infomation:

Data Address: H'00000000

Palette Address: H'00000000

Width/Height Size:

Width(Pixel): 0

Height(Pixel): 0

Buffer Size: H'00000000 Byte

View Information:

View Mode:

○ Full Size
● Part Size

Position:

X Position: 0

Y Position: 0

Start Position:

● Top
○ Bottom

Width/Height Size:

Width(Pixel): 0

Height(Pixel): 0

Figure 5.73 Image Properties Dialog Box

Specifies the method to display the **Image** window.

[Color Information] Specifies the color information of the image to be displayed.

- [Mode] Specifies the format.
  - [MONOCHROME] Displays in black and white.
  - [RGB] Displayed in R (red), G (green), and B (blue)
  - [BGR] Displayed in B (blue), G (green), and R (red)
  - [YCbCr] Displayed by Y (brightness), Cb (color difference in blue), and Cr (color difference in red)

- [Bit/Pixel] Specifies Bit/Pixel according to the selected [Mode]. (Valid when RGB/BGR is selected)

- [Sampling] Specifies the format of sampling. (Valid when YCbCr is selected)

- [Format] Specifies Chunky/planar. (Valid when YCbCr is selected)

[Buffer Information] Specifies the area to store data, size, and the address of the palette.

- [Data Address] Specifies the start address of memory where image data is to be displayed. (Displayed in hexadecimal)

- [Palette Address] Specifies the start address of memory of the color palette data. (Displayed in hexadecimal) (Valid when 8Bit is selected for RGB/ BGR)

- [Width/Height Size] Specifies the width and height of the image.
  - [Width (Pixel)] Specifies the width of the image. (When a prefix is omitted, the values are input and displayed in decimal.)
  - [Height (Pixel)] Specifies the height of the image. (When a prefix is omitted, the values are input and displayed in decimal.)
  - [Buffer Size] Displays the buffer size of the image from the width and height (Displayed in hexadecimal)

[View Information] Specifies the location, size, and the data start location of the part displayed among the entire image.

[View Mode] Specifies the entire/part to be displayed in the image.

- [Full Size] Displays the entire image.
- [Part Size] Displays part of the image.

[Start Position]

- [Top] Displays data from the upper left.
- [Bottom] Displays data from the lower left.

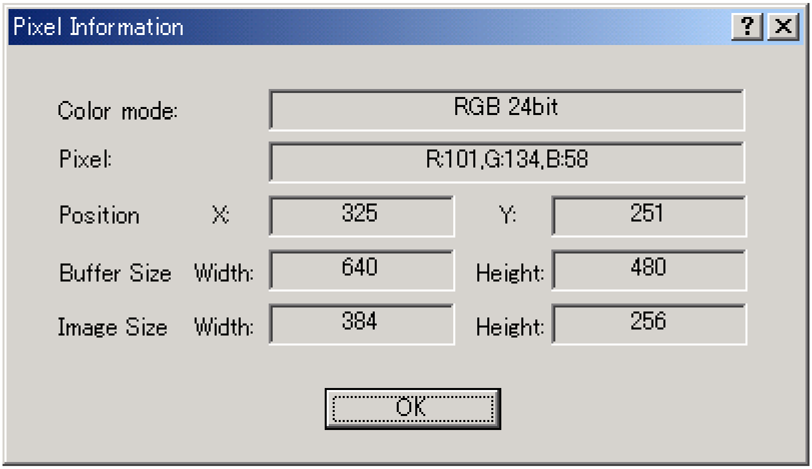
[Position] Specifies the start position of the image where part of the image is to be displayed. This is valid when [Part Size] is selected.

- [X Position] Specifies the X axis of the start location. (When a prefix is omitted, the values are input and displayed in decimal.)
- [Y Position] Specifies the Y axis of the start location. (When a prefix is omitted, the values are input and displayed in decimal.)

[Width/Height Size] Specifies the height and width of the image to be displayed partly.

- [Width(Pixel)] Displays the width of display. (When a prefix is omitted, the values are input and displayed in decimal.)
- [Height(Pixel)] Displays the height of display. (When a prefix is omitted, the values are input and displayed in decimal.)

5.53 Pixel Information Dialog Box



Color mode: RGB 24bit

Pixel: R:101,G:134,B:58

Position X: 325 Y: 251

Buffer Size Width: 640 Height: 480

Image Size Width: 384 Height: 256

Figure 5.74 Pixel Information Window Dialog Box

The cursor location displays pixel information.

[Color Mode] Displays the format of the image.

[Pixel] Displays color information of the cursor location. (Displayed in decimal)

[Position] Displays the cursor location in X and Y axis. (Displayed in decimal)
[X] Displays the X axis of the cursor location.
[Y] Displays the Y axis of the cursor location.

[Buffer Size] Displays the buffer size. (Displayed in decimal)
[Width] Displays the buffer width.
[Height] Displays the buffer height.

[Image Size] Displays the width and height of the display. (Displayed in decimal)
[Width] Displays the width.
[Height] Displays the height.

5.54 Waveform Window

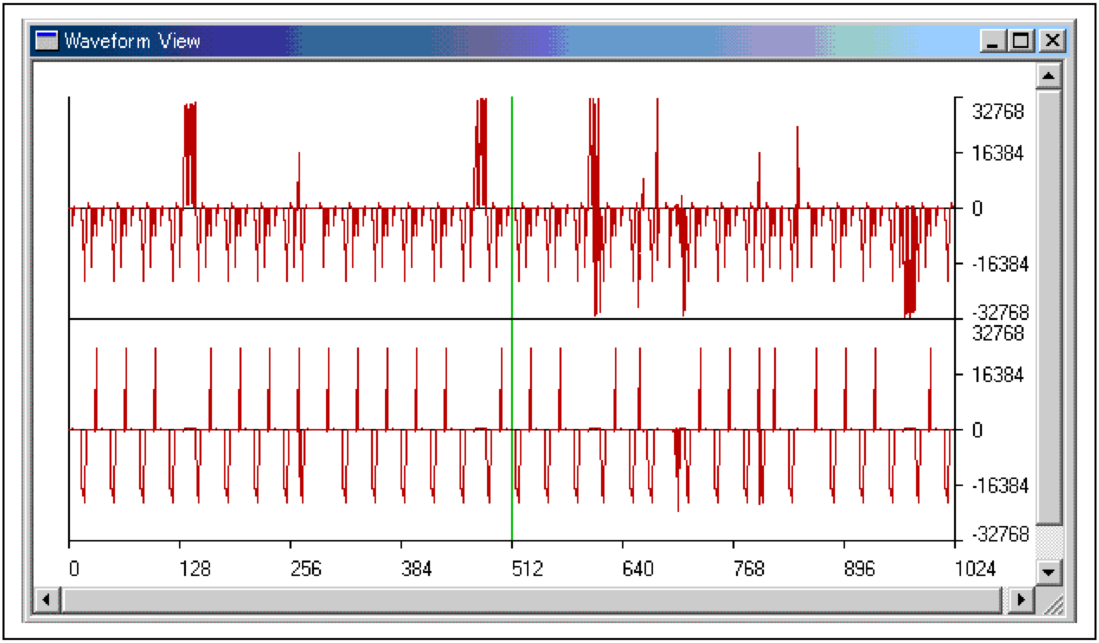


Figure 5.75 Waveform Window

Displays memory view in waveform. The X axis shows the number of sampling data and the Y axis shows the sampling value.

Clicking the right mouse button on the **Waveform View** window displays the popup menus. The menus include the following options.

5.54.1 Auto Refresh

Updates the display when instruction execution has stopped.

5.54.2 Refresh Now

Updates the display.

5.54.3 Zoom In

The horizontal axis is enlarged and displayed.

5.54.4 Zoom Out

The horizontal axis is reduced and displayed.

5.54.5 Reset Zoom

The size is returned to its original size.

5.54.6 Zoom Magnification

The zoom magnification can be selected from 2, 4, or 8 in the submenu.

5.54.7 Scale

The size of the X axis can be selected from 128, 256, or 512 Pixel from the submenu.

5.54.8 Clear Cursor

The cursor display is cleared.

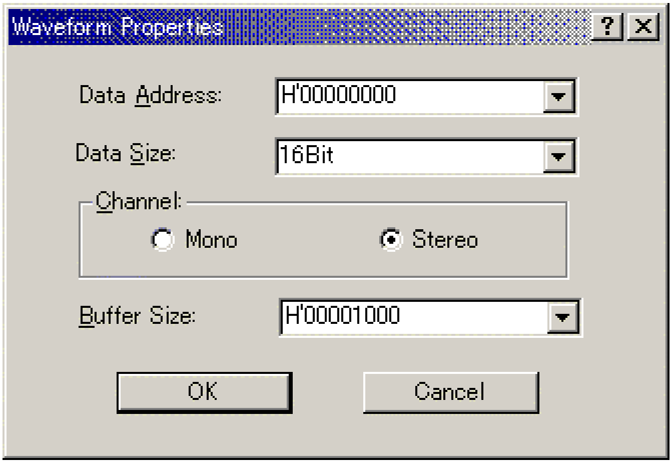
5.54.9 Sample Information…

Displays the **Sample Information** dialog box. The sampling information is displayed.

5.54.10 Property…

Displays the **Waveform Property** dialog box. The waveform data format can be specified.

5.55 Waveform Properties Dialog Box


Waveform Properties

Data Address: H'00000000

Data Size: 16Bit

Channel: ○ Mono ● Stereo

Buffer Size: H'00001000

Figure 5.76 Waveform Properties Dialog Box

Specifies the waveform format. The following items can be specified.

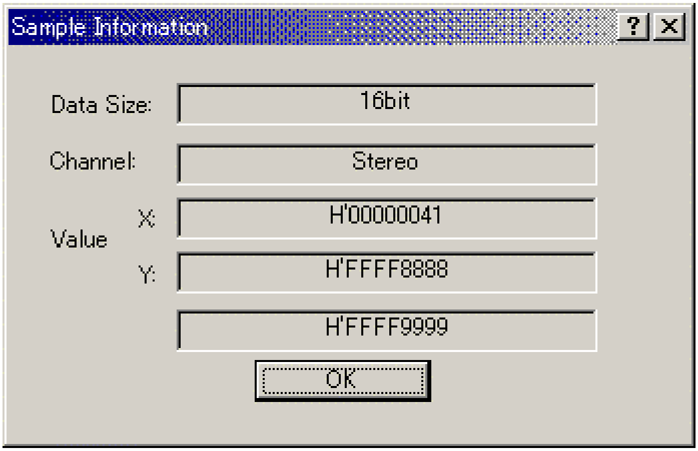
[Data Address] Specifies the start address of data in memory. (Displayed in hexadecimal)

[Data Size] Selects 8Bit or 16Bit.

[Channel] Specifies Mono or Stereo.

[Buffer Size] Specifies the buffer size of data. (Displayed in hexadecimal)

5.56 Sample Information Dialog Box



Sample Information

Data Size: 16bit

Channel: Stereo

Value X: H'00000041

Y: H'FFFF8888

H'FFFF9999

Figure 5.77 Sample Information Dialog Box

Displays the sampling information of the cursor location in the Waveform View window. The following information is displayed.

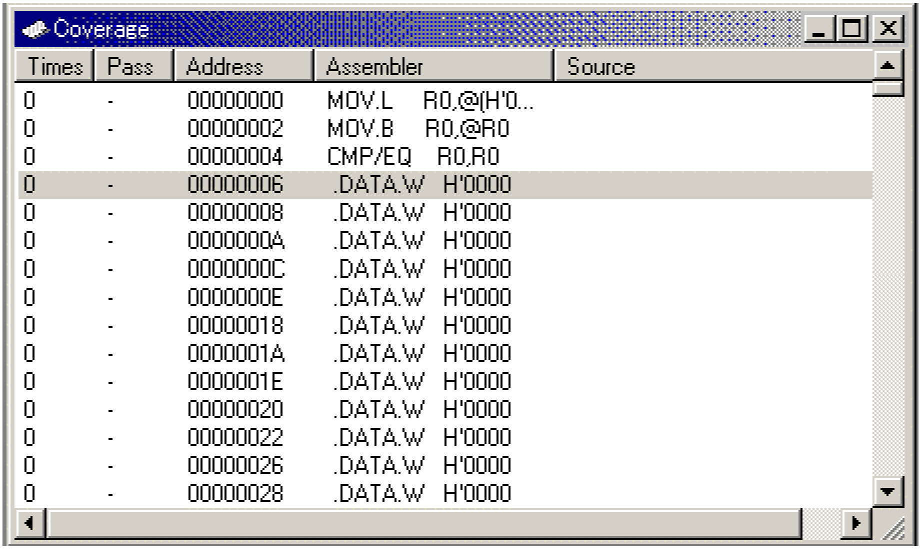
[Data Size] Displays 8bit or 16bit.

[Channel] Displays the data channel.

[Value] [X] Displays the X axis of cursor location.

[Y] Displays the Y axis of cursor location (displays Y axes for both the upper and lower plots when Stereo is selected).

5.57 Coverage Window



Times	Pass	Address	Assembler	Source
0	-	00000000	MOV.L R0,@(H'0...	
0	-	00000002	MOV.B R0,@R0	
0	-	00000004	CMP/EQ R0,R0	
0	-	00000006	.DATA.W H'0000	
0	-	00000008	.DATA.W H'0000	
0	-	0000000A	.DATA.W H'0000	
0	-	0000000C	.DATA.W H'0000	
0	-	0000000E	.DATA.W H'0000	
0	-	00000018	.DATA.W H'0000	
0	-	0000001A	.DATA.W H'0000	
0	-	0000001E	.DATA.W H'0000	
0	-	00000020	.DATA.W H'0000	
0	-	00000022	.DATA.W H'0000	
0	-	00000026	.DATA.W H'0000	
0	-	00000028	.DATA.W H'0000	

Figure 5.78 Coverage Window

Displays the instruction execution information in C/C++ and assembly-language level. Note that the conditions to acquire instruction execution information can be set through the **Open Coverage** dialog box or **Coverage Range** dialog box. When the **Coverage** window is closed, the settings of the acquired instruction execution information and the conditions to acquire information will be cleared.

The items to be displayed are as follows:

[Times] Number of times instruction was executed

[Pass] Conditions to execute condition branch instructions
T: Branches because the condition is satisfied
F: Does not branch because the condition is not satisfied

[Address] Instruction Address

[Assembler] Disassembled and then displayed.

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

Clicking the right mouse button on the **Coverage** window displays the popup menus. The menus include the following options.

5.57.1 View Source

Displays the **Source** window corresponding to the cursor location on the **Coverage** window.

5.57.2 Go to Address…

Modifies the address displayed in the **Coverage** window.

5.57.3 Set Range…

Launches the **Coverage Range** dialog box, allowing the user to specify the conditions to acquire instruction execution information.

5.57.4 Enable Coverage

Sets whether to enable or disable the acquisition of instruction execution information.

5.57.5 Clear Data...

Clears the acquired instruction execution information.

5.57.6 Save Data...

Launches the **Save Data** dialog box, allowing the user to save the coverage information in a file.

5.57.7 Load Data...

Launches the **Load Data** dialog box, allowing the user to load the coverage information from a file.

5.57.8 Refresh

Displays the latest instruction execution information.

5.57.9 Lock Refresh

Lock or unlock the refresh of coverage information. Note that when [Lock Refresh] is [On], window updates only the Times and Pass column after program execution.

5.58 Open Coverage Dialog Box

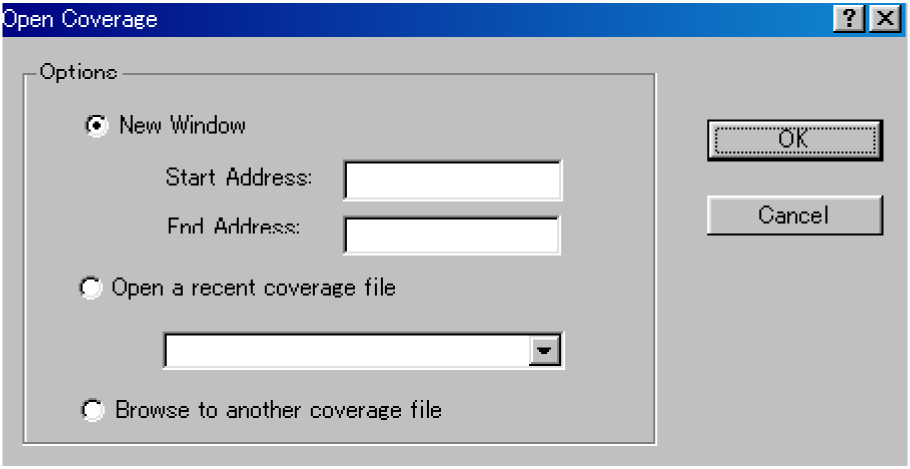


Figure 5.79 Open Coverage Dialog Box

Clicking the coverage icon opens the **Open Coverage** dialog box.

[New Window], **[Open a recent coverage file]**, or **[Browse to another coverage file]** can be selected. When **[New Window]** is selected, the start and end addresses of the coverage information range to be displayed must be specified as follows:

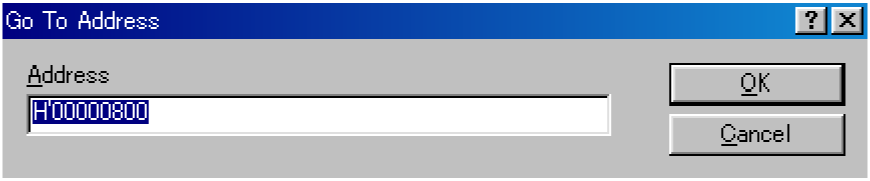
[Start Address] Start address of coverage information display
(When a prefix is omitted, the values is input in hexadecimal.)

[End Address] End address of coverage information display
(When a prefix is omitted, the value is input in hexadecimal.)

When **[Open a recent coverage file]** is selected, up to four recent files that have been saved are displayed.

When **[Browse to another coverage file]** is selected, a file open dialog box will appear to prompt the user to select a coverage information file.

5.59 Go To Address Dialog Box



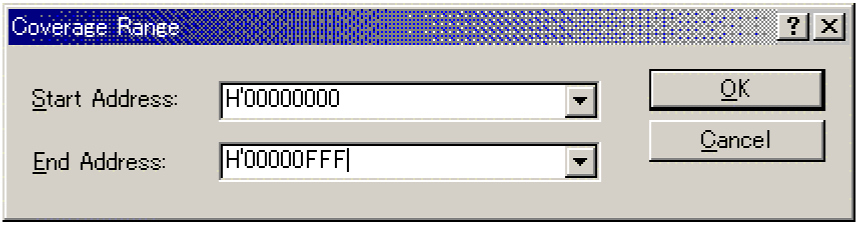
Go To Address

Address: H'00000800

Figure 5.80 Go To Address Dialog Box

Modifies the address displayed in the **Coverage** window.

5.60 Coverage Range Dialog Box



Coverage Range

Start Address: H'00000000

End Address: H'00000FFF

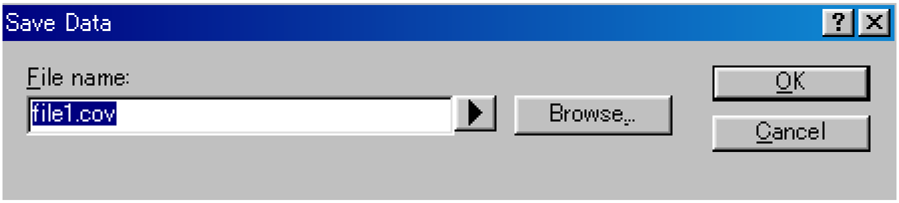
Figure 5.81 Coverage Range Dialog Box

Specifies the condition to acquire instruction execution information. The following items can be specified.

[Start Address] Start address (When a prefix is omitted, the value is input in hexadecimal.)

[End Address] End address (When a prefix is omitted, the value is input in hexadecimal.)

5.61 Save Data Dialog Box



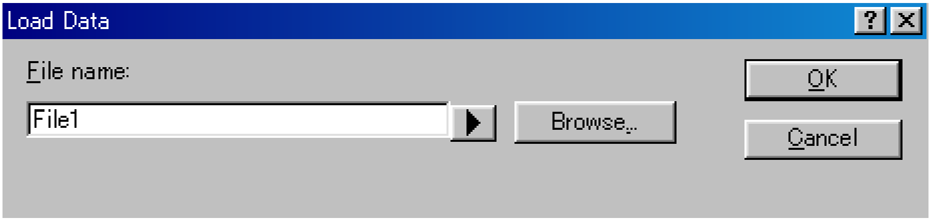
File name: file1.cov

Figure 5.82 Save Data Dialog Box

Specifies the location and name of a coverage information file to be saved. The placeholder or the browse button can be used.

If a file name extension is omitted, .COV is automatically added. If a file name extension other than .COV or .TXT is specified, an error message will be displayed.

5.62 Load Data Dialog Box



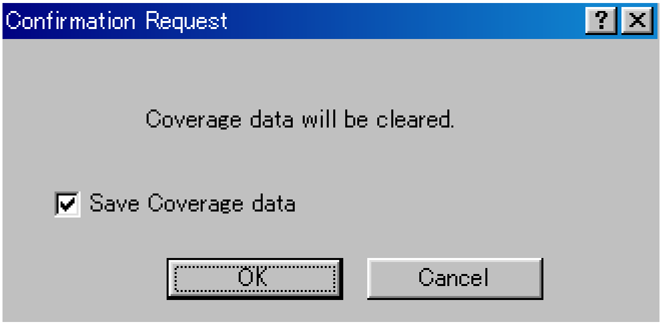
File name: File1

Figure 5.83 Load Data Dialog Box

Specifies the location and name of a coverage information file to be loaded. The placeholder or the browse button can be used.

Only .COV files can be loaded. If a file name extension other than .COV is specified, an error message will be displayed.

5.63 Confirmation Request Dialog Box



Confirmation Request

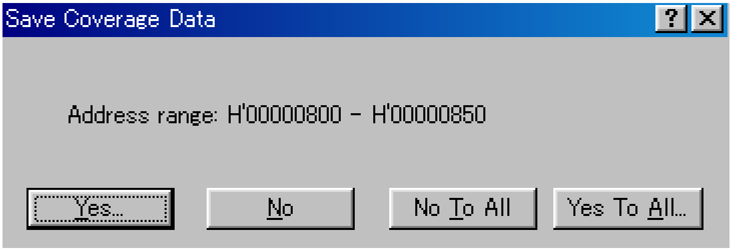
Coverage data will be cleared.

☑ Save Coverage data

Figure 5.84 Confirmation Request Dialog Box

When **[Clear Data]** or **[Set Range...]** is clicked or when an attempt is made to close coverage window, the **Confirmation Request** dialog box will appear.

5.64 Save Coverage Data Dialog Box



Save Coverage Data

Address range: H'00000800 - H'00000850

Figure 5.85 Save Coverage Data Dialog Box

When **[Save Session]** is selected from the **[File]** menu, the **Save Coverage Data** dialog box will appear, allowing the user to save the coverage window data in separate files or a single file.

When multiple coverage windows are open, the **Save Coverage Data** dialog box will appear for each open coverage window.

Clicking the **[Not To All]** button close the dialog box without saves the data.

Clicking the **[Yes To All]** button saves the data of all coverage windows in a single file.

Section 6 Command Lines

Table 6.1 lists the commands.

Table 6.1 Simulator/Debugger Commands

No.	Command Name	Abbreviation	Function
1	!	-	Comment
2	ANALYSIS	AN	Enables or disables performance analysis
3	ANALYSIS_RANGE	AR	Sets or displays performance analysis functions
4	ANALYSIS_RANGE_ DELETE	AD	Deletes a performance analysis range
5	ASSEMBLE	AS	Assembles instructions into memory
6	ASSERT	-	Checks if an expression is true or false
7	BREAKPOINT	BP	Sets a breakpoint at an instruction address
8	BREAK_ACCESS	BA	Specifies a memory range access as a break condition
9	BREAK_CLEAR	BC	Deletes breakpoints
10	BREAK_CYCLE	BCY	Specifies a cycle as a break condition
11	BREAK_DATA	BD	Specifies a memory data value as a break condition
12	BREAK_DISPLAY	BI	Displays a list of breakpoints
13	BREAK_ENABLE	BE	Enables or disables a breakpoint
14	BREAK_REGISTER	BR	Specifies a register data as a break condition
15	BREAK_SEQUENCE	BS	Sets sequential breakpoints
16	CHANGE_CONFIGURATION	CC	Sets the current configuration
17	CHANGE_PROJECT	CP	Sets the current project
18	CLOCK_RATE	CKR	Sets a clock rate
19	COVERAGE	CV	Enables or disables coverage measurement
20	COVERAGE_DISPLAY	CVD	Displays coverage information
21	COVERAGE_LOAD	CVL	Loads coverage information
22	COVERAGE_RANGE	CVR	Sets a coverage range
23	COVERAGE_SAVE	CVS	Saves coverage information
24	DEFAULT_OBJECT_FORMAT	DO	Sets the default object (program) format
25	DISASSEMBLE	DA	Disassembles memory contents
26	ERASE	ER	Clears the **Command Line** window
27	EVALUATE	EV	Evaluates an expression

Table 6.1 Simulator/Debugger Commands (cont)

No.	Command Name	Abbreviation	Function
28	FILE_LOAD	FL	Loads an object (program) file
29	FILE_SAVE	FS	Saves memory to a file
30	FILE_VERIFY	FV	Verifies file contents against memory
31	GO	GO	Executes user program
32	GO_RESET	GR	Executes user program from reset vector
33	GO_TILL	GT	Executes user program until temporary breakpoint
34	HALT	HA	Halts the user program
35	INITIALIZE	IN	Initializes the debugging platform
36	LOG	LO	Controls command output logging
37	MAP_DISPLAY	MA	Displays memory mapping
38	MAP_SET	MS	Allocates a memory area
39	MEMORY_DISPLAY	MD	Displays memory contents
40	MEMORY_EDIT	ME	Modifies memory contents
41	MEMORY_FILL	MF	Fills a memory area
42	MEMORY_MOVE	MV	Moves a block of memory
43	MEMORY_TEST	MT	Tests a block of memory
44	OPEN_WORKSPACE	OW	Opens a workspace
45	PROFILE	PR	Enables or disables profile
46	PROFILE_DISPLAY	PD	Displays profile information
47	PROFILE_SAVE	PS	Saves the profile information to file
48	QUIT	QU	Exits HEW
49	RADIX	RA	Sets default input radix
50	REGISTER_DISPLAY	RD	Displays CPU register values
51	REGISTER_SET	RS	Changes CPU register contents
52	RESET	RE	Resets CPU
53	RESPONSE	RS	Sets a window refresh area
54	SLEEP	-	Delays command execution
55	STEP	ST	Steps program (by instructions or source lines)
56	STEP_MODE	SM	Selects the step mode
57	STEP_OUT	SP	Steps out of the current function
58	STEP_OVER	SO	Steps program, not stepping into functions
59	STEP_RATE	SR	Sets or displays rate of stepping
60	STEP_UNIT	SN	Sets the unit of execution

Table 6.1 Simulator/Debugger Commands (cont)

No.	Command Name	Abbreviation	Function
61	SUBMIT	SU	Executes a command file
62	SYMBOL_ADD	SA	Defines a symbol
63	SYMBOL_CLEAR	SC	Deletes a symbol
64	SYMBOL_LOAD	SL	Loads a symbol information file
65	SYMBOL_SAVE	SS	Saves a symbol information file
66	SYMBOL_VIEW	SV	Displays symbols
67	TCL	-	Enables or disables the TCL
68	TRACE	TR	Displays trace information
69	TRACE_ACQUISITION	TA	Enables or disables trace information acquisition
70	TRACE_SAVE	TV	Outputs trace information into a file
71	TRACE_STATISTIC	TST	Analyzes statistic information

The following describes the syntax of each command.

6.1 !(COMMENT)

Abbreviation: None

Description:

Allows a comment to be entered, useful for documenting log files.

Syntax:

! <text>

Parameter	Type	Description
<text>	Text	Output text

Example:

! Start of test routine	Outputs comment 'Start of test routine' into the **Command Line** window (and to the log file, if logging is active).

6.2 ANALYSIS

Abbreviation: AN

Description:

Enables/disables performance analysis. Counts are not automatically reset before running.

Syntax:

an [<state>]

Parameter	Type	Description
None		Displays the performance analysis state
<state>	Keyword	Enables or disables performance analysis
	Enable	Enables performance analysis
	Disable	Disables performance analysis
	Reset	Resets performance analysis counts

Examples:

ANALYSIS	Displays performance analysis state.
AN enable	Enables performance analysis.
AN disable	Disables performance analysis.
AN reset	Resets performance analysis counts.

6.3 ANALYSIS_RANGE

Abbreviation: AR

Description:

Sets a function for which the performance analysis is provided, or displays a function for which the performance analysis is provided without parameters.

Syntax:

ar [<function name>]

Parameter	Type	Description
None		Displays all functions for which the performance analysis is provided
<function name>	Character string	Name of function for which the performance analysis is provided

Examples:

ANALYSIS_RANGE sort	Provides the performance analysis for the function sort.
AR	Displays the function for which the performance analysis is provided.

6.4 ANALYSIS_RANGE_DELETE

Abbreviation: AD

Description:

Deletes the specified function, or all functions if no parameters are specified (it does **not** ask for confirmation).

Syntax:

ad [<index>]

Parameter	Type	Description
None		Deletes all functions
<index>	Numeric	Index number of function to delete

Examples:

ANALYSIS_RANGE_DELETE 6	Deletes the function with index number 6.
AD	Deletes all functions.

6.5 ASSEMBLE

Abbreviation: AS

Description:

Assembles mnemonics and writes them into memory. In assembly mode, '.' exits, '^' steps back a byte, the ENTER key steps forward a byte.

Syntax:

as <address>

Parameter	Type	Description
<address>	Numeric	Address at which to start assembling

Example:

AS H’1000	Starts assembling from H’1000.

6.6 ASSERT

Abbreviation: None

Description:

Checks if an expression is true or false. It can be used to terminate the batch file when the expression is false. If the expression is false, an error is returned. This command can be used to write test harnesses for subroutines.

Syntax:

assert <expression>

Parameter	Type	Description
<expression>	Expression	Expression to be checked

Example:

ASSERT #R0 == 0x100	Returns an error if R0 does not contain 0x100.

6.7 BREAKPOINT

Abbreviation: BP

Description:

Specifies a breakpoint at the address where the instruction is written.

Syntax:

bp <address> [<count>] [<Action>]

Parameter	Type	Description
<address>	Numeric	The address of a breakpoint
<count>	Numeric	The number of times the instruction at the specified address is to be fetched (1 to 16383, default = 1).
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs (saves) data to a file
	Output (O)	Outputs (reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved

<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> <interrupt type2> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt SH-1, SH-2, SH-DSP series Interrupt vector number (0 to FF) SH-3, SH-4, SH3-DSP series INTEVT (0 to H’FFF)
<interrupt type2>	Numeric	Value of INTEVT2 (0 to H’FFF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Note: <interrupt type2> can only be specified for the SH3-DSP series.

Examples:

BREAKPOINT 0 2	A break occurs when an attempt is made to execute the instruction at address H'0 for the second time.
BP C0 Input in.dat 100 2 8	Eight two-byte data fields are written from file “in.dat” to H’100 when an attempt is made to execute the instruction at address H'C0.

6.8 BREAK_ACCESS

Abbreviation: BA

Description:

Specifies a memory range as a break condition

Syntax:

ba <start address> [<end address>] [<mode>] [<Action>]

Parameter	Type	Description
<start address>	Numeric	The start address of a breakpoint
<end address>	Numeric	The end address of a breakpoint (optional, default = <start address>)
<mode>	Keyword	Access type (optional, default = RW).
	R	A break occurs when the specified range is read.
	W	A break occurs when the specified range is written to.
	RW	A break occurs when the specified range is read or written to.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs (saves) data to a file
	Output (O)	Outputs (reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> <interrupt type2> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt • SH-1, SH-2, SH2-DSP series interrupt vector number (0 to H’FF) • SH-3, SH-4, SH3-DSP series INTEVT (0 to H’FFF)
<interrupt type2>	Numeric	Value of INTEVT2 (0 to H’FFF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Note: <interrupt type2> can only be specified for the SH3-DSP series.

Examples:

BREAK_ACCESS 0 1000 W	A break occurs when the specified range from address H'0 to address H'1000 is written to.
BA FFFF	A break occurs when address H'FFFF is accessed.

Note: For the SH3-DSP series, specify values within the range H'A5000000 to H'A501FFFF (X and Y memory virtual addresses, corresponding to physical addresses H'05000000 to H'0501FFFF) as the start end addresses for X or Y memory accesses by the MOVX or MOVY instruction.

6.9 BREAK_CLEAR

Abbreviation: BC

Description:

Deletes breakpoints.

Syntax:

bc <index>

Parameter	Type	Description
<index>	Numeric	Index of the breakpoint to be canceled. If the index is omitted, all breakpoints are deleted.

Examples:

BREAK_CLEAR 0	The first breakpoint is deleted.
BC	All breakpoints are deleted.

6.10 BREAK_CYCLE

Abbreviation: BCY

Description:

Specifies the number of cycles as a break condition.

Syntax:

by <cycle> [<count>] [<Action>]

Parameter	Type	Description
<cycle>	Numeric	The condition matching the number of cycles <cycle>×n.
<count>	Keyword	The condition satisfying the number of times. (optional, default =ALL)
	All	Break condition is satisfied every time the condition is matched.
	Numeric	1 to H'FFFF Break condition is satisfied only when it matches the specified number of times.

<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs(saves) data to a file
	Output (O)	Outputs(reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	**Type**	**Description**
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	**Type**	**Description**
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> <interrupt type2> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt • SH-1, SH-2, SH2-DSP series interrupt vector number (0 to H'FF) • SH-3, SH-4, SH3-DSP series INTEVT (0 to H'FFF)
<interrupt type2>	Numeric	Value of INTEVT2 (0 to H'FFF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Note: <interrupt type2> can only be specified for the SH3-DSP series.

Examples:

BREAK_CYCLE 1000 20	Specifies breaks to occur H'20 times in every H'1000 cycles.
BCY 5000	Specifies a break to occur in every H'5000 cycles.

6.11 BREAK_DATA

Abbreviation: BD

Description:

Specifies a memory data value as a break condition.

Syntax:

bd <address> <data> [<size>] [<option>] [<Action>]

Parameter	Type	Description
<address>	Numeric	The address where the break condition is checked.
<data>	Numeric	Access data
<size>	Keyword	Size (optional, default = L).
	byte	Byte size
	word	Word size
	longword	Longword size
	single	Single-precision floating-point size
	double	Double-precision floating-point size

<option>	Keyword	Match or mismatch of data. The default is EQ.
	EQ	A break occurs when the data matches the specified value.
	NE	A break occurs when the data does not match the specified value.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs(saves) data to a file
	Output (O)	Outputs(reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> <interrupt type2> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt • SH-1, SH-2, SH2-DSP series interrupt vector number (0 to FF) • SH-3, SH-4, SH3-DSP series INTEVT (0 to H’FFF)
<interrupt type2>	Numeric	Value of INTEVT2 (0 to H’FFF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Note: <interrupt type2> can only be specified for the SH3-DSP series.

Examples:

BREAK_DATA 0 100 L EQ	A break occurs when H'100 is written to memory address H'0 in longword.
BD C0 FF B NE	A break occurs when a value other than H'FF is written to memory address H'C0 in byte.
BD 4000 1000	A break occurs when H'1000 is written to memory address H'4000 in longword.

Note: For the SH3-DSP series, specify values within the range H'A5000000 to H'A501FFFF (X and Y memory virtual addresses, corresponding to physical addresses H'05000000 to H'0501FFFF) as the start end addresses for X or Y memory accesses by the MOVX or MOVY instruction.

6.12 BREAK_DISPLAY

Abbreviation: BI

Description:

Displays a list of breakpoints.

Syntax:

bi

Parameter	Type	Description
None		Displays a list of breakpoints

Examples:

BREAK_DISPLAY	A list of breakpoints is displayed.
BI	A list of breakpoints is displayed.

6.13 BREAK_ENABLE

Abbreviation: BE

Description:

Enables or disables a breakpoint.

Syntax:

be <flag> [<index>]

Parameter	Type	Description
<flag>	Keyword	Enables or disables a breakpoint
	E	Enable
	D	Disable
<index>	Numeric	Index of the breakpoint to be canceled. If the index is omitted, all breakpoints are deleted.

Examples:

BREAK_ENABLE D 0	The first breakpoint is disabled.
BE E	All breakpoints are enabled.

6.14 BREAK_REGISTER

Abbreviation: BR

Description:

Specifies a register data as a break condition

Syntax:

br <register name> [<data> <size>] [<option>] [<Action>]

Parameter	Type	Description
<register>	Character string	Register name.
<data>	Numeric	Access data.
<size>	Keyword	Access size. If no size is specified, the size of the specified register is assumed. Note that when data is specified, the size must not be omitted.
	byte	Byte size
	word	Word size
	longword	Longword size
	single	Single-precision floating-point size
	double	Double-precision floating-point size
<option>	Keyword	Match or mismatch of data. The default is EQ.
	EQ	A break occurs when the data matches the specified value.
	NE	A break occurs when the data does not match the specified value.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs(saves) data to a file
	Output (O)	Outputs(reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> <interrupt type2> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt • SH-1, SH-2, SH2-DSP series interrupt vector number (0 to H’FF) • SH-3, SH-4, SH3-DSP series INTEVT (0 to H’FFF)
<interrupt type2>	Numeric	Value of INTEVT2 (0 to H’FFF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Note: <interrupt type2> can only be specified for the SH3-DSP series.

Examples:

BREAK_REGISTER R0 FFFF W EQ	A break occurs when the lower two bytes of the R0 register change to H'FFFF.
BR R10	A break occurs when the R10 register is written to.

6.15 BREAK_SEQUENCE

Abbreviation: BS

Description:

Sets sequential breakpoints

Syntax:

bs <address1> [<address2> [<address 3> [...]]] [<Action>]

Parameter	Type	Description
<address1> - <address8>	Numeric	Addresses of sequential breakpoints. Up to eight addresses can be specified.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs(saves) data to a file
	Output (O)	Outputs(reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H’01 to H’FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> <interrupt type2> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt • SH-1, SH-2, SH-DSP series interrupt vector number (0 to H’FF) • SH-3, SH-4, SH3-DSP series INTEVT (0 to H’FFF)
<interrupt type2>	Numeric	Value of INTEVT2 (0 to H’FFF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Note: <interrupt type2> can only be specified for the SH3-DSP series.

Examples:

BREAK_SEQUENCE 1000 2000	A break occurs when addresses H'1000 and H'2000 are passed in this order.
BS 1000	A break occurs when address H'1000 is executed.

6.16 CHANGE_CONFIGURATION

Abbreviation: CC

Description:

Sets the current configuration.

Syntax:

cc <config name>

Parameter	Type	Description
<config name>	Character string	Configuration name

Example:

CC Debug	Sets the current configuration to Debug.

6.17 CHANGE_PROJECT

Abbreviation: CP

Description:

Sets the current project.

Syntax:

cp <project name>

Parameter	Type	Description
<project name>	Character string	Project name

Example:

CP PROJ2	Sets the current project to PROJ2.

6.18 CLOCK_RATE

Abbreviation: CKR

Description:

Sets the internal/external clock rate. This command can only be used in the SH-4 series.

Syntax:

ckr [<rate>]

Parameter	Type	Description
None		Displays the clock rate.
<rate>	Numeric	When the external clock rate is 1, the internal clock rate is 1 to 16. (default = 1). However, default is 3 only for SH-4 with BSC.

Example:

CLOCK RATE 6	Sets the clock rate to 1:6.

6.19 COVERAGE

Abbreviation: CV

Description:

Enables or disables the coverage range measurement or resets coverage information.

Syntax:

cv [<state>]

Parameter	Type	Description
none		Displays coverage state.
<state>	enable	Enables coverage measurement.
	disable	Disables coverage measurement.
	reset	Resets result of coverage measurement.

Examples:

COVERAGE	Displays coverage state.
CV enable	Enables coverage measurement.
CV r	Resets result of coverage measurement.

6.20 COVERAGE_DISPLAY

Abbreviation: CVD

Description:

Displays coverage information.

Syntax:

cvd

Parameter	Type	Description
none		Displays coverage information.

Example:

COVERAGE_DISPLAY	Displays coverage information.

6.21 COVERAGE_LOAD

Abbreviation: CVL

Description:

Loads the coverage information from a .COV file.
If a wrong file format is specified or the specified file is not found, a warning message will be displayed.

Syntax:

cvl <filename>

Parameter	Type	Description
filename	Character string	File name

Examples:

COVERAGE_LOAD TEST	Loads the coverage information from the TEST.COV file.
CLV COVERAGE.COV	Loads the coverage information from the COVERAGE.COV file.

6.22 COVERAGE_RANGE

Abbreviation: CVR

Description:

Sets the coverage range or displays the range of coverage measurement without parameters.

Syntax:

cvr [<start> <end>]

Parameter	Type	Description
none		Displays the coverage measurement range.
<start>	Numeric	Start address of the coverage measurement range.
<end>	Numeric	End address of the coverage measurement range.

Examples:

COVERAGE_RANGE H’1000 H’10FF	Measures the coverage of addresses between H’1000 and H’10FF.
CVR	Displays the range of coverage measurement.

6.23 COVERAGE_SAVE

Abbreviation: CVS

Description:

Saves the coverage information in a .COV or .TXT file.
If a wrong file extension is specified, an error message will be displayed.

Syntax:

cvs <filename>

Parameter	Type	Description
filename	Character string	File name

Examples:

COVERAGE_SAVE TEST	Saves the coverage information in the TEST.COV file.
CVS COVERAGE.COV	Saves the coverage information in the COVERAGE.COV file.

6.24 DEFAULT_OBJECT_FORMAT

Abbreviation: DO

Description:

Sets the default format for loading object (program) files. The format set with this command is only valid when the format specification is omitted from the FILE_LOAD command.

Syntax:

do [<format>]

Parameter	Type	Description
none		Displays the default format settings.
<format>	Keyword	Object format
	Binary	Binary type
	Elf/Dwarf2	Elf/Dwarf2 type
	IntelHex	Intel-Hex type
	S-Record	S type

Example:

DEFAULT_OBJECT_FORMAT	Displays the default format settings.
DO binary	Sets the default format to binary .

6.25 DISASSEMBLE

Abbreviation: DA

Description:

Disassembles memory contents to assembly-language code. The display of disassembled memory is fully symbolic.

Syntax:

da <address> [<length>]

Parameter	Type	Description
<address>	Numeric	Start address
<length>	Numeric	Number of instructions (optional, default = 16)

Examples:

DISASSEMBLE H’100 5	Disassembles 5 lines of code starting at H’100.
DA H’3E00 20	Disassembles 20 lines of code starting at H’3E00.

6.26 ERASE

Abbreviation: ER

Description:

Clears the **Command Line** window

Syntax:

er

Parameter	Type	Description
none		Clears the **Command Line** window

Example:

ER	Clears the **Command Line** window.

6.27 EVALUATE

Abbreviation: EV

Description:

Provides a calculator function, evaluating simple and complex expressions, with parentheses, mixed radices, and symbols. All operators have the same priority but parentheses may be used to change the order of evaluation. The operators have the same meaning as in C/C++. Expressions can also be used in any command where a number is required. Register names may be used, but must always be prefixed by the ‘#’ character. The result is displayed in hexadecimal, decimal, octal, or binary.

Syntax:

ev <expression>

Parameter	Type	Description
<expression>	Expression	Expression to be evaluated

Valid operators:

&&	logical AND	||	logical OR	<<	left arithmetic shift	>>	right arithmetic shift
+	addition	-	subtraction	*	multiplication	/	division
%	modulo	|	bitwise OR	&	bitwise AND	~	bitwise NOT
^	bitwise exclusive OR	!	logical NOT	==	equal to	!=	unequal to
>	greater than	<	less than	>=	greater than or equal to	<=	less than or equal to

Examples:

EV H’123 + (D’73 | B’10)	Result: H’16E D’366 O’556 B’00000000000000000000000101101110
EV #R1 * #R2	Result: H’121 D’289 O’441 B’00000000000000000000000100100001

6.28 FILE_LOAD

Abbreviation: FL

Description:

Loads an object code file to memory with the specified offset. Existing symbols are cleared, and the new ones are defined. If an offset is specified this will be added to the symbols. The file extension default is **.MOT**.

Syntax:

fl [<format>] <filename> [<offset>] [<state>]

Parameter	Type	Description
<format>	Keyword	Object format (optional, default = DEFAULT_OBJECT_FORMAT settings)
	Binary	Binary type
	Elf/Dwarf2	Elf/Dwarf2 type
	IntelHex	Intel-Hex type
	S-Record	S type
<filename>	Character string	File name
<offset>	Numeric	Offset to be added to load address (optional, default = 0)
<state>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

Examples:

FILE_LOAD A:\\BINARY\\TESTFILE.A22	Loads Motorola S-Record file "testfile.a22".
FL ANOTHER.MOT H’200	Loads Motorola S-Record file "another.mot" with an offset of H’200 bytes.

6.29 FILE_SAVE

Abbreviation: FS

Description:

Saves the specified memory area data to a file. The user is warned if about to overwrite an existing file. The file extension default is **.MOT**. Symbols are **not** automatically saved.

Syntax:

fs [<format>] <filename> <start> <end>

Parameter	Type	Description
<format>	Keyword	Object format (optional, default = DEFAULT_OBJECT_FORMAT settings)
	Binary	Binary type
	IntelHex	Intel-Hex type
	S-Record	S type
<filename>	Character string	File name
<start>	Numeric	Start address
<end>	Numeric	End address

Examples:

FILE_SAVE TESTFILE 0 H’2013	Saves address range 0-H’2013 as Motorola S-Record file "TESTFILE.MOT".
FS D:\\USER\\ANOTHER.A22 H’4000 H’4FFF	Saves address range H’4000-H’4FFF as Motorola S-Record format file "ANOTHER.A22".

6.30 FILE_VERIFY

Abbreviation: FV

Description:

Verifies file contents against memory contents. The file data must be in a Motorola S-Record format. The file extension default is **.MOT**.

Syntax:

fv <filename> [<offset>]

Parameter	Type	Description
<filename>	Character string	File name
<offset>	Numeric	Offset to be added to file address (optional, default = 0)

Examples:

FILE_VERIFY A:\\BINARY\\TEST.A22	Verifies Motorola S-Record file "TEST.A22" against memory.
FV ANOTHER 200	Verifies Motorola S-Record file "ANOTHER.MOT" against memory with an offset of H'200 bytes.

6.31 GO

Abbreviation: GO

Description:

Executes object code (the user program). While the user program is executing, the **Performance Analysis** window is updated. While the user system is halted, a PC value is displayed.

Syntax:

go [<state>] [<address>]

Parameter	Type	Description
<state>	Keyword	Specifies whether or not to continue command processing during user program execution (optional, default = wait)
	wait	Causes command processing to wait until user program stops
	continue	Continues command processing during execution
<address>	Numeric	Start address for PC (optional, default = PC value)

Wait is the default and this causes command processing to wait until user program stops executing.

Continue allows you to continue to enter commands (but they may not work depending on the debugging platform).

Examples:

GO	Executes the user program from the current PC value. Command processing cannot be continued.
GO CONTINUE H'1000	Executes the user program from H'1000. Command processing can be continued.

6.32 GO_RESET

Abbreviation: GR

Description:

Executes the user program starting at the address specified in the reset vector.

While the user program is executing, the **Performance Analysis** window is updated.

Syntax:

gr [<state>]

Parameter	Type	Description
<state>	Keyword	Specifies whether or not to continue command processing during user program execution (optional, default = wait)
	wait	Causes command processing to wait until user program stops
	continue	Continues command processing during execution

Wait is the default and this causes command processing to wait until user program stops executing.

Continue allows you to continue to enter commands (but they may not work depending on the debugging platform)

Example:

GR	Executes the user program starting at the address specified in the reset vector (does not continue command processing).

6.33 GO_TILL

Abbreviation: GT

Description:

Executes the user program from the current PC with temporary breakpoints. This command takes multiple addresses as parameters, and these are used to set temporary PC breakpoints (these breakpoints only exist for the duration of the command).

Syntax:

gt [<state>] <address>...

Parameter	Type	Description
<state>	Keyword	Specifies whether or not to continue command processing during user program execution (optional, default = wait)
	wait	Causes command processing to wait until user program stops
	continue	Continues command processing during execution
<address>...	Numeric	Temporary breakpoint address (list)

Wait is the default and this causes command processing to wait until user program stops executing

Continue allows you to continue to enter commands (but they may not work depending on the debugging platform)

Example:

GO_TILL H’1000	Continues execution until the PC reaches address H’1000.

6.34 HALT

Abbreviation: HA

Description:

Halts the user program. This command can be used after the GO command if the GO command uses continue for option.

Syntax:

ha

Parameter	Type	Description
none		Halts the user program

Example:

HA	Halts the user program.

6.35 INITIALIZE

Abbreviation: IN

Description:

Initializes all breakpoints and memory mapping. It also initializes debugging platform, as if you had reselected the target DLL.

Syntax:

in

Parameter	Type	Description
none		Initializes debugging platform.

Example:

IN	Initializes debugging platform.

6.36 LOG

Abbreviation: LO

Description:

Controls logging of command output to file. If no parameters are specified, logging status is displayed. If an existing file is specified, you will be warned; if you answer 'No', data will be overwritten to the existing file, otherwise the file will be added. Logging is only supported for the command line interface.

Syntax:

lo [<state>|<filename>]

Parameter	Type	Description
none		Displays logging status
<state>	Keyword	Starts or suspends logging
	+	Starts logging
	-	Suspends logging
<filename>	Numeric	Specifies the logging output file

Examples:

LOG TEST	Stores the logging in file TEST.
LO -	Suspends logging.
LOG +	Resumes logging.
LOG	Displays logging status

6.37 MAP_DISPLAY

Abbreviation: MA

Description:

Displays memory mapping.

Syntax:

ma

Parameter	Type	Description
none		Displays the current memory mapping

Example:

MA	Displays the current memory mapping.

6.38 MAP_SET

Abbreviation: MS

Description:

Allocates a memory area.

Syntax:

ms <start address> [<end address>] [<mode>]

Parameter	Type	Description
<start address>	Numeric	Specified start address
<end address>	Numeric	Specified end address (optional, default = start address)
<mode>	Keyword	Access type (optional, default = RW)
	R	Read only
	W	Write only
	RW	Displays the current memory mapping

Examples:

MAP_SET 0000 3FFF RW	A read/write-enabled area is allocated to addresses H'0000 to H'3FFF.
MS 5000	A read/write-enabled area is allocated to address H'5000.

6.39 MEMORY_DISPLAY

Abbreviation: MD

Description:

Displays memory contents.

Syntax:

md <address> [<length>] [<mode>]

Parameter	Type	Description
<address>	Numeric	Start address
<length>	Numeric	Length (optional, default = H’100 bytes)
<mode>	Keyword	Display format (optional, default = byte)
	byte	Displays in byte units
	word	Displays in word units (2 bytes)
	long	Displays in longword units (4 bytes)
	ascii	Displays in ASCII codes
	single	Displays in single-precision floating-point format
	double	Displays in double-precision floating-point format

Examples:

MEMORY_DISPLAY H’C000 H’100 WORD	Displays H’100 bytes of memory starting at H’C000 in word units
MEMORY_DISPLAY H’1000 H’FF	Displays H’FF bytes of memory starting at H’1000 in byte units

6.40 MEMORY_EDIT

Abbreviation: ME

Description:

Allows memory contents to be modified. When editing memory the current location may be modified in a similar way to that described in the **ASSEMBLE** command description.

When editing, '.' exits edit mode, '^' goes back one data unit, and blank line goes forward without modification.

Syntax:

me <address> [<mode>] [<state>]

Parameter	Type	Description
<address>	Numeric	Address to edit
<mode>	Keyword	Format (optional, default = byte)
	byte	Edits in byte units
	word	Edits in word units
	long	Edits in longword units
	ascii	Edits in ASCII codes
	single	Edits in the single-precision floating-point format
	double	Edits in the double-precision floating-point format
<state>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

Example:

ME H’1000 WORD	Modifies memory contents in word units starting from H’1000 (with verification)

6.41 MEMORY_FILL

Abbreviation: MF

Description:

Modifies the contents in the specified memory area to the specified data value.

Syntax:

mf <start> <end> <data> [<mode>] [<state>]

Parameter	Type	Description
<start>	Numeric	Start address
<end>	Numeric	End address
<data>	Numeric	Data value
<mode>	Keyword	Data size (optional, default = byte)
	byte	Byte
	word	Word
	long	Longword
	single	Single-precision floating-point
	double	Double-precision floating-point
<state>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

Examples:

MEMORY_FILL H’C000 H’C0FF H’55AA WORD	Modifies memory contents in the range from H’C000 to H’C0FF to word data H’55AA.
MF H’5000 H’7FFF H’21	Modifies memory contents in the range from H’5000 to H’7FFF to data H’21.

6.42 MEMORY_MOVE

Abbreviation: MV

Description:

Moves data in the specified memory area.

Syntax:

mv <start> <end> <dest> [<state>]

Parameter	Type	Description
<start>	Numeric	Start address
<end>	Numeric	End address (including this address)
<dest>	Numeric	Destination start address
<state>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

Examples:

MEMORY_MOVE H’1000 H’1FFF H’2000	Moves memory contents in the area from H’1000 to H’1FFF into H’2000.
MV H’FB80 H’FF7F H’3000	Moves memory contents in the area from H’FB80 to H’FF7F into H’3000.

6.43 MEMORY_TEST

Abbreviation: MT

Description:

Performs read, write, and verification testing in the specified address range. The original contents of memory have been replaced by the newly written data. The test will access the memory according to the map settings.

This simulator/debugger does not support the MEMORY_TEST command.

Syntax:

mt <start> <end>

Parameter	Type	Description
<start>	Numeric	Start address
<end>	Numeric	End address (including this address)

Examples:

MEMORY_TEST H’8000 H’BFFF	Tests from H’8000 to H’BFFF.
MT H’4000 H’5000	Tests from H’4000 to H’5000.

6.44 OPEN_WORKSPACE

Abbreviation: OW

Description:

Opens a workspace.

Syntax:

ow <filename>

Parameter	Type	Description
filename	Character string	Workspace file name

Example:

OW WKSP.HWS	Opens the WKSP.HWS file.

6.45 PROFILE

Abbreviation: PR

Description:

Enables, disables or sets the profiler display and resets the profiler information.

Syntax:

pr [<state>]

Parameter	Type	Description
None		Displays the profiler information.
<state>	Keyword	Enables, disables or sets the profiler display and resets the profiler information.
	enable	Enables the profiler.
	tree-off	Enables the profiler but does not trace function calls during profile information acquisition.
	disable	Disables the profiler.
	reset	Resets the profiler information.

Examples:

PROFILE ENABLE	Enables the profiler.
pr r	Resets the profiler information.

6.46 PROFILE_DISPLAY

Abbreviation: PD

Description:

Displays the profiler information.

Syntax:

pd [<mode>] [<state1>] [<state2>] [<count>]

Parameter	Type	Description
<mode>	Keyword	Specifies the method of displaying the profiler information. (optional, default=list)
	tree	Displays in tree format
	list	Displays in list format
<state1>	Keyword	Specifies whether or not to include child function information in the parent function cycle information. (optional, default=n)
	i	Specifies child function information to be included in the display.
	n	Specifies child function not to be included in the display.
<state2>	Keyword	Specifies whether or not to control displaying functions that are not executed (optional, default=a).
	e	Displays only executed functions.
	a	Displays all functions.
<count>	Numeric	Specifies the nesting level for calling functions to be displayed. This can be specified only when the <mode> parameter is 'tree' (optional, default = 16).

Examples:

PROFILE_DISPLAY TREE I	Specifies the profiler information to be displayed in tree format and to include child functions.
pd	Specifies the profiler information to be displayed in list format, without child function information.

6.47 PROFILE_SAVE

Abbreviation: PS

Description:

Saves the profiler information to a file. The default file extension is **.PRO**.

Syntax:

ps [<filename>]

Parameter	Type	Description
None		Saves the profiler information on all download modules to files.
<filename>	Character string	Specifies the name of the file to which profiler information is saved.

Example:

PROFILE_SAVE PR_INFO	Saves profiler information to a file named PR_INFO.PRO.

6.48 QUIT

Abbreviation: QU

Description:

Exits HEW. Closes a log file if it is open.

Syntax:

qu

Parameter	Type	Description
None		Exits HEW

Example:

QU	Exits HEW.

6.49 RADIX

Abbreviation: RA

Description:

Sets default input radix. If no parameters are specified, the current radix is displayed. Radix can be changed by using B’, H’, D’, or O’ before numeric data.

Syntax:

ra [<mode>]

Parameter	Type	Description
none		Displays current radix
<mode>	Keyword	Sets radix to specified type
	H	Sets radix to hexadecimal
	D	Sets radix to decimal
	O	Sets radix to octal
	B	Sets radix to binary

Examples:

RADIX	Displays the current radix.
RA H	Sets the radix to hexadecimal.

6.50 REGISTER_DISPLAY

Abbreviation: RD

Description:

Displays CPU register contents.

Syntax:

rd

Parameter	Type	Description
none		Displays all register contents

Example:

RD	Displays all register contents

6.51 REGISTER_SET

Abbreviation: RS

Description:

Changes the contents of a register.

Syntax:

rs <register> <value> [<mode>]

Parameter	Type	Description
<register>	Keyword	Register name
<value>	Numeric	Register value
<mode>	Keyword	Data size (optional, default = corresponding register size)
	byte	Byte
	word	Word
	long	Longword
	single	Single-precision floating-point
	double	Double-precision floating-point

Examples:

RS PC_StartUp	Sets the program counter to the address defined by the symbol _StartUp
RS R0 H’1234 WORD	Sets word data H’1234 to R0.

6.52 RESET

Abbreviation: RE

Description:

Resets the microprocessor. All register values are set to the initial values of the device. Memory mapping and breakpoints are not initialized.

Syntax:

re

Parameter	Type	Description
none		Resets the microprocessor

Example:

RE	Resets the microprocessor.

6.53 RESPONSE

Abbreviation: RP

Description:

Specifies the frequency of the window update.
When a long refresh interval is specified, the simulation becomes faster but the response, such as for the break button, becomes slower. Set a frequency appropriate for the machine used.

Syntax:

rp [<instruction number>]

Parameter	Type	Description
<instruction number>	Numeric	Specifies the frequency, in number of instruction executions, the window is to update. 1 to 65535 (default=40000)

Example:

RESPONSE 9	Sets the window to refresh every 9 information executions.

6.54 SLEEP

Abbreviation: None

Description:

Delays command execution for a specified period.

Syntax:

sleep <milliseconds>

Parameter	Type	Description
<milliseconds>	Numeric	Delayed time (ms)

The value must always be specified in decimal.

Example:

SLEEP D’9000	Delays 9 seconds.

6.55 STEP

Abbreviation: ST

Description:

Single step (in source line or instruction units) execution. Performs a specified number of instructions, from current PC. Default is stepping by lines if source debugging is available. Count default is 1.

Syntax:

st [<mode>] [<count>]

Parameter	Type	Description
<mode>	Keyword	Type of single step (optional)
	instruction	Steps by assembly instruction
	line	Steps by source code line
<count>	Numeric	Number of steps (optional, default = 1)

Example:

STEP 9	Steps code for 9 steps.

6.56 STEP_MODE

Abbreviation: SM

Description:

Selects the step mode.

Syntax:

sm <mode>

Parameter	Type	Description
<mode>	Keyword	Type of step mode
	Auto	If Source windows is active, steps by source code line If Disasembly windows is active, steps by assembly instruction
	Assembly	Steps by assembly instruction
	Source	Steps by source code line

Example:

STEP_MODE atuo	Sets the step mode to auto.

6.57 STEP_OUT

Abbreviation: SP

Description:

Steps the program out of the current function. (i.e., a step up). This works for both assembly-language and source level debugging.

Syntax:

sp

Parameter	Type	Description
none		Steps the program out of the current function

Example:

SP	Steps the program out of the current function.

6.58 STEP_OVER

Abbreviation: SO

Description:

Performs a specified number of instructions from current PC.

This command differs from STEP in that it does not perform single step operation in subroutines or interrupt routines. These are executed at full speed.

Syntax:

so [<mode>] [<count>]

Parameter	Type	Description
<mode>	Keyword	Type of stepping (optional)
	instruction	Steps by assembly instruction
	line	Step by source code line
<count>	Numeric	Number of steps (optional, default = 1)

Example:

SO	Steps over 1-step code.

6.59 STEP_RATE

Abbreviation: SR

Description:

Controls the speed of stepping in the STEP and STEP_OVER commands. A rate of 6 causes the fastest stepping. A value of 0 is the slowest.

Syntax:

sr [<rate>]

Parameter	Type	Description
none		Displays the step rate
<rate>	Numeric	Step rate 0 to 6 (6 = fastest)

Examples:

SR	Displays the current step rate.
SR 6	Specifies the fastest step rate.

6.60 STEP_UNIT

Abbreviation: SN

Description:

Sets and displays the execution unit. This command is only supported in the SH-3, SH-3E, SH3-DSP, and SH-4 series.

Syntax:

sn [<mode>]

Parameter	Type	Description
<mode>	Keyword	Sets and displays an instruction unit (optional, default = display).
	stage	Executed in pipeline stage units. Pipeline does not become disordered.
	instruction	Executed in instruction units. Pipeline becomes disordered.

Example:

STEP_UNIT INSTRUCTION	Executed in instruction units (initiates a break after memory access).

6.61 SUBMIT

Abbreviation: SU

Description:

Executes a file of emulator commands. This command can be used even in a command file to be processed. Any error aborts the file.

Syntax:

su <filename>

Parameter	Type	Description
<filename>	Character string	File name

Examples:

SUBMIT COMMAND.HDC	Processes the file COMMAND.HDC.
SU A:SETUP.TXT	Processes the file SETUP.TXT on drive A:.

6.62 SYMBOL_ADD

Abbreviation: SA

Description:

Adds a symbol, or changes an existing one.

Syntax:

sa <symbol> <value>

Parameter	Type	Description
<symbol>	Character string	Symbol name
<value>	Numeric	Value

Examples:

SYMBOL_ADD start H’1000	Defines the symbol start at H’1000.
SA END_OF_TABLE 1FFF	Uses current default radix and defines END_OF_TABLE at H'1FFF .

6.63 SYMBOL_CLEAR

Abbreviation: SC

Description:

Deletes a symbol. If no parameters are specified, deletes all symbols (after confirmation).

Syntax:

sc [<symbol>]

Parameter	Type	Description
none		Deletes all symbols
<symbol>	Character string	Symbol name

Examples:

SYMBOL_CLEAR	Deletes all symbols (after confirmation).
SC start	Deletes the symbol ‘start’.

6.64 SYMBOL_LOAD

Abbreviation: SL

Description:

Loads symbols from file. File must be in XLINK Pentica-b format (i.e. 'XXXXH name'). The symbols are added to the existing symbol table.

Syntax:

sl <filename>

Parameter	Type	Description
<filename>	Character string	File name

Examples:

SYMBOL_LOAD TEST.SYM	Loads the file TEST.SYM.
SL MY_CODE.SYM	Loads the file MY_CODE.SYM.

6.65 SYMBOL_SAVE

Abbreviation: SS

Description:

Saves symbols to a file in XLINK Pentica-b format. The symbol file extension default is **.SYM**.

Syntax:

ss <filename>

Parameter	Type	Description
<filename>	Character string	File name

Examples:

SYMBOL_SAVE TEST	Saves symbol table to TEST.SYM.
SS MY_CODE.SYM	Saves the symbol table to MY_CODE.SYM.

6.66 SYMBOL_VIEW

Abbreviation: SV

Description:

Displays all defined symbols, or those containing the case sensitive string pattern.

Syntax:

sv [<pattern>]

Parameter	Type	Description
none		Displays all symbols
<pattern>	Character string	Character string that should be contained in the symbols to be displayed

Examples:

SYMBOL_VIEW BUFFER	Displays all symbols containing the word BUFFER.
SV	Displays all the symbols.

6.67 TCL

Abbreviation: None

Description:

Enables or disables the TCL.

Syntax:

tcl [<state>]

Parameter	Type	Description
None		Displays the TCL information.
<state>	Keyword	Enables or disables the TCL
	enable	Enables the TCL
	disable	Disables the TCL

Examples:

TCL	Displays the TCL information.
TCL enable	Enables the TCL.
TCL d	Disables the TCL.

6.68 TRACE

Abbreviation: TR

Description:

Displays the trace buffer contents. The record in the buffer that was executed first is 0; older records have positive offset values.

Syntax:

tr [[<start rec> [<count>]] | [<clear>]]

Parameter	Type	Description
<start rec>	Numeric	Record to start display (optional, default = most recent record - 9)
<count>	Numeric	Number of records to be displayed (optional, default = 10)
<clear>	Keyword	Clears all trace records (optional)
	clear	Clears all trace records

Note: When a negative value is specified for <start rec> (-0 cannot be specified), the value is recognized as a PTR value.

Examples:

TR 0 20	Displays twenty lines of trace buffer contents starting from the top of the buffer.
TR	Displays ten lines of trace buffer contents starting from the end of the buffer (the ten most recently executed lines).
TR –20 10	Displays trace buffer contents for which the PTR value is between –20 and -11.
TR C	Clears all trace records.

6.69 TRACE_ACQUISITION

Abbreviation: TA

Description:

Enables or disables trace information acquisition

Syntax:

ta <mode>

Parameter	Type	Description
<mode>	Keyword	Enables or disables trace information acquisition.
	E	Trace information acquisition is enabled.
	D	Trace information acquisition is disabled.

Examples:

TRACE_ACQUISITION E	Trace information acquisition is enabled.
TA D	Trace information acquisition is disabled.

6.70 TRACE_SAVE

Abbreviation: TV

Description:

Saves the trace information in files. The files are in text format, so the default file extension is **.TXT**.

Syntax:

tv<filename>

Parameter	Type	Description
<filename>	Character string	File name

Examples:

TRACE_SAVE TEST	Saves the trace information in TEST.TXT.
TV TRACE.TXT	Saves the trace information in TRACE.TXT

6.71 TRACE_STATISTIC

Abbreviation: TST

Description:

Analyzes the statistic information under the specified conditions.

Syntax:

tst <item> <string>

Parameter	Type	Description
<item>	Character string	Statistic information item to be analyzed
<string>	Character string	Character string that specifies conditions

Example:

TST CODE1 E630	Analyzes the statistic information under condition CODE1 = E630.

Section 7 Messages

7.1 Information Messages

The simulator/debugger outputs information messages as listed in table 7.1 to notify users of execution status.

Table 7.1 Information Messages

Message	Contents
Break Access	The break access condition was satisfied and execution has stopped.
Break Cycle	The break cycle condition was satisfied and execution has stopped
Break Data	The break data condition was satisfied and execution has stopped.
Break Register	The break register condition was satisfied and execution has stopped.
Break Sequence	The break sequence condition was satisfied and execution has stopped.
PC Breakpoint	The breakpoint condition was satisfied and execution has stopped.
Sleep	Execution has been stopped by the SLEEP instruction.
Step Normal End	The step execution succeeded.
Stop	Execution has been stopped by the **[Stop]** button.
Trace Buffer Full	Since the **Break** mode was selected by **Trace buffer full handling** in the **Trace Acquisition** dialog box and the trace buffer became full, execution was terminated.

7.2 Error Messages

The simulator/debugger outputs error messages to notify users of the errors of user programs or operation. Table 7.2 lists the error messages.

Table 7.2 Error Messages

Message	Contents
Address Error	One of the following states occurred: • A PC value was an odd number. • An instruction was read from the internal I/O area. • Word data was accessed to an address other than a multiple of 2. • Longword data was accessed to an address other than a multiple of 4. • The VBR or SP was a value other than a multiple of 4. • An error occurred in the exception processing of an address error. Correct the user program to prevent the error from occurring.
Exception Error	An error occurred during exception processing. Correct the user program to prevent the error from occurring.
File Open Error	An error occurred during opening a file with the break of file-input/output action. Correct the file setting.
File Input Error	An error occurred during reading a file with the break of file-input/output action. Correct the file setting.
File Output Error	An error occurred during writing to a file with the break of file-input/output action. Correct the file setting.
FPU Disable	An attempt was made to execute an FPU instruction while the FPU is disabled (SR.FD = 1). Correct the user program to prevent the error from occurring.
FPU Error	One of the following states occurred during floating-point operation: • An FPU error occurred. • An invalid operation occurred. • A division by zero occurred. • An overflow occurred. • An underflow occurred. • An inaccurate operation occurred. Correct the user program to prevent the error from occurring.

Table 7.2 Error Messages (cont)

Message	Contents
General Invalid Instruction	Either of the following states occurred: • A code other than an instruction was executed. • An error occurred in the exception processing of a reserved instruction exception. Correct the user program to prevent the error from occurring.
Illegal CCR2 Set	The CCR2 value is illegal. Check the setting.
Illegal DSP Operation	Either of the following states occurred: • A shift of more than 32 bits was executed with the PSHA instruction. • A shift of more than 16 bits was executed with the PSHL instruction. Correct the user program to prevent the error from occurring.
Illegal LRU Set	LRU value of the cache is invalid. Check the setting.
Illegal Operation	Either of the following states occurred: • A division by zero occurred during DIV1 instruction execution. • Zero was written to by the SETRC instruction. Correct the user program to prevent the error from occurring.
Illegal PR bit	An attempt was made to execute an FPU instruction while the PR bit value of the FPSCR is illegal. Correct the user program to prevent the error from occurring.
Initial Page Write	Initial page write occurred during simulation. Take necessary procedures such as updating the TLB contents.
Instruction TLB Illegal LRU	An LRU value in the instruction TLB is illegal. Check the setting.
Instruction TLB Miss	An instruction TLB miss occurred during memory access. Take necessary procedures such as updating the TLB contents.
Instruction TLB Protection Violation	An instruction TLB protection exception occurred during memory access. Take necessary procedures such as updating the TLB contents.
Interrupt Exception	An interrupt exception occurred, execution halted
Invalid DSP Instruction Code	An invalid instruction code was detected in the DSP parallel instruction. Correct the user program to prevent the error from occurring.
Invalid Slot Instruction	Either of the following states occurred: • An instruction that changes a PC value (a branch instruction) immediately after a delayed branch instruction was executed. • An error occurred during the exception processing of an invalid slot instruction. Correct the user program to prevent the error from occurring.

Table 7.2 Error Messages (cont)

Message	Contents
Illegal Combination BSC Register	An attempt was made to access the area for which the BSC register setting is invalid. Correct the user program to prevent the error from occurring.
Memory Access Error	One of the following states occurred: • A memory area that had not been allocated was accessed. • Data was written to a memory area having the write protect attribute. • Data was read from a memory area having the read disable attribute. • A memory area in which memory does not exist was accessed. Allocate memory, change the memory attribute, or correct the user program to prevent the memory from being accessed.
Multiple Exception	Multiple exceptions occurred. Correct the user program to prevent the error from occurring.
Slot FPU Disable	An attempt was made to execute an FPU instruction in a delay slot while the FPU is disabled (SR.FD = 1). Correct the user program so that no error occurs.
System Call Error	System call error occurred. Modify the incorrect contents of registers R0, R1, and parameter block.
TLB Invalid	TLB invalid exception occurred during simulation or during command execution. Take necessary procedures such as updating the TLB contents.
TLB Miss	TLB miss occurred during simulation or during command execution. Take necessary procedures such as updating the TLB contents.
TLB Multiple Hit	Multiple TLB entries were hit when a virtual address was accessed during simulation or command execution. TLB is not correctly set. Modify TLB contents and user program (handler routine).
TLB Protection Violation	Illegal TLB protection exception occurred during simulation.
Unified TLB Miss	A unified TLB miss occurred during memory access. Take necessary procedures such as updating the Unified TLB (UTLB) contents.
Unified TLB Multiple Hit	Multiple unified TLB entries were hit when a virtual address was accessed. TLB is not correctly set. Modify TLB contents and user program (handler routine).
Unified TLB Protection Violation	A unified TLB protection exception occurred during memory access. Take necessary procedures such as updating the Unified TLB (UTLB) contents.

Appendix A - GUI Command Summary

Menu	Item	Accelerator	Toolbar Graphic
View	Workspace	Alt+K	
	Output	Alt+U	
	Breakpoints	Shift+Ctrl+B	
	Coverage…	Shift+Ctrl+O	
	Command Line	Ctrl+L	>_
	Disassembly	Ctrl+D	
	IO	Ctrl+I	I/O
	Image...	Shift+Ctrl+G	
	Labels	Shift+Ctrl+A	
	Locals	Ctrl+Shift+W	
	Memory...	Ctrl+M	
	Performance Analysis	Shift+Ctrl+P	
	Profile	Shift+Ctrl+F	
	Registers	Ctrl+R	R1
	Status	Ctrl+U	
	Trace	Ctrl+T	
	Watch	Ctrl+W	
	Waveform...	Shift+Ctrl+V	
	TLB…		TLB
	Cache…		
	Simulated I/O	Shift+Ctrl+I	

Menu	Item	Accelerator	Toolbar Graphic
	Stack Trace	Ctrl+K	
View	Trigger	Shift+Ctrl+R	
Options	Debug Settings…		
	Radix ➢		
	Hex		16
	Decimal		10
	Oct		8
	Bin		2
	Simulator ➢		
	System…		
	Memory Resource...		
Debug	Reset CPU		
	Go	F5	
	Reset Go	Shift+F5	
	Go to Cursor		
	Set PC to Cursor		I PC
	Run…		
	Step In	F11	
	Step Over	F10	
	Step Out	Shift+F11	
	Step…		
	Step Mode ➢		
	Auto		
	Assembly		
	Source		
	Halt Program	Esc	STOP
	Initialize		
	Disconnect		
	Download Modules		
	Unload Modules		

Menu	Item	Accelerator	Toolbar Graphic
Memory	Search…		
	Copy…		
	Compare…		
	Fill…		
	Refresh		
	Configure Overlay…		