User's Manual

SH Series Simulator/Debugger

User's Manual

Renesas Electronics
www.renesas.com

Rev.1.0 1998.10

Preface

The SH Series Simulator/Debugger (referred to in this manual as the simulator/debugger) is a software tool that simulates programs for the SuperH RISC engine series of microprocessors to support program development and debugging on a host computer.

This user's manual describes the simulator/debugger functions and its usage. Read this manual and the following manuals, and understand their contents before using the simulator/debugger.

- Hitachi Debugging Interface User’s Manual
- SuperH RISC engine Series C/C++ Compiler User’s Manual
- SuperH RISC engine Series Cross Assembler User’s Manual
- H Series Linkage Editor, Librarian, Object Converter User’s Manual

For details on each SuperH RISC engine series microprocessor, refer to the appropriate hardware and programming manuals.

Notes:
1. This manual assumes the operating environment to be the English version of Microsoft® Windows®95 operating system on an IBM PC.
2. Microsoft®, Windows®, and WindowsNT® are registered trademarks of Microsoft Corporation in the United States and/or in other countries.
3. IBM PC is a personal computer administered by International Business Machines Corporation.
4. ELF/DWARF is the name of an object format developed by the Tool Interface Standard Committee.
5. All other products or brand names are trademarks or registered trademarks of their respective holders.

Contents

Figures

Tables

Section 1 Overview

The simulator/debugger V.4 provides simulation functions for SuperH RISC engine series (SH-1, SH-2, SH-2E, SH-3, SH-3E, SH-4, and SH-DSP) microprocessors and provides debugging functions for programs written in C, C++, or assembly language. Therefore, the simulator/debugger promotes efficient debugging of programs for the SuperH RISC engine series. This simulator/debugger operates with the Hitachi Debugging Interface (HDI), a Windows®-based interface program.

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
- H series linkage editor
- H series librarian
- H series object converter

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 trace instructions or subroutines
  - Functions to stop or continue execution when an error occurs during user program execution
  - Function-unit performance measurement
  - A comprehensive set of break functions
  - Functions to set or edit memory maps
  - Functions to display function call history
- The breakpoint, memory map, performance, and trace can be set in the dialog box under Windows®. Environments corresponding to each memory map of the SuperH RISC engine series microprocessors can be set in the dialog box.

1.2 Target User Program

Load modules in SYSROF format, S-type format, and ELF/DWARF 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 SYSROF and S-type user program. Figure 1.2 shows the creation of ELF/DWARF-format user program.

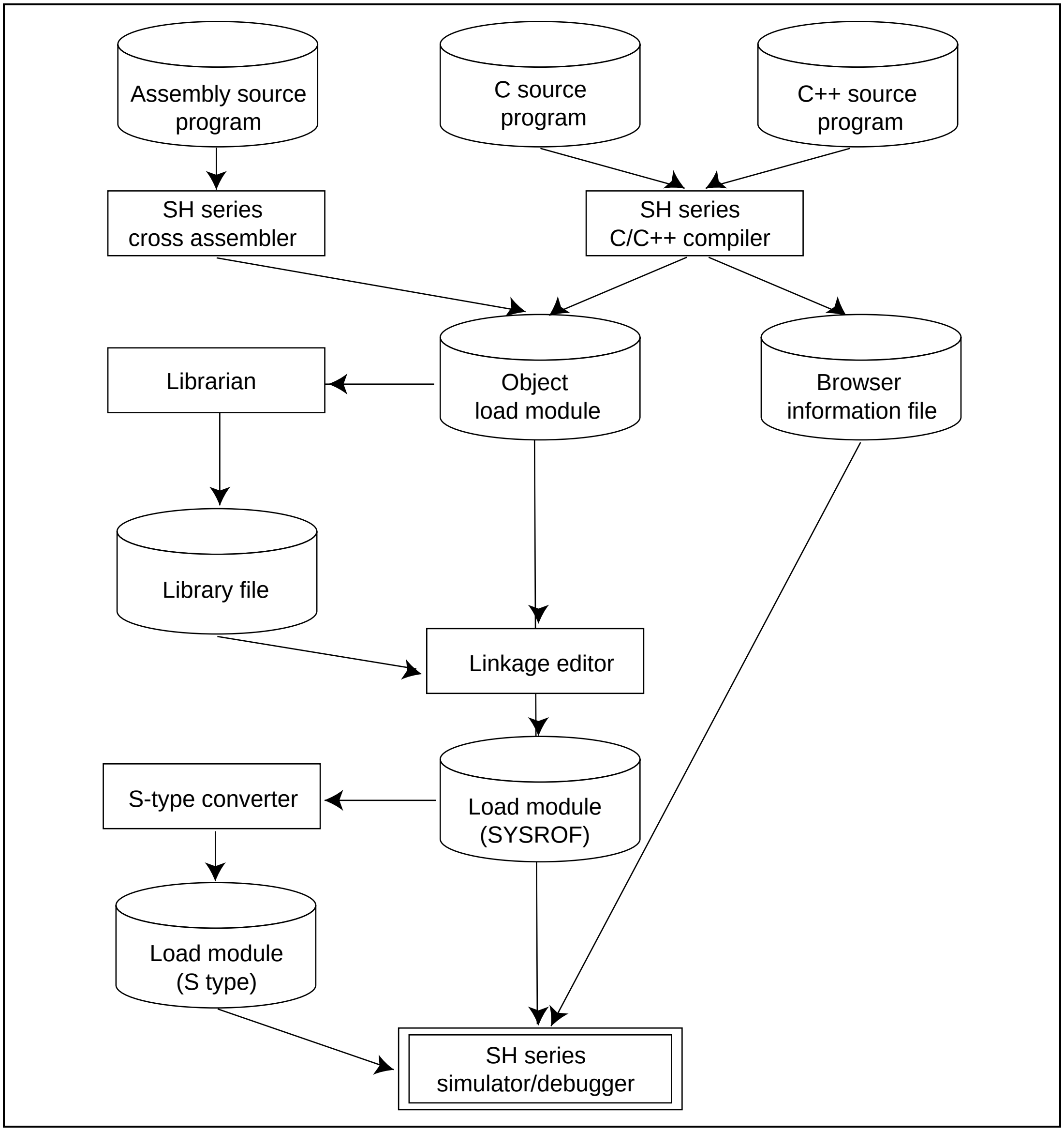


Figure 1.1 Creation of SYSROF and S-Type User Programs

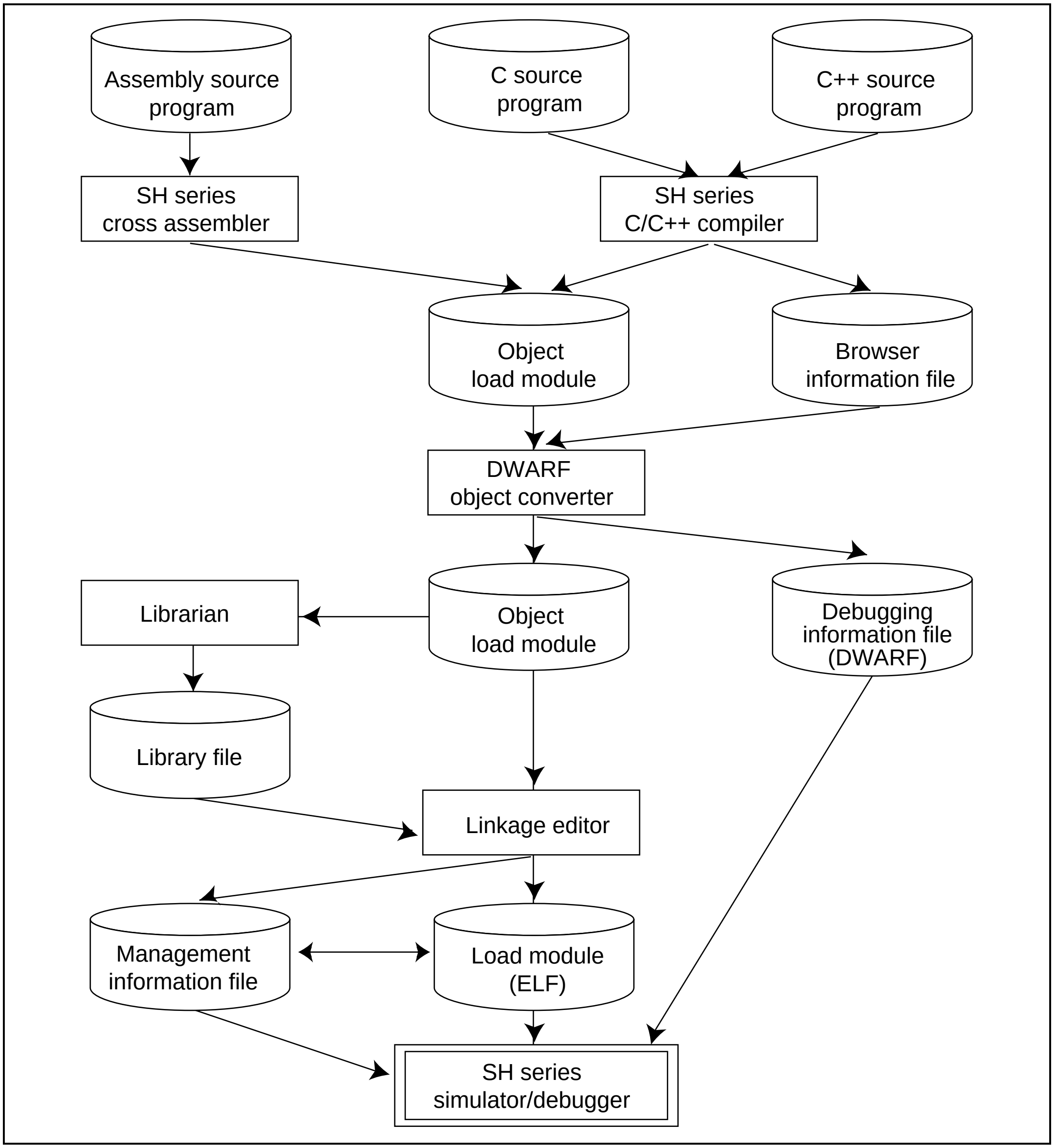


Figure 1.2 Creation of ELF/DWARF-Format User Programs

1.3 Simulation Range

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

- All CPU instructions (pipeline simulation)
- Exception processing
- Registers
- All address areas
- MMU (only for SH-3, SH-3E, and SH-4 series)
- Cache (only for SH-3, SH-3E, and SH-4 series)
- DMAC (only for SH-4 series)
- BSC (only for SH-4 series)

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 Simulator/Debugger Functions

This section describes the SuperH RISC engine series simulator/debugger. Note that the endian, MMU, cache, and control registers can be used only in the SH-3, SH-3E, and SH-4 series, and the BSC and DMAC can be used only in the SH-4 series.

2.1 Simulator/Debugger Memory Management

2.1.1 Memory Map Specification

A memory map can be specified in the **System Configuration** 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 5.4, System Configuration Dialog Box. The user program can be executed in all areas except for the internal I/O area.

2.1.2 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 Modify** 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.

2.2 Endian

In the SH-3, SH-3E, and SH-4 series, 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. Use **[Endian]** in the **System Configuration** dialog box to specify the endian.

The specified endian is valid for all accesses to external memory; word or longword data is written to or read from the memory in the specified byte order.

2.3 Pipeline Reset Processing

The simulator/debugger, which simulates the pipeline, 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 instruction execution cycles is zero-cleared.

2.4 Memory Management Unit (MMU)

For the SH-3, SH-3E, and SH-4 series, the simulator/debugger simulates MMU operations such as TLB operations, address translation, or MMU-related exceptions (TLB miss, TLB protection exception, TLB invalid exception, and initial page write). 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 and SH-3E 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 5.40, TLB Dialog Box, section 5.41, TLB Modify Dialog Box, and section 5.42, 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 5.44, Instruction TLB Dialog Box, through section 5.49, 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.**

2.5 Cache

2.5.1 Displaying Cache Contents

For the SH-3, SH-3E, and SH-4 series, the simulator/debugger simulates cache operations. Cache operations during user program execution can be monitored, and the cache hit ratio can be displayed. The cache functions depend on the CPU.

SH-3 and SH-3E series:

The following dialog boxes are provided to manipulate the 128-entry 4-way cache:

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

For details, refer to section 5.50, Cache Dialog Box, and section 5.51, Cache Modify Dialog Box. For the cache control register, refer to section 5.23, CCR Dialog Box.

The cache is mapped in the range H'F0000000 to H'F1FFFFFF.

SH-4 Series:

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 5.53, Instruction Cache Dialog Box, through section 5.56, Operand Cache Modify Dialog Box. For the cache control registers, refer to section 5.23, CCR Dialog Box, and section 5.24, QACR0 and QACR1 Dialog Boxes.

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 selecting the [Load Program...] menu, to the area where the cache is mapped, or copying memory data to this area by selecting the [Move Memory...] menu, clear the AT bit of the MMUCR to zero to disable the MMU.

2.5.2 Cache Hit Ratio

The simulator debugger checks and displays the cache hit ratio.

Checking and Displaying the Cache Hit Ratio: The cache hit ratio is displayed in percentage in the **System Status** window. The displayed value is the ratio of the cache hit count to the cache access count, that is, the sum of the cache hit count and cache miss count.

Initializing the Cache Hit Ratio: The cache hit ratio is initialized when the simulator/debugger is initiated, the pipeline is reset, or the CCR register value is modified.

2.6 Bus State Controller (BSC)

For the SH-4 series, 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 2.1 lists the memory types that can be specified for the SH4 series.

Table 2.1 Memory Types for the SH-4 Series

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 2.1 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 5.7, Memory Map Dialog Box. For details on BSC control register setting, refer to section 5.30, MCR Dialog Box, through section 5.39, RFCR Dialog Box.

2.7 Direct Memory Access Controller (DMAC)

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

For details on DMAC control register setting, refer to section 5.25, SAR0-SAR3 Dialog Boxes, through section 5.29, DMAOR Dialog Box.

2.8 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, 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 **[Mode]** setting in the **System Configuration** dialog box.

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

- When **[Continue]** is selected (continue 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 and SH-3E Series:

- When **[Continue]** is selected (continue 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 registers EXPEVT and INTEVT. 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 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 (continue 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 registers EXPEVT and INTEVT. 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 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.

2.9 Control Registers

For the SH-3, SH-3E, 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. Therefore, a user program using exception processing, MMU control, cache control, BSC control, and DMAC control can be simulated and debugged.

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
	QACR0 and QACR1*:	Queue address control registers 0 and 1
BSC	BCR1 and BCR2*:	Bus control registers 1 and 2
	WCR1 to WCR3*:	Wait state control registers 1 to 3
	MCR*:	Individual memory control register
	RTCSR*:	Refresh timer control/status register
	RTCNT*:	Refresh timer counter register
	RTCOR*:	Refresh time constant register
	RFCR*:	Refresh count register

DMAC	SAR0 to SAR3*:	DMA source address registers 0 to 3
	DAR0 to DAR3*:	DMA destination address registers 0 to 3
	DMATCR0 to DMATCR3*:	DMA transfer count registers 0 to 3
	CHCR0 to CHCR3*:	DMA channel control registers 0 to 3
	DMAOR*:	DMA operation register

Note: The registers marked with * are supported only for the SH-4 series.

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 **Control Registers** window and the dialog box for each register. For details, refer to section 5.14, Control Registers Window, through section 5.39, RFCR Dialog Box.

2.10 Trace

The simulator/debugger writes the results of the execution of each instruction execution into the trace buffer. The trace buffer can hold the results for up to 1024 instruction executions. The acquisition conditions of the trace information acquisition can be specified in the **Trace Acquisition** dialog box displayed by using the **[Acquisition]** button 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 SH-DSP Series:

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

SH-3 and SH-3E Series:

- C/C++ or assembly-language source programs
- 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

- Instruction number that was decoded (enclosed by [ ] when no memory access was performed)
- Instruction number that was executed
- Instruction number that accessed memory
- Instruction number that wrote back data
- Data access information (destination and accessed data)

SH-4 Series:

- C/C++ or assembly-language source programs
- Total number of instruction execution cycles
- 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)

The trace information can be searched. The search conditions can be specified in the **Trace Search** dialog box displayed by using the **[Find]** button in the **Trace** window.

For details, refer to section 5.11, Trace Window, through section 5.13, Trace Search Dialog Box.

2.11 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 2.2 to be executed by the user program. When the I/O processing is executed, the **Simulated I/O** window must be open.

Table 2.2 I/O Functions

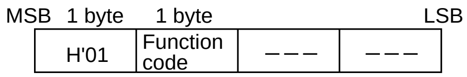
Function Code	Function Name	Description
H'21	GETC	Inputs one byte from the standard input device
H'22	PUTC	Outputs one byte to the standard output device
H'23	GETS	Inputs one line from the standard input device
H'24	PUTS	Outputs one line to the standard output device
H'25	FOPEN	Opens a file
H'06	FCLOSE	Closes a file
H'27	FGETC	Inputs one byte from a file
H'28	FPUTC	Outputs one byte to a file
H'29	FGETS	Inputs a line from a file
H'2A	FPUTS	Outputs a line to a file
H'0B	FEOF	Checks for end of file
H'0C	FSEEK	Moves the file pointer
H'0D	FTELL	Returns the current position of the file pointer

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

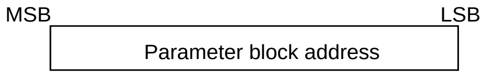
1. Set the address specialized for I/O processing in the **[System Call Address]**, and execute the program.
2. When detecting a subroutine call instruction (BSR, JSR, or BSRF), that is, a system call to the specialized address during user program execution, the simulator/debugger performs I/O processing 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 2.2) to the R0 register



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



- 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 system call 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 2.2

(2) Function name

(3) Function code

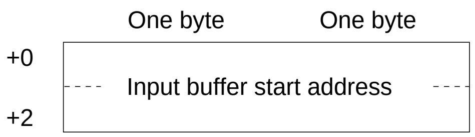
(4) I/O overview

(5) I/O parameter block

(6) I/O parameters

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

Parameter Block

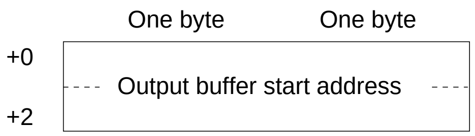


Parameters

- 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

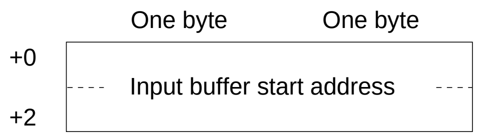


Parameters

- 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

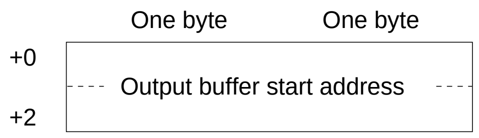


Parameters

- 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



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

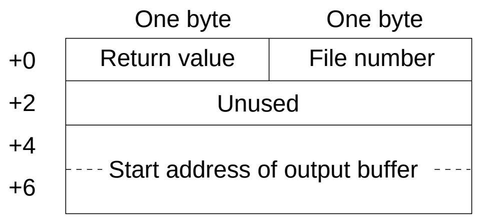
	One byte	One byte
+0	Return value	File number
+2	Unused	
+4	Start address of input buffer	
+6		

Parameters

- Return value (output)
  0: Normal completion
  –1: EOF detected or error
- 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



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 or error
- 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.

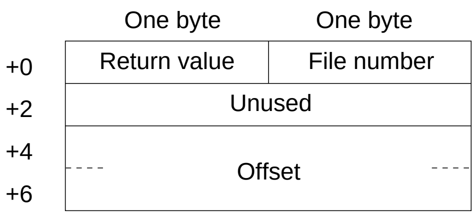
  1: The offset specifies the position as a byte count from the current file pointer.

  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



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
```

2.12 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

2.12.1 Break Due to the Satisfaction of a Break Command Condition

There are five 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_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

When a break condition is satisfied during user program execution, the instruction at the breakpoint may or may not be executed before a break depending on the type of break, as listed in table 2.3.

Table 2.3 Processing When a Break Condition is Satisfied

Command	Instruction When a Break Condition is Satisfied
BREAKPOINT	Not executed
BREAK_ACCESS	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 and execution stops.

2.12.2 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 **System Configuration** dialog box specifies whether to stop or continue the simulation when such an error occurs. Table 2.4 lists the error messages, error causes, and the action of the simulator/debugger in the continuation mode.

Table 2.4 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 a memory area where memory does not exist	
Illegal Operation	Zero division executed by the DIV1 instruction	Operates similar to the actual device operation.
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 and SH-3E 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 2.5 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 2.5 Register States at Simulation Error Stop

Error Message	PC Value
Memory Access Error	• When an instruction is read: — SH-DSP The third instruction address before the instruction that caused the error. — SH-1, SH-2, SH-2E, SH-3, SH-3E, and SH-4 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.

2.12.3 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

2.12.4 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.

2.12.5 Break Due to the [STOP] Button

Users can forcibly terminate execution by pressing 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.

2.13 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 **Find Memory** dialog box
- Register values displayed in the **Registers** window (only single-precision floating-point data)
- 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 **System Configuration** 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.

2.14 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 **Program** 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 2.12, Break Conditions.
- When register values are modified while simulation stops due to the above break conditions.
- While single-step execution is performed.

Note that the functions called after an interrupt is not displayed.

For details, refer to section 5.58, Stack Trace Window.

Section 3 Installation

The simulator/debugger operates in conjunction with the Hitachi Debugging Interface (HDI). This section describes how to install the HDI software without using the installer in the provided CD-ROM and to confirm the simulator/debugger operation with the HDI.

3.1 Installing the HDI Software

The HDI software can be installed in Windows® 95 or WindowsNT® operating on an IBM PC. Install the HDI software by using the installation disk as follows:

- Run Windows®.
- Insert the first HDI installation disk into the **A** drive.
- Click **[Run]** from the start menu to display the following dialog box. Type `A:\hdisimsh.exe` and click the **[OK]** button.

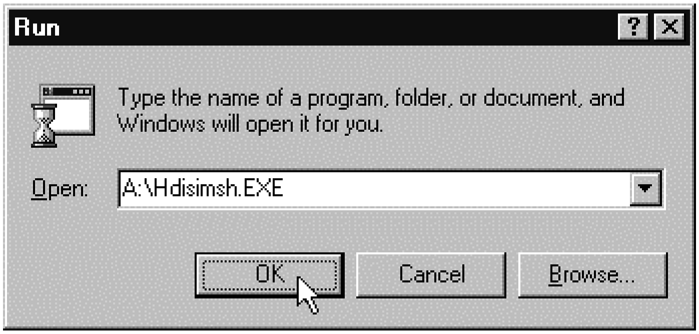
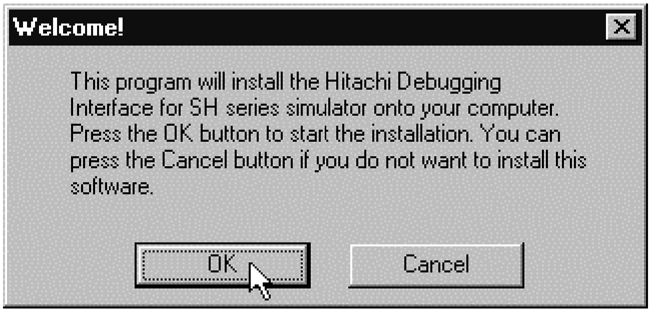

Run
Type the name of a program, folder, or document, and Windows will open it for you.
Open:
A:\Hdisimsh.EXE
OK
Cancel
Browse...

Figure 3.1 Run Dialog Box

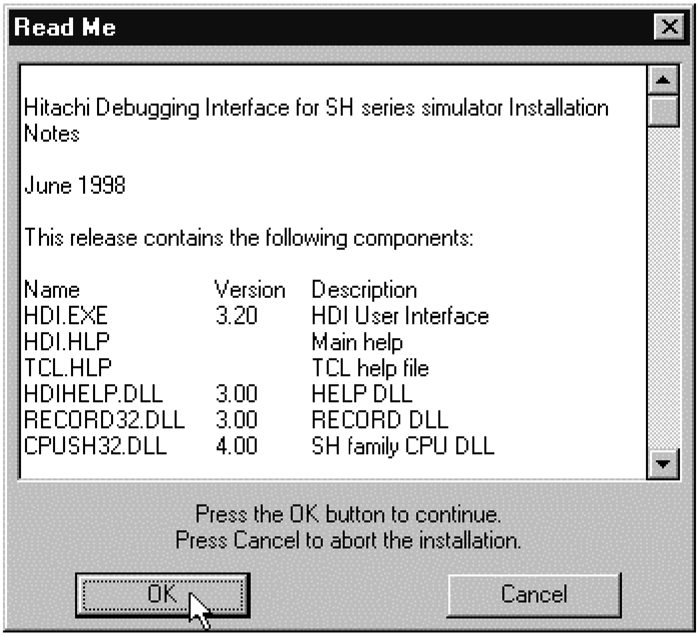
- This runs the HDI installer, and the following **Welcome!** dialog box will be displayed. Click the **[OK]** button to proceed with the installation.



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

Figure 3.2 Welcome! Dialog Box

- The following dialog box then displays version information on the HDI you are installing. Confirm that the version information is the same as the version information written on the floppy disk label, then click the **[OK]** button to proceed.



Hitachi Debugging Interface for SH series simulator Installation Notes

June 1998

This release contains the following components:

Name	Version	Description
HDI.EXE	3.20	HDI User Interface
HDI.HLP		Main help
TCL.HLP		TCL help file
HDIHELP.DLL	3.00	HELP DLL
RECORD32.DLL	3.00	RECORD DLL
CPUSH32.DLL	4.00	SH family CPU DLL

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

Figure 3.3 Read Me Dialog Box

- The **Select Destination Directory** dialog box then allows you to select a directory for installing the HDI and to click the **[OK]** button. When installing into the default directory `C:\HDI`, just click the **[OK]** button.

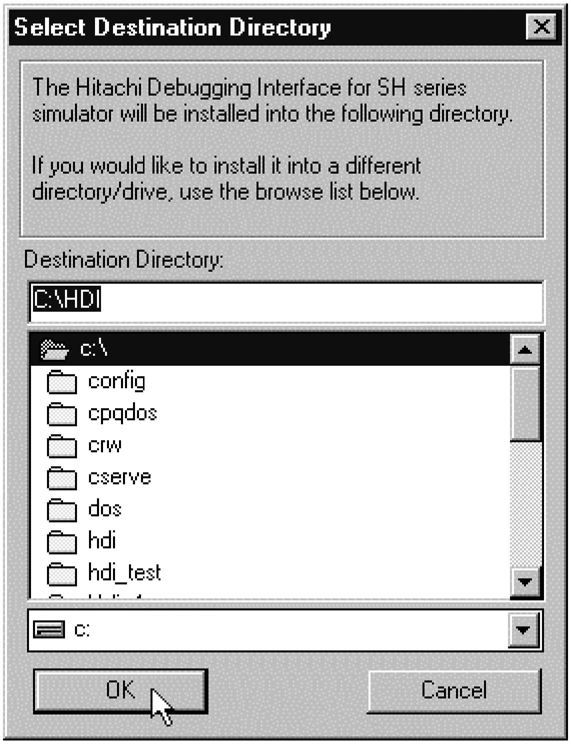

Select Destination Directory
The Hitachi Debugging Interface for SH series simulator will be installed into the following directory.
If you would like to install it into a different directory/drive, use the browse list below.
Destination Directory:
C:\HDI
c:\
config
cpqdos
crw
cserve
dos
hdi
hdi_test
c:
OK
Cancel

Figure 3.4 Select Destination Directory Dialog Box

- When the specified directory already exists, the **Install** dialog box is displayed. Check the information concerning the installation then click the **[Yes]** button to proceed. If you want to change the directory, click the **[No]** button. The **Select Destination Directory** dialog box then allows you to select another directory.

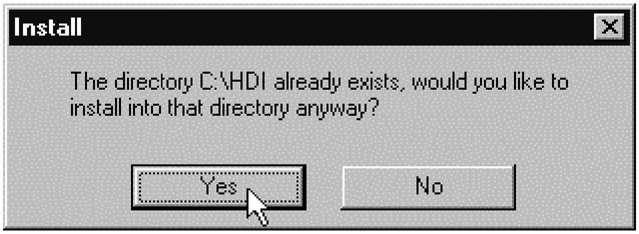
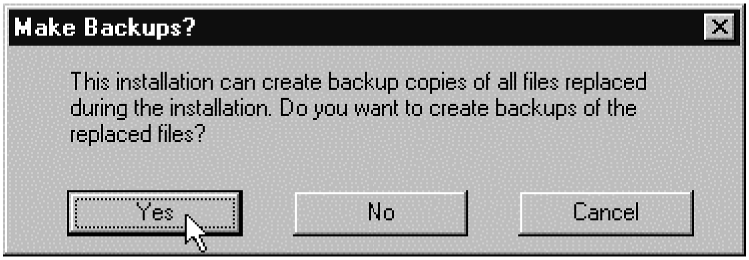

Install
The directory C:\HDI already exists, would you like to install into that directory anyway?
Yes
No

Figure 3.5 Install Dialog Box

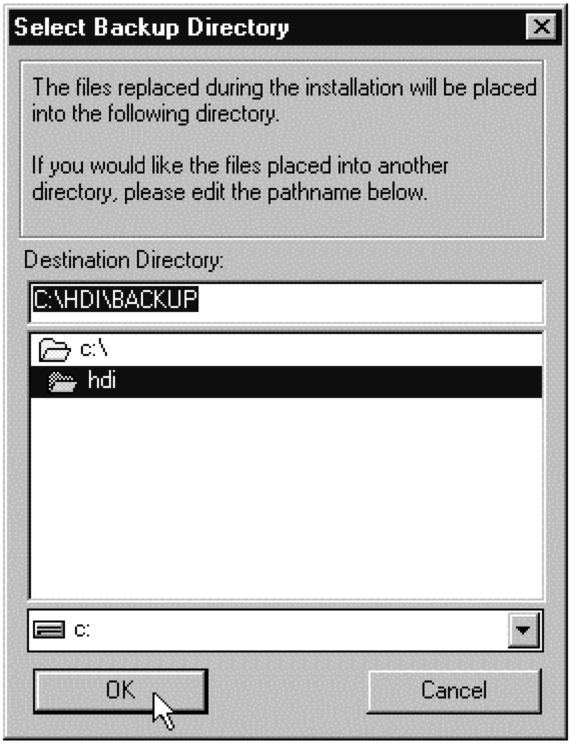
- Clicking the **[Yes]** button in the **Install** dialog box displays the **Make Backups?** dialog box to ask you whether backups should be made of the files replaced by the installation. Click the **[Yes]** button to save any files, such as earlier versions of the HDI, that may be replaced as part of the installation, or the **[No]** button if you do not want to make a backup.


Make Backups?

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

Figure 3.6 Make Backups? Dialog Box

- When you choose the **[Yes]** button in the **Make Backups?** dialog box, the **Select Backup Directory** dialog box is displayed. Specify the backup directory name then click the **[OK]** button to proceed. If saving into the default directory `C:\HDI\BACKUP`, just click the **[OK]** button.


Select Backup Directory

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

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

Destination Directory:

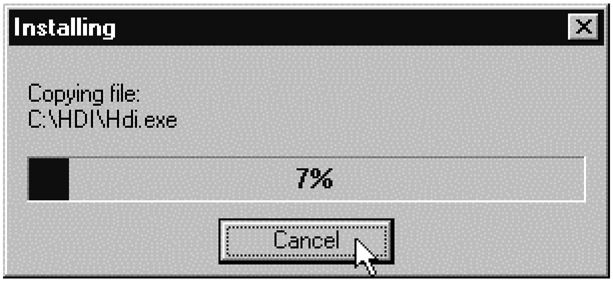
C:\HDI\BACKUP

c:\
hdi

c:

Figure 3.7 Select Backup Directory Dialog Box

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

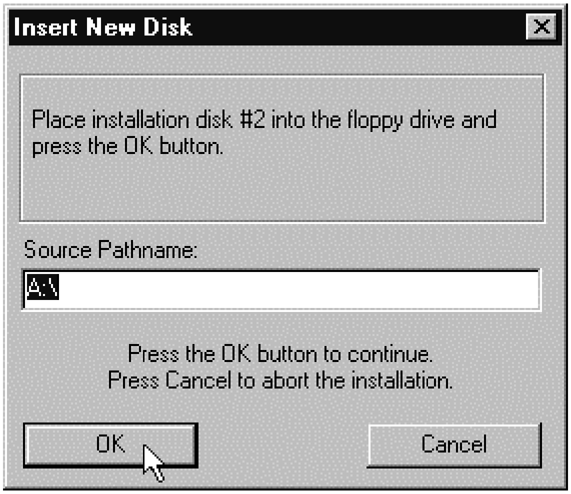

Installing

Copying file:
C:\HDI\Hdi.exe

7%

Figure 3.8 Installing Dialog Box

- The **Insert New Disk** dialog box will appear during installation. Insert the second installation disk into the A drive and click the **[OK]** button.


Insert New Disk

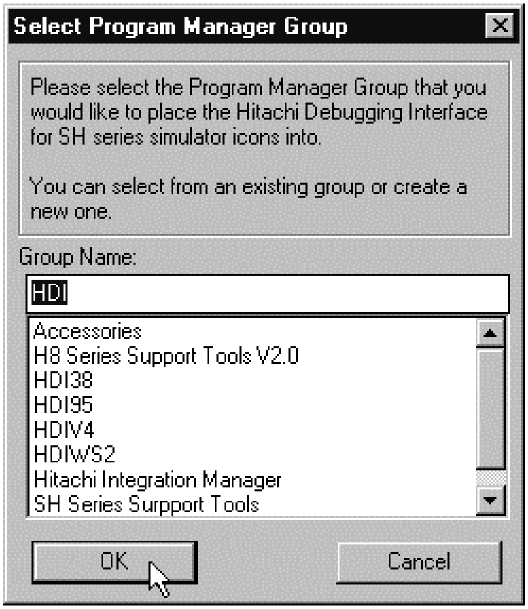
Place installation disk #2 into the floppy drive and press the OK button.

Source Pathname:
A:\

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

Figure 3.9 Insert New Disk Dialog Box

- Finally the **Select Program Manager Group** dialog box allows you to specify the program group name for the HDI icons. If specifying the default group name (HDI), just click the **[OK]** button.


Select Program Manager Group

Please select the Program Manager Group that you would like to place the Hitachi Debugging Interface for SH series simulator icons into.

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

Group Name:

HDI

Accessories
H8 Series Support Tools V2.0
HDI38
HDI95
HDIV4
HDIWS2
Hitachi Integration Manager
SH Series Surpport Tools

Figure 3.10 Select Program Manager Group Dialog Box

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

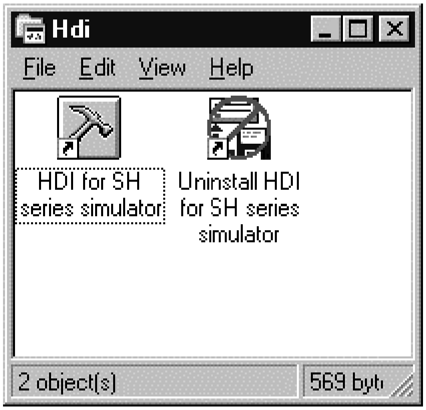

Hdi
File Edit View Help
HDI for SH series simulator
Uninstall HDI for SH series simulator
2 object(s)
569 byt

Figure 3.11 Hdi Program Group

These icons represent the following functions:

HDI for SH series simulator: The HDI program
Uninstall HDI for SH series simulator: The uninstall program

The installing procedure is now completed.

3.2 Files Associated with the HDI Software

This installer installs the files into the following directories.

Directory Specified with the Select Destination Directory Dialog Box: The files listed in table 3.1 are installed.

Table 3.1 HDI Software Files Installed into the Specified Directory

File Name	Function
HDI.EXE	HDI Hitachi debugging interface program file
UNWISE.EXE	Uninstall program file
*.HLP	Help file used by the HDI program
*.DLL	Dynamic linked library file used by the HDI program
HDI.HRF	HDI information file for Hitachi Integrated Manager (HIM)
HDISIMSH.TXT	Read Me file displayed during installation
HDISIMSH.LOG	Installation log file used during the uninstall procedure

Windows Directory (C:\WINDOWS, C:\WIN95, or C:\WINNT): The file shown in table 3.2 is installed.

Table 3.2 HDI Software File Installed into the Windows Directory

File Name	Function
HDI.INI	HDI setting file
HIMTOOLS.HDB	HIM database

3.3 Checking the System

The next step is to run the HDI software to check that the simulator/debugger is working correctly.

Double-click the **[HDI for SH series simulator]** icon to start the simulator/debugger.

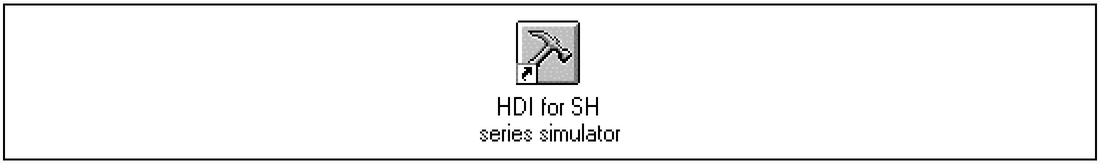
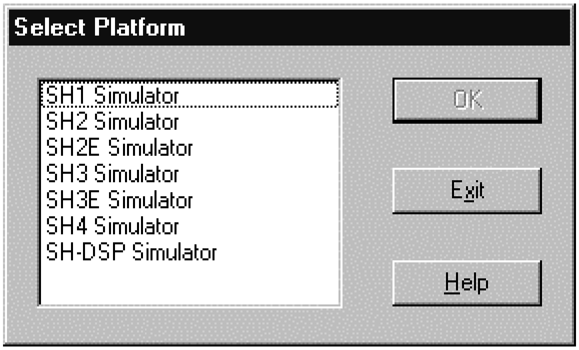


Figure 3.12 HDI for SH series simulator Icon

The **Select Platform** dialog box is then displayed. Select the platform and click the **[OK]** button.



SH1 Simulator
SH2 Simulator
SH2E Simulator
SH3 Simulator
SH3E Simulator
SH4 Simulator
SH-DSP Simulator

Figure 3.13 Select Platform Dialog Box

If the simulator/debugger is set up correctly, the Hitachi Debugging Interface window will be displayed, and "Link up" will be shown in the status bar at the bottom of the window.

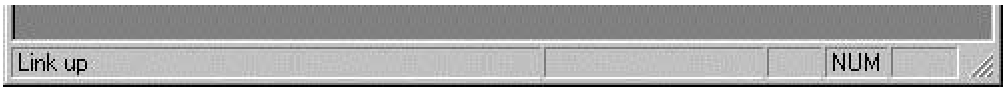


Figure 3.14 HDI Status Bar

Section 4 Tutorial

The following describes a sample program debugging session, designed to introduce the main features of the simulator/debugger used in conjunction with the Hitachi Debugging Interface software.

4.1 Introduction

This sample program is based on a C program that sorts ten random data items first in ascending order, then in descending order.

The sample program performs the following actions:

- With the `main` function, random data to be sorted is generated.
- With the `sort` function, an array is input, within which the random data generated by the `main` function is stored, and the data is sorted in ascending order.
- With the `change` function, the array generated by the `sort` function is input, and the data is sorted in descending order.

The sample program is provided on the installation disk as file `sort.c`. A compiled version of the tutorial is provided in the SYSROF format in file `sort.abs`.

4.2 Running HDI

To run the HDI Interface program, double-click the **[HDI for SH series simulator]** icon.

Figure 4.1 Icon of HDI for SH series simulator

4.3 Selecting the Target

The HDI can be extended to support multiple target platforms. If your system is set up for more than one platform, you will be prompted to choose a platform for the current session.

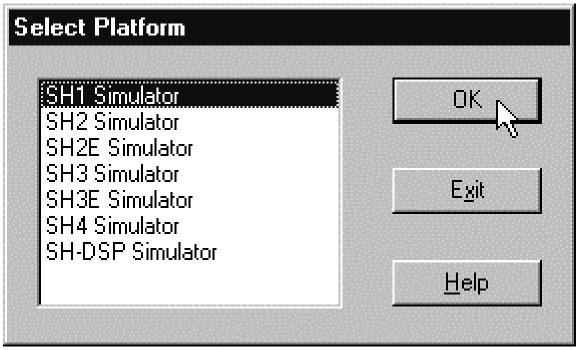
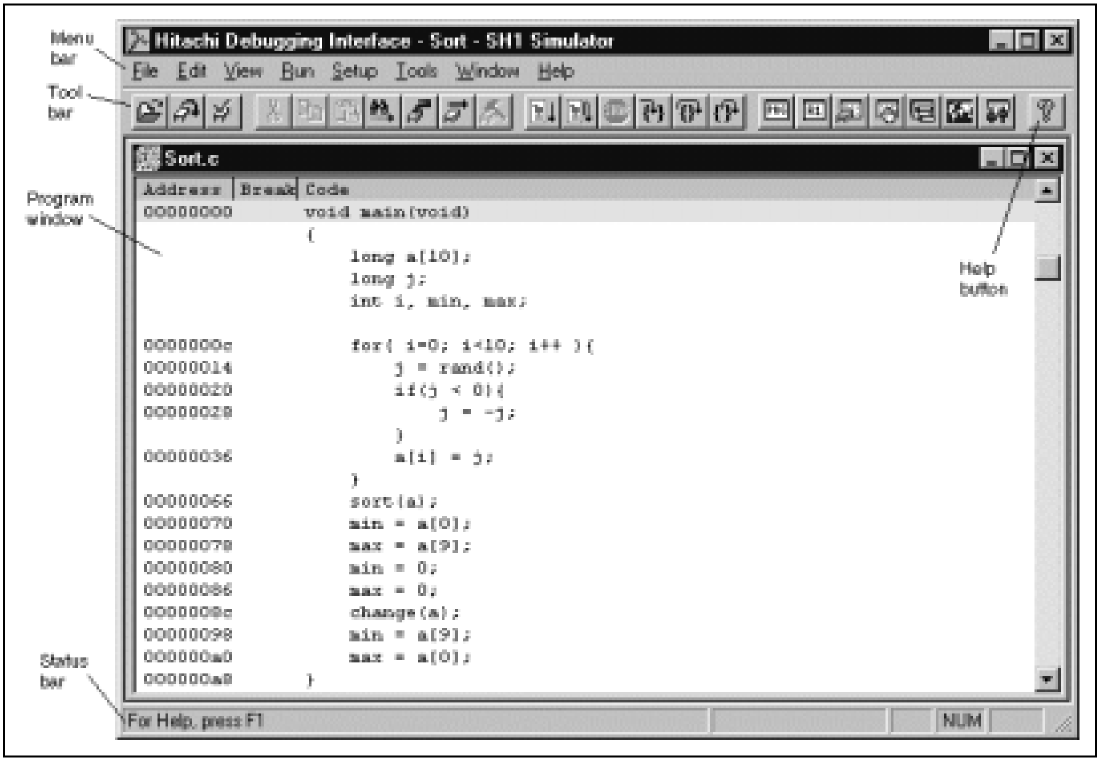


Figure 4.2 Select Platform Dialog Box

For this tutorial, select `SH1 Simulator` and click the **[OK]** button to continue.

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

When the simulator/debugger has been successfully set up, the Hitachi Debugging Interface window will be displayed, with the message "Link up" in the status bar. Figure 4.3 shows the key functions of the window:



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

0000000c             for( i=0; i<10; i++ ){
00000014                 j = rand();
00000020                 if(j < 0){
00000028                     j = -j;
                         }
00000036                 a[i] = j;
                     }
00000066             sort(a);
00000070             min = a[0];
00000078             max = a[9];
00000080             min = 0;
00000086             max = 0;
0000008c             change(a);
00000098             min = a[9];
000000a0             max = a[0];
000000a8         }
```

Figure 4.3   Hitachi Debugging Interface Window

The key functions of Hitachi Debugging Interface are described in the following sections.

Menu bar: Give you access to the HDI commands for using the HDI debugger.

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

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

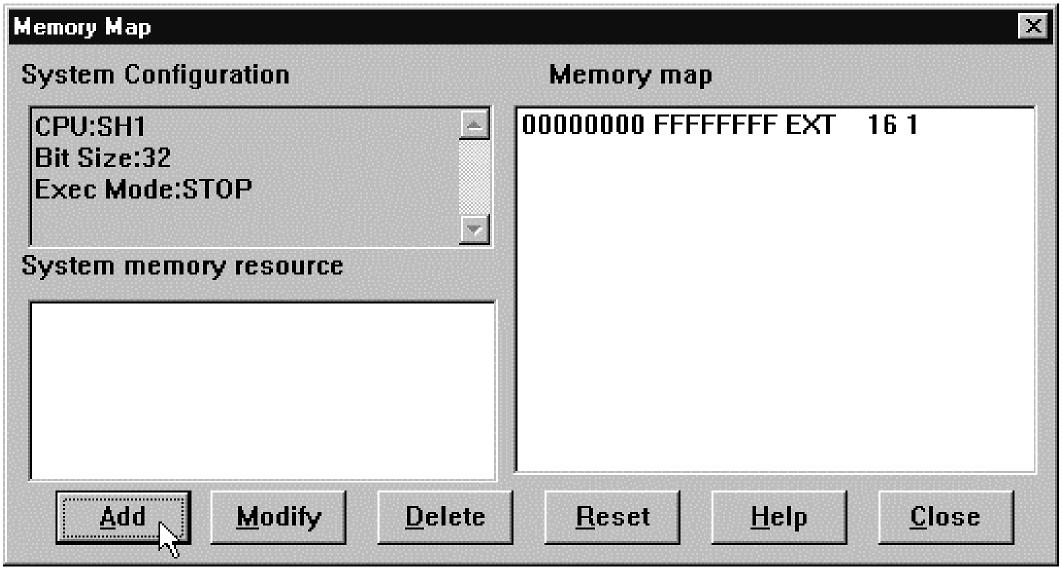
Status bar: Displays the status of the simulator/debugger, and progress information about downloading.

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

4.4 Mapping the Memory Resource

The next step is to map the memory resource used to operate an application being developed.

- Choose **[Memory Mapping Window]** from the **[View]** menu to display the current memory mapping.

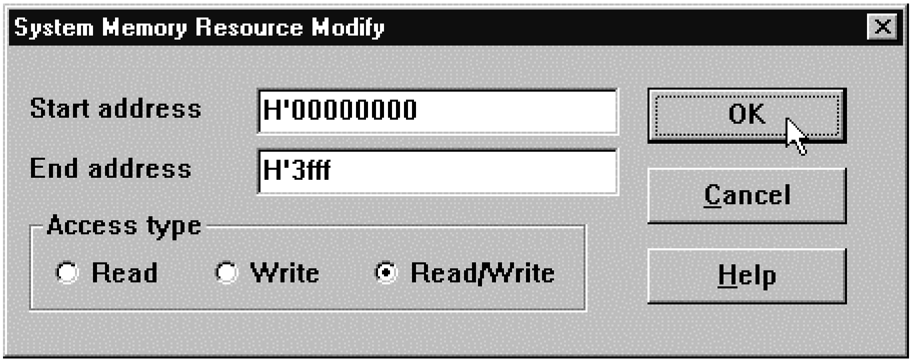


CPU:SH1
Bit Size:32
Exec Mode:STOP

```
00000000 FFFFFFFF EXT    16 1
```

Figure 4.4 Memory Map Dialog Box

- Clicking the **[Add]** button displays the **System Memory Resource Modify** dialog box.



Start address H'00000000
End address H'3fff
Access type
Read Write [✓ Read/Write]

Figure 4.5 System Memory Resource Modify Dialog Box

In the [**Access type**] box, you can specify one of the following three access types:

— Read: Only read enabled

— Write: Only write enabled

— Read/Write: Read and write enabled

For this tutorial, map the memory area of addresses ranging from H'00000000 to H'00003FFF as a read/write enabled area.

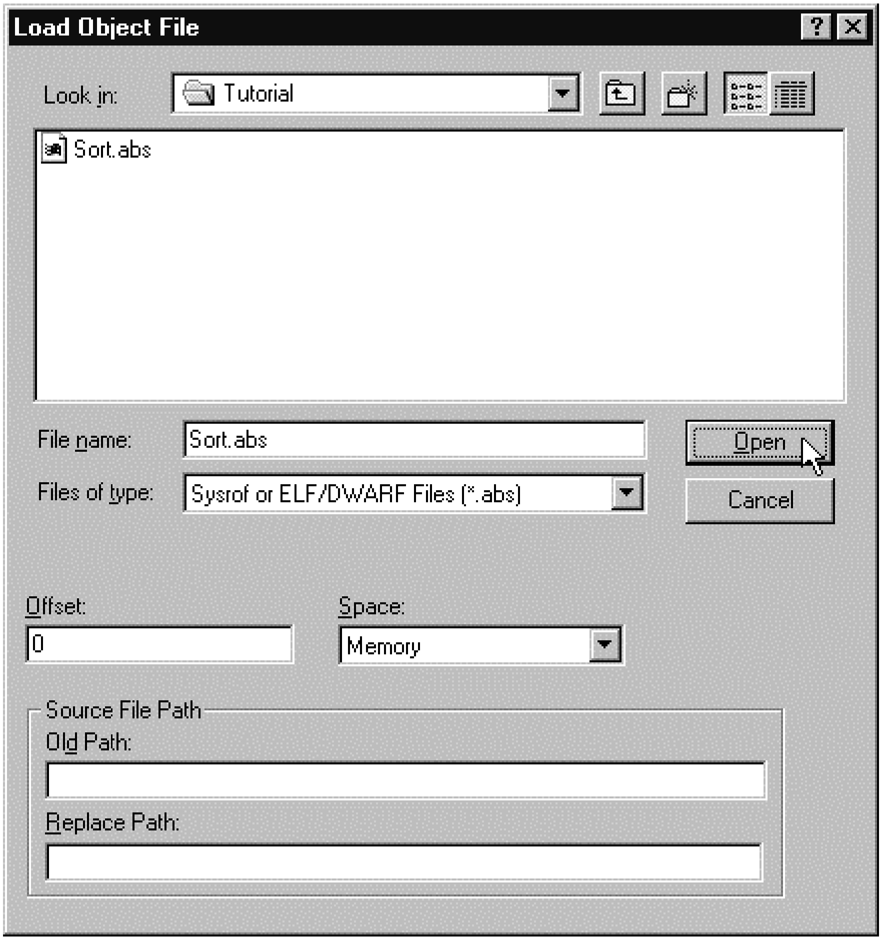
- Edit the **[Start address]** and **[End address]** fields to H'00000000 and H'00003FFF, respectively, set the **[Access type]** as **[Read/Write]**, and click the **[OK]** button.

  The **Memory Map** dialog box will now show the modified ranges.

- Click the **[Close]** button to close the dialog box.

4.5 Downloading the Tutorial Program

- To open the **Load Object File** dialog box, choose **[Load Program...]** from the **[File]** menu, select the file `sort.abs`, and click the **[OK]** button.



Load Object File

Look in: Tutorial

Sort.abs

File name: Sort.abs

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

Offset: 0

Space: Memory

Source File Path

Old Path:

Replace Path:

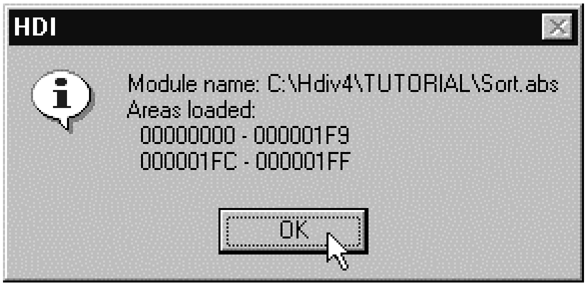
Figure 4.6 Load Object File Dialog Box

The sample program is compiled with `C:\HDI\TUTORIAL`. When using a directory other than `C:\HDI\TUTORIAL`, set as follows:

[Old Path]	C:\HDI\TUTORIAL
[Replace Path]	Path for the directory where the sample program is stored

- Click the [Open] button

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


HDI

```
Module name: C:\Hdiv4\TUTORIAL\Sort.abs
Areas loaded:
  00000000 - 000001F9
  000001FC - 000001FF
```

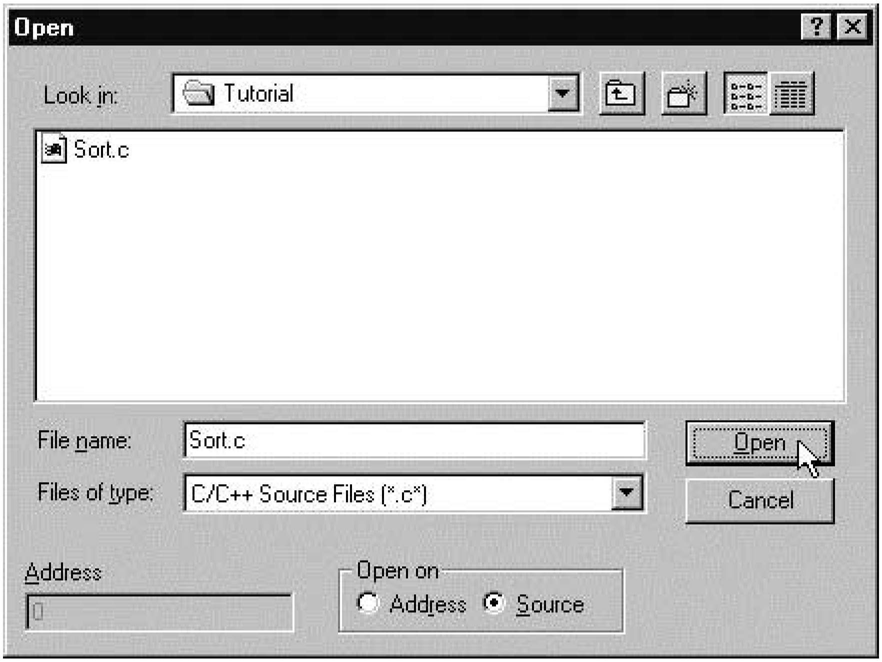
Figure 4.7 HDI Dialog Box

- Click the **[OK]** button to continue.

4.6 Displaying the Source Program

The HDI allows you to debug a program at the source level, so that you can see a listing of the C program alongside the machine code as you debug. To do this, you need to read the C source file that corresponds to the object file.

- Choose **[Program Window...]** from the **[View]** menu.
- You will be prompted for the C source file that corresponds to the object file you have loaded.



Look in: Tutorial

Sort.c

File name: Sort.c

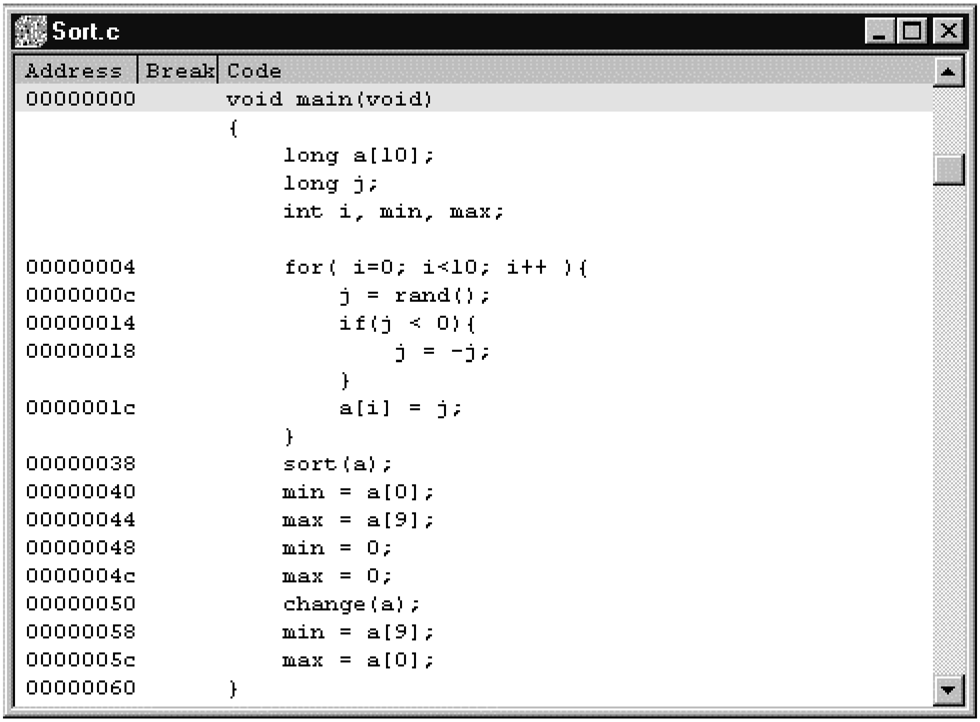
Files of type: C/C++ Source Files (*.c*)

Address: 0

Open on: ( ) Address [✓ Source]

Figure 4.8 Open Program Window Dialog Box

- Select **[sort.c]** and click the **[OK]** button to display the **Program** window.



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

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

Figure 4.9   Program Window (Displaying the Source Program)

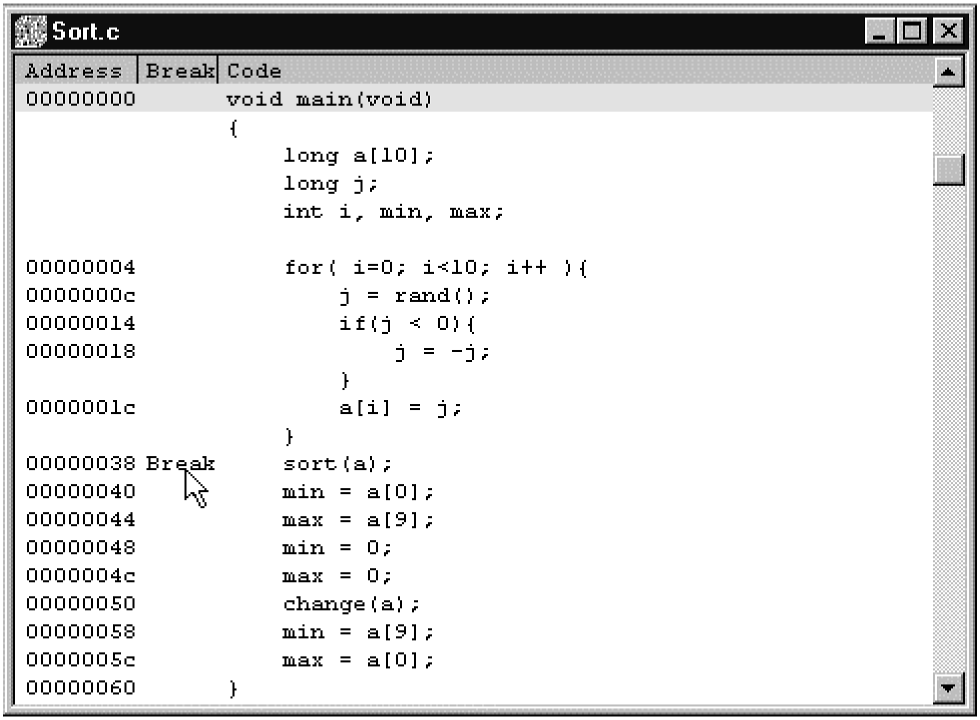
- If necessary, choose the **[Font]** option from the **[Customise]** submenu on the **[Setup]** menu to choose a font and size suitable for your computer.

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

4.7 Setting a PC Breakpoint

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

- Double-click in the **[Break]** column on the line containing the `sort` function call.



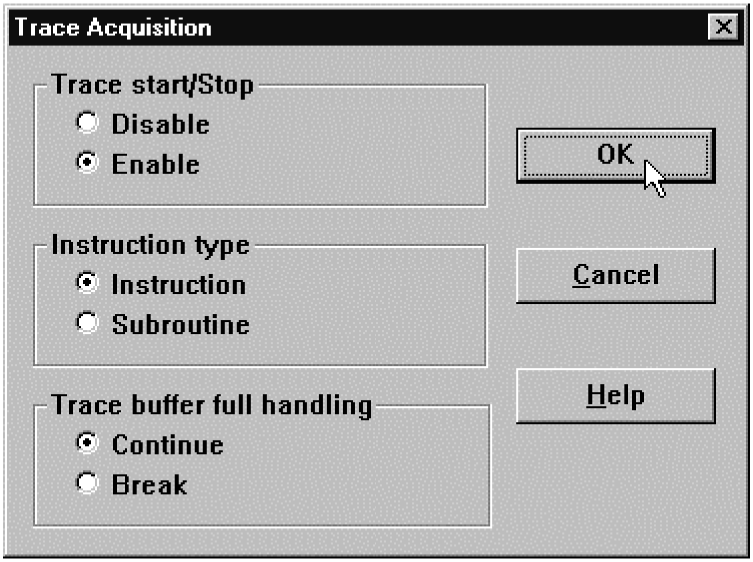
```
Address  Break Code
00000000       void main(void)
               {
                   long a[10];
                   long j;
                   int i, min, max;

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

Figure 4.10 Program Window (Setting the Breakpoint)

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

4.8 Setting Trace Information Acquisition Conditions

- Choose **[Trace Window]** from the **[View]** menu to open the **Trace** window.
  Clicking the **[Acquisition]** button in the **Trace** window displays the following **Trace Acquisition** dialog box.


Trace Acquisition

Trace start/Stop
( ) Disable
(●) Enable

Instruction type
(●) Instruction
( ) Subroutine

Trace buffer full handling
(●) Continue
( ) Break

Figure 4.11 Trace Acquisition Dialog Box

- Set **[Trace start/Stop]** to **[Enable]** in the **Trace Acquisition** dialog box, and click the **[OK]** button to make the trace information acquisition effective.

4.9 Setting Performance Analysis

- Choose **[Performance Analysis Window]** from the **[View]** menu to open the **Performance Analysis** dialog box.

  Clicking the **[Add...]** button in the **Performance Analysis** dialog box displays the following **Performance Option** dialog box.

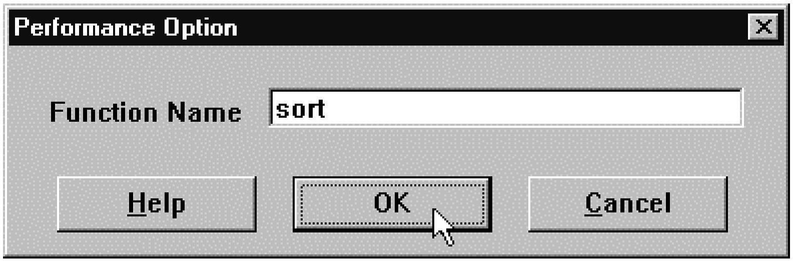
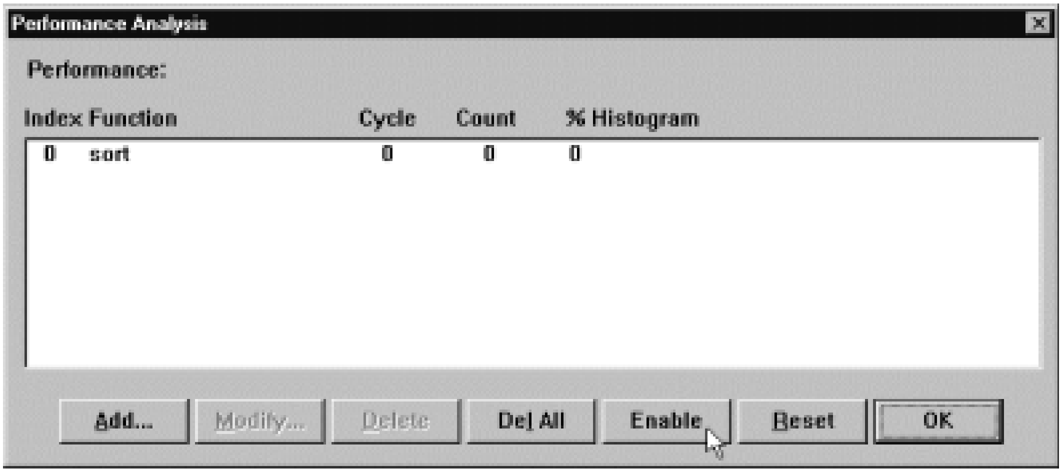


Figure 4.12 Performance Option Dialog Box

- Set **[Function Name]** as `sort` in the **Performance Option** dialog box, and click the **[OK]** button.



Index	Function	Cycle	Count	% Histogram
0	sort	0	0	0

Figure 4.13 Performance Analysis Dialog Box (Setting)

- Confirm that `sort` is set to **[Function]** in the **Performance Analysis** dialog box, and click the **[Enable]** button to make the performance analysis information acquisition effective.

4.10 Setting the Stack Pointer

Set the stack pointer to run the program.

- Choose **[Register Window]** from the **[View]** menu to open the **Registers** window.
- Double-click **[R15]** in the **Registers** window to modify the value of the stack pointer R15.

The following dialog box enables the value to be modified.

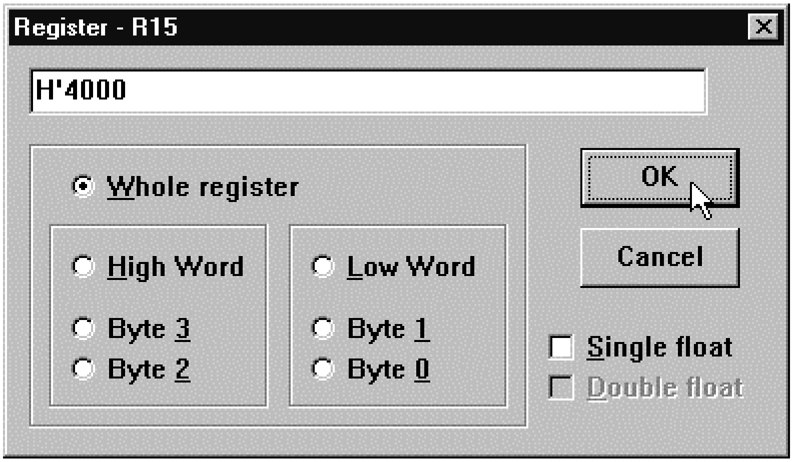


Figure 4.14 Register Dialog Box

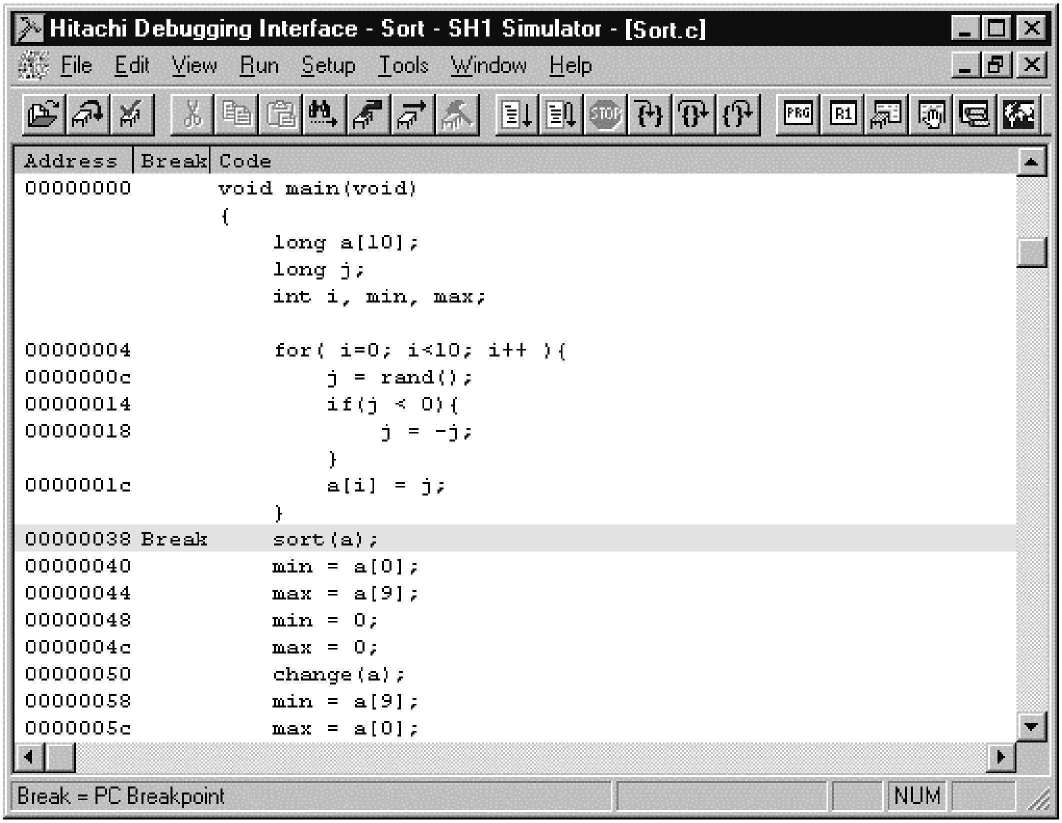
- Set `H'4000` for the value of the stack pointer in this sample program, and click the **[OK]** button.

4.11 Executing the Program

- To run the program, choose **[Go]** from the **[Run]** menu, or click the **[Go]** button on the toolbar.

Figure 4.15 Go Button

The program will be executed up to the breakpoint you inserted, and a statement will be highlighted in the **Program** window to show that the program has halted, with the message Break=PC Breakpoint in the status bar.


Hitachi Debugging Interface - Sort - SH1 Simulator - [Sort.c]
File Edit View Run Setup Tools Window Help
Address Break Code

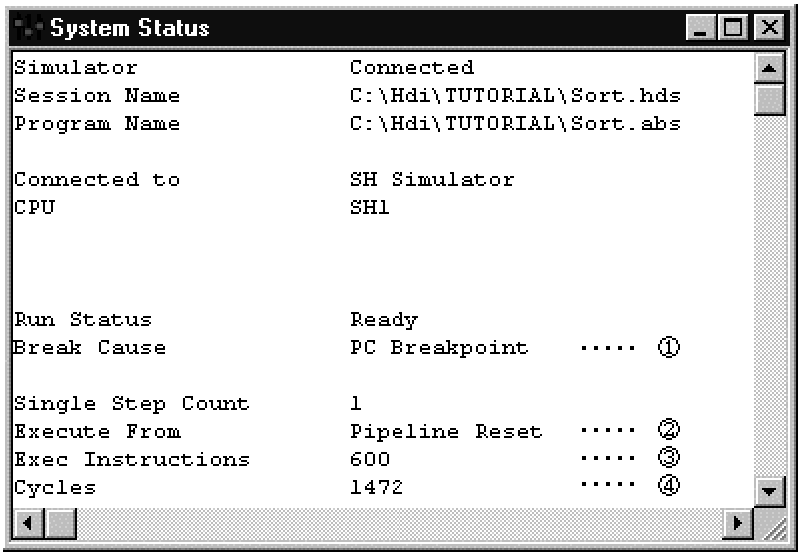
```
00000000          void main(void)
                  {
                      long a[10];
                      long j;
                      int i, min, max;

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

Break = PC Breakpoint   NUM

Figure 4.16 Program Window (Break Status)

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

- Choose **[Status Window]** from the **[View]** menu.



```
Simulator                  Connected
Session Name               C:\Hdi\TUTORIAL\Sort.hds
Program Name               C:\Hdi\TUTORIAL\Sort.abs

Connected to               SH Simulator
CPU                        SH1



Run Status                 Ready
Break Cause                PC Breakpoint     .....  ①

Single Step Count          1
Execute From               Pipeline Reset    .....  ②
Exec Instructions          600               .....  ③
Cycles                     1472              .....  ④
```

Figure 4.17   System Status Window

This shows the following execution results:

(1) The cause of the break was a PC Break.

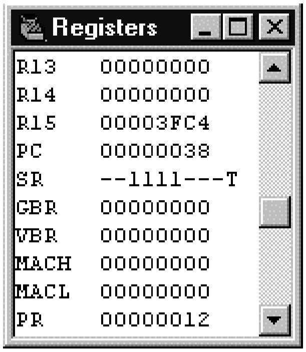
(2) Execution was from the pipeline reset.

(3) The number of instructions executed by the GO command was 600.

(4) The number of cycles executed from the pipeline reset was 1472.

You can also see the value of the registers in the **Registers** window.

- Choose **[Register Window]** from the **[View]** menu.



```
R13    00000000
R14    00000000
R15    00003FC4
PC     00000038
SR     --1111---T
GBR    00000000
VBR    00000000
MACH   00000000
MACL   00000000
PR     00000012
```

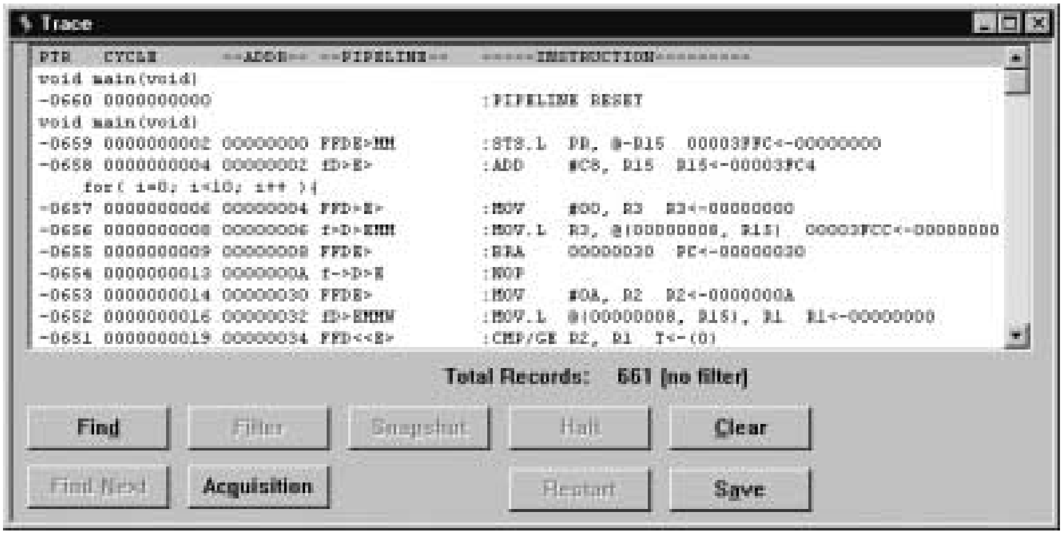
Figure 4.18   Registers Window

You can see the value of each register at the program stop.

4.12 Using the Trace Buffer

The trace buffer allows us to look back over previous execution cycles to see what accesses took place.

- Open the **Trace** window by choosing **[Trace Window]** from the **[View]** menu. Scroll up the window so that you can see the first few cycles.
- If necessary, choose the **[Font]** option from the **[Customise]** submenu on the **[Setup]** menu to choose a font and size suitable for your computer.



```
PTR   CYCLE       --ADDR-- --PIPELINE--     -----INSTRUCTION---------
void main(void)
-0660 0000000000                            :PIPELINE RESET
void main(void)
-0659 0000000002 00000000 FFDE>MM           :STS.L   PR, @-R15   00003FFC<-00000000
-0658 0000000004 00000002 fD>E>             :ADD     #C8, R15   R15<-00003FC4
    for( i=0; i<10; i++ ){
-0657 0000000006 00000004 FFD>E>            :MOV     #00, R3   R3<-00000000
-0656 0000000008 00000006 f>D>EMM           :MOV.L   R3, @(00000008, R15)   00003FCC<-00000000
-0655 0000000009 00000008 FFDE>             :BRA     00000030   PC<-00000030
-0654 0000000013 0000000A f->D>E            :NOP
-0653 0000000014 00000030 FFDE>             :MOV     #0A, R2   R2<-0000000A
-0652 0000000016 00000032 fD>EMMW           :MOV.L   @(00000008, R15), R1   R1<-00000000
-0651 0000000019 00000034 FFD<<E>           :CMP/GE  R2, R1   T<-(0)
```

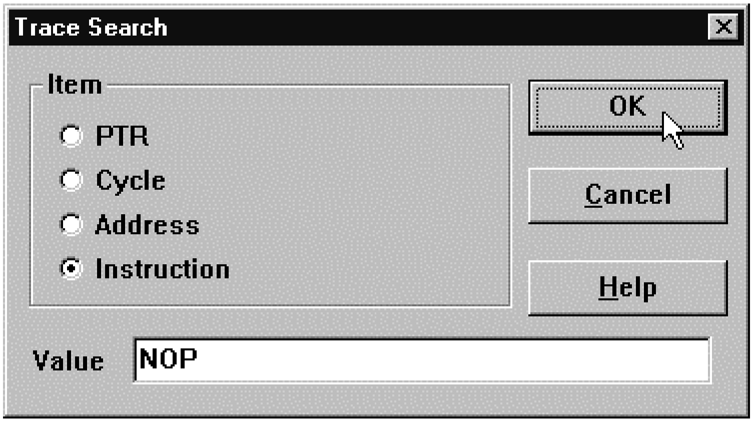
Total Records: 661 [no filter]

Figure 4.19 Trace Window (Displaying Trace Information)

You can see that the `main` function is executed from cycle 0000000000 (pipeline reset) in the trace display shown in figure 4.19.

4.13 Trace Search

Click **[Find]** in the **Trace** window to open the **Trace Search** dialog box.

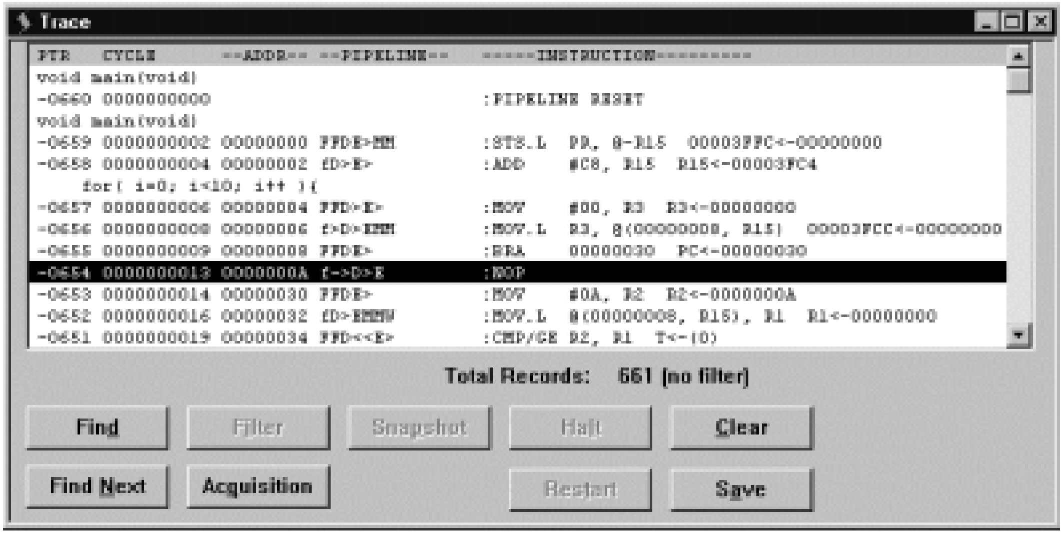


Item: ( ) PTR ( ) Cycle ( ) Address (•) Instruction

Value: NOP

Figure 4.20 Trace Search Dialog Box

A trace search is executed by specifying the search item **[Item]** and search contents **[Value]** then clicking the **[OK]** button. When the corresponding trace information is found, the first line of the information is highlighted. When continuing trace search with the same contents, click the **[Find Next]** button. The next corresponding line is highlighted.



```
PTR    CYCLE       --ADDR-- --PIPELINE--    ------INSTRUCTION----------
void main(void)
-0660 0000000000                            :PIPELINE RESET
void main(void)
-0659 0000000002 00000000 FFDE>MM           :STS.L   PR, @-R15   00003FFC<-00000000
-0658 0000000004 00000002 fD>E>             :ADD     #C8, R15   R15<-00003FC4
     for( i=0; i<10; i++ ){
-0657 0000000006 00000004 FFD>E>            :MOV     #00, R3   R3<-00000000
-0656 0000000008 00000006 f>D>EMM           :MOV.L   R3, @(00000008, R15)   00003FCC<-00000000
-0655 0000000009 00000008 FFDE>             :BRA     00000030   PC<-00000030
-0654 0000000013 0000000A f->D>E            :NOP
-0653 0000000014 00000030 FFDE>             :MOV     #0A, R2   R2<-0000000A
-0652 0000000016 00000032 fD>EMMW           :MOV.L   @(00000008, R15), R1   R1<-00000000
-0651 0000000019 00000034 FFD<<E>           :CMP/GE  R2, R1   T<-(0)
```

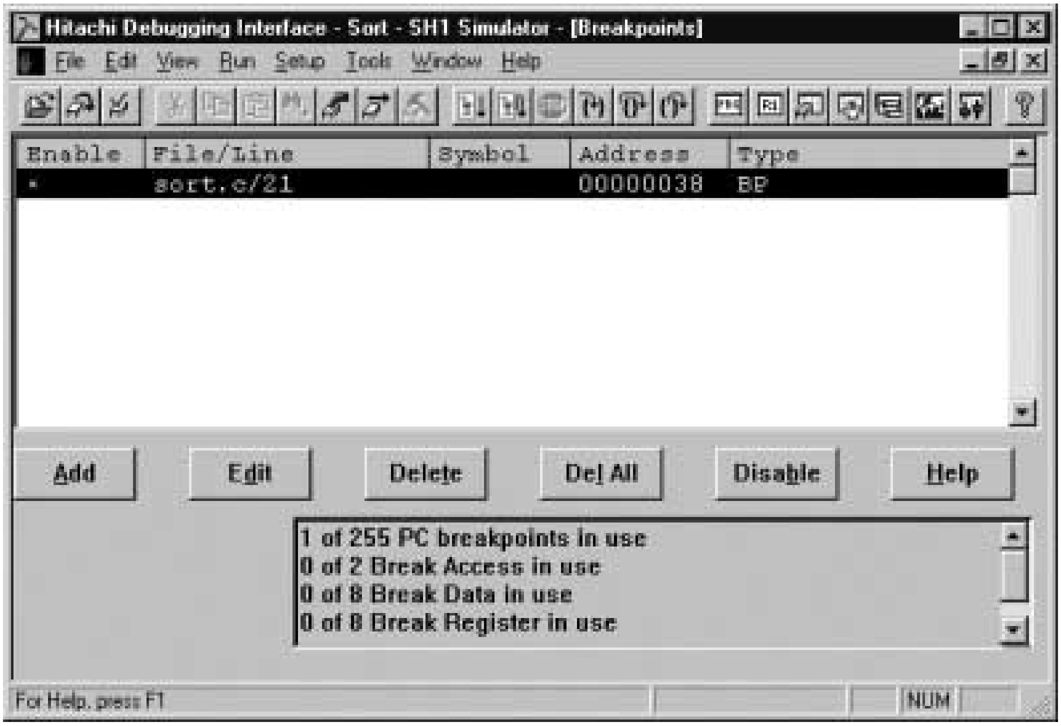
Total Records: 661 [no filter]

Figure 4.21 Trace Window (Search Results)

4.14 Reviewing Breakpoints

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

- Choose **[Breakpoint Window]** from the **[View]** menu.



Hitachi Debugging Interface - Sort - SH1 Simulator - [Breakpoints]

```
Enable  File/Line     Symbol   Address   Type
*       sort.c/21              00000038  BP
```

```
1 of 255 PC breakpoints in use
0 of 2 Break Access in use
0 of 8 Break Data in use
0 of 8 Break Register in use
```

Figure 4.22 Breakpoints Window

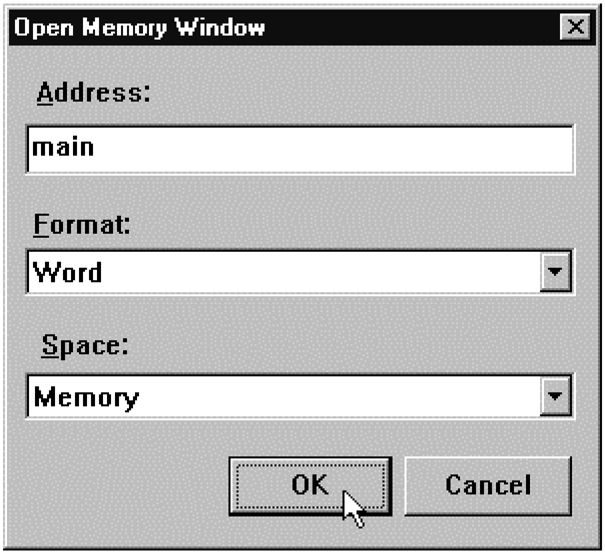
The **Breakpoints** window also allows you to enable and disable breakpoints, define new breakpoints, and delete breakpoints.

- Close the **Breakpoints** window.

4.15 Viewing Memory

You can view the contents of a memory block in the **Memory** window. For example, to view the memory corresponding to array `main` in word size:

- Choose **[Memory Window…]** from the **[View]** menu, enter `main` in the **[Address]** field, and set **[Format]** as `Word`.



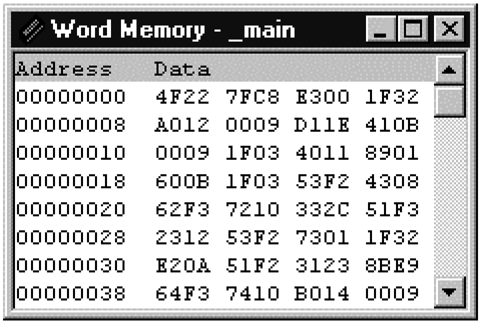
Address: main

Format: Word

Space: Memory

Figure 4.23 Open Memory Window Dialog Box

- Click the **[OK]** button to open the **Word Memory** window showing the specified area of memory.



```
Address    Data
00000000   4F22 7FC8 E300 1F32
00000008   A012 0009 D11E 410B
00000010   0009 1F03 4011 8901
00000018   600B 1F03 53F2 4308
00000020   62F3 7210 332C 51F3
00000028   2312 53F2 7301 1F32
00000030   E20A 51F2 3123 8BE9
00000038   64F3 7410 B014 0009
```

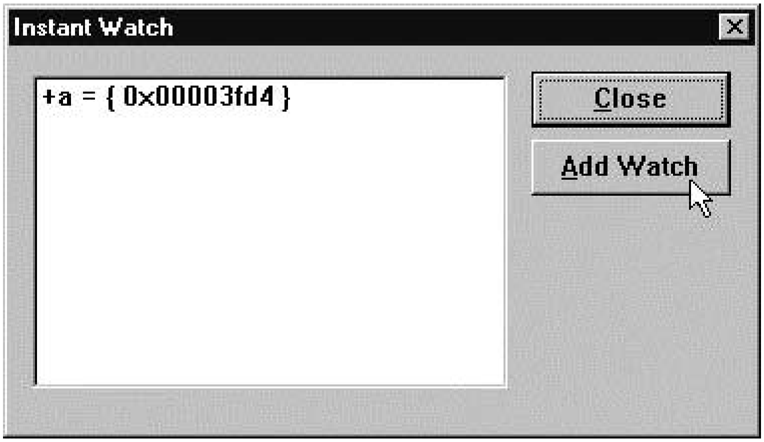
Figure 4.24 Word Memory Window

4.16 Watching Variables

As you step through a program, it is possible to watch the values of variables used in your program and to verify that they change in the way that you expected. For example, set a watch on the long-type array "a" declared at the beginning of the program, by using the following procedure:

- Click to position the cursor to the left of "a" in the **Program** window.
- Click in the **Program** window with the right mouse button to display a pop-up menu, and choose **[Instant Watch]**.

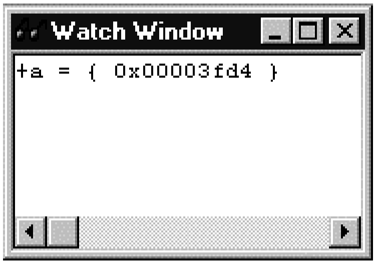
  The following dialog box will be displayed.



```
+a = { 0x00003fd4 }
```

Figure 4.25 Instant Watch Dialog Box

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



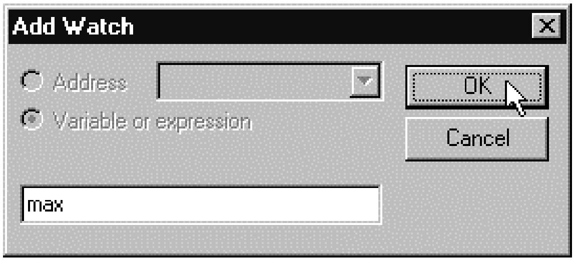
```
+a = { 0x00003fd4 }
```

Figure 4.26 Watch Window (Displaying the Array)

You can also add a variable to the **Watch** window by specifying its name.

- Press the right mouse button on the **Watch** window and choose **[Add Watch]** from the pop-up menu.

  The following dialog box appears.

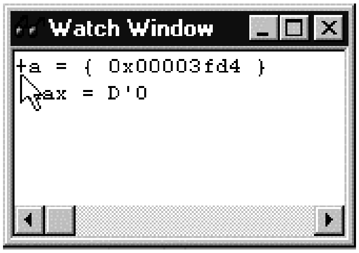


```
max
```

Figure 4.27 Add Watch Dialog Box

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

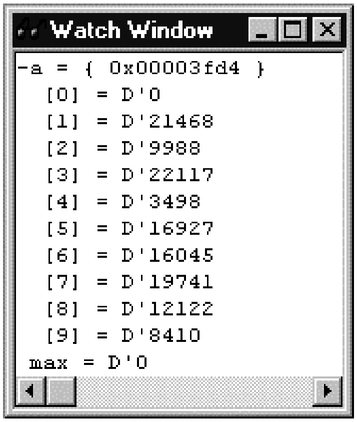
  The **Watch** window will now also show the long-type variable `max`.



```
+a = { 0x00003fd4 }
 max = D'0
```

Figure 4.28 Watch Window (Displaying the Variable)

You can double-click the + symbol to the left of any variable in the **Watch** window to expand the variable and show the individual elements in the array.



```
-a = { 0x00003fd4 }
  [0] = D'0
  [1] = D'21468
  [2] = D'9988
  [3] = D'22117
  [4] = D'3498
  [5] = D'16927
  [6] = D'16045
  [7] = D'19741
  [8] = D'12122
  [9] = D'8410
 max = D'0
```

Figure 4.29 Watch Window (Displaying Array Elements)

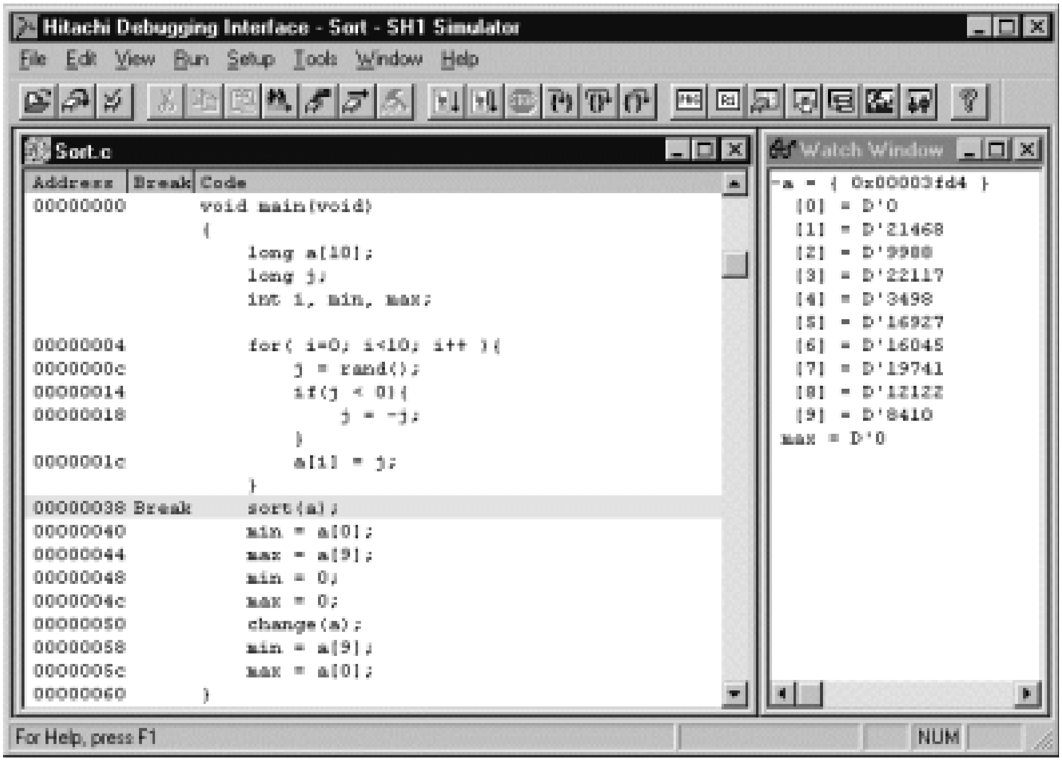
4.17 Stepping Through a Program

The simulator/debugger provides a range of step menu commands that allow efficient program debugging.

Table 4.1 Step Menu Commands

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

To demonstrate program stepping, confirm that the `sort` function statement at address H'00000038 has been executed.


Hitachi Debugging Interface - Sort - SH1 Simulator
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window
For Help, press F1
NUM

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

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

```
-a = { 0x00003fd4 }
  [0] = D'0
  [1] = D'21468
  [2] = D'9988
  [3] = D'22117
  [4] = D'3498
  [5] = D'16927
  [6] = D'16045
  [7] = D'19741
  [8] = D'12122
  [9] = D'8410
 max = D'0
```

Figure 4.30 Program Window (Stepping)

4.17.1 Executing [Step In]

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

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

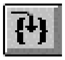
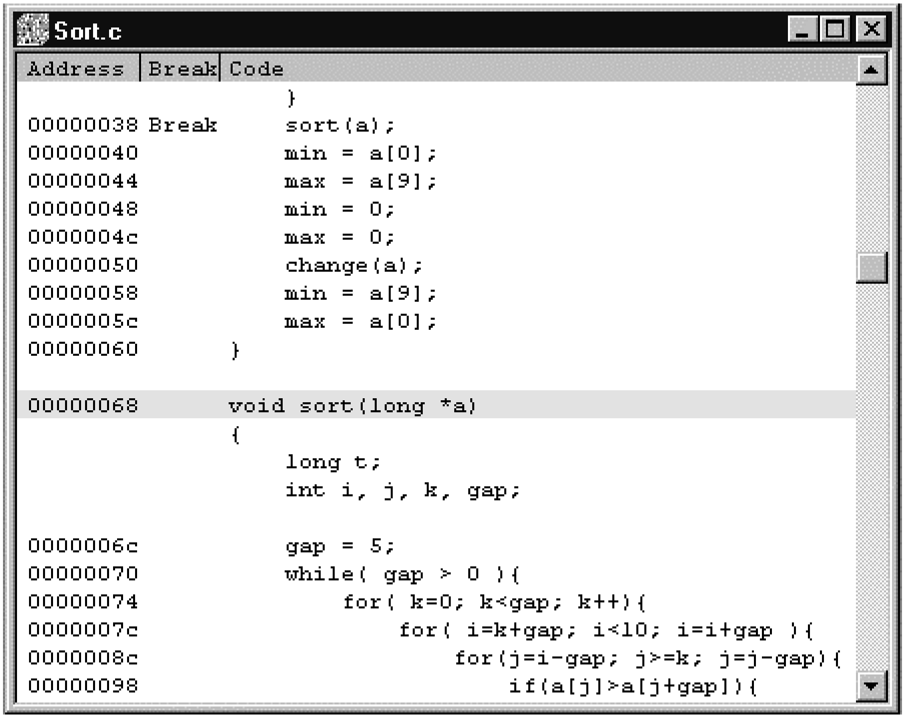


Figure 4.31 Step In Button



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

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

0000006c           gap = 5;
00000070           while( gap > 0 ){
00000074               for( k=0; k<gap; k++){
0000007c                   for( i=k+gap; i<10; i=i+gap ){
0000008c                       for(j=i-gap; j>=k; j=j-gap){
00000098                           if(a[j]>a[j+gap]){
```

Figure 4.32 Program Window (Step In)

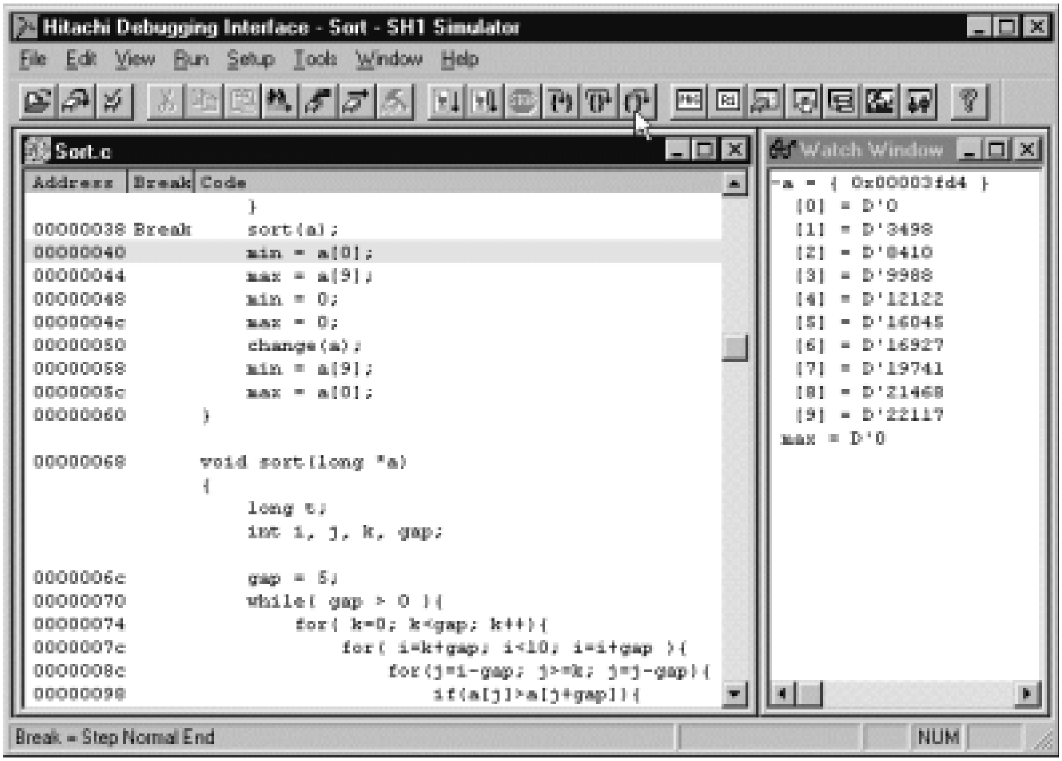
- The highlighted line moves to the first statement of the `sort` function in the **Program** window.

4.17.2 Executing [Step Out]

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

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

Figure 4.33 Step Out Button


Hitachi Debugging Interface - Sort - SH1 Simulator
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window

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

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

0000006c            gap = 5;
00000070            while( gap > 0 ){
00000074                for( k=0; k<gap; k++){
0000007c                    for( i=k+gap; i<10; i=i+gap ){
0000008c                        for(j=i-gap; j>=k; j=j-gap){
00000098                            if(a[j]>a[j+gap]){
```

```
-a = { 0x00003fd4 }
  [0] = D'0
  [1] = D'3498
  [2] = D'8410
  [3] = D'9988
  [4] = D'12122
  [5] = D'16045
  [6] = D'16927
  [7] = D'19741
  [8] = D'21468
  [9] = D'22117
 max = D'0
```

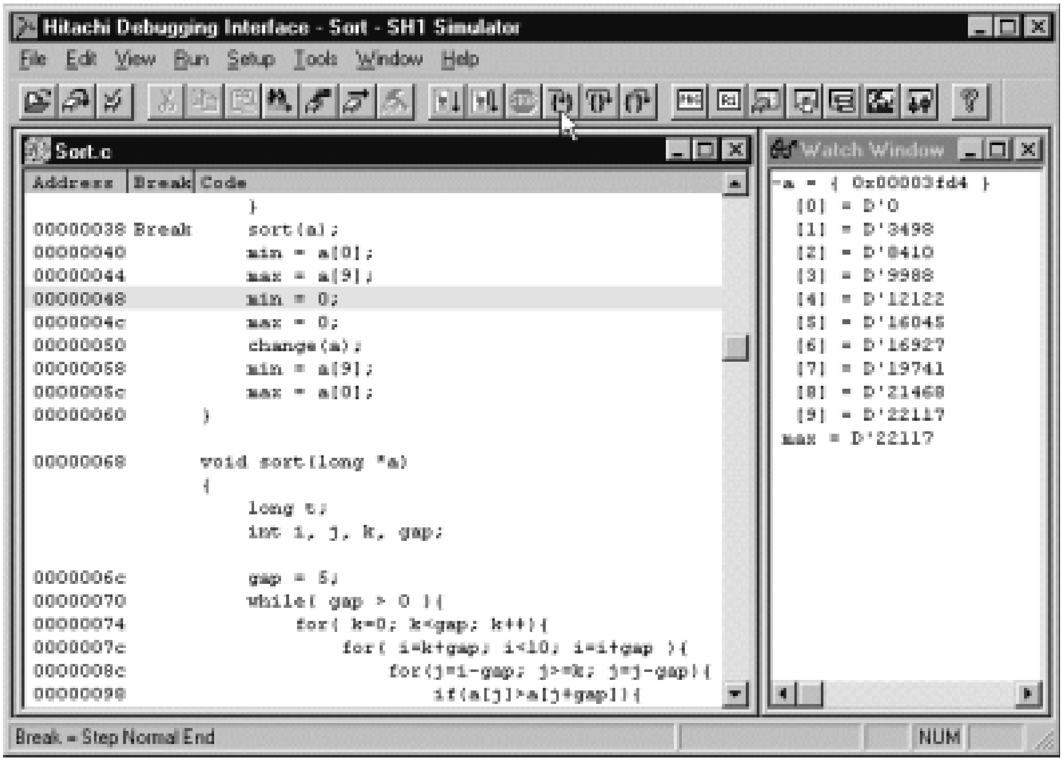
Break = Step Normal End   NUM

Figure 4.34 Program Window (Step Out)

- The data of variable “`a`” displayed in the **Watch** window is sorted in ascending order.

RENESAS

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


Hitachi Debugging Interface - Sort - SH1 Simulator
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window
Break = Step Normal End
NUM

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

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

0000006c              gap = 5;
00000070              while( gap > 0 ){
00000074                  for( k=0; k<gap; k++){
0000007c                      for( i=k+gap; i<10; i=i+gap ){
0000008c                          for(j=i-gap; j>=k; j=j-gap){
00000098                              if(a[j]>a[j+gap]){
```

```
-a = { 0x00003fd4 }
  [0] = D'0
  [1] = D'3498
  [2] = D'8410
  [3] = D'9988
  [4] = D'12122
  [5] = D'16045
  [6] = D'16927
  [7] = D'19741
  [8] = D'21468
  [9] = D'22117
 max = D'22117
```

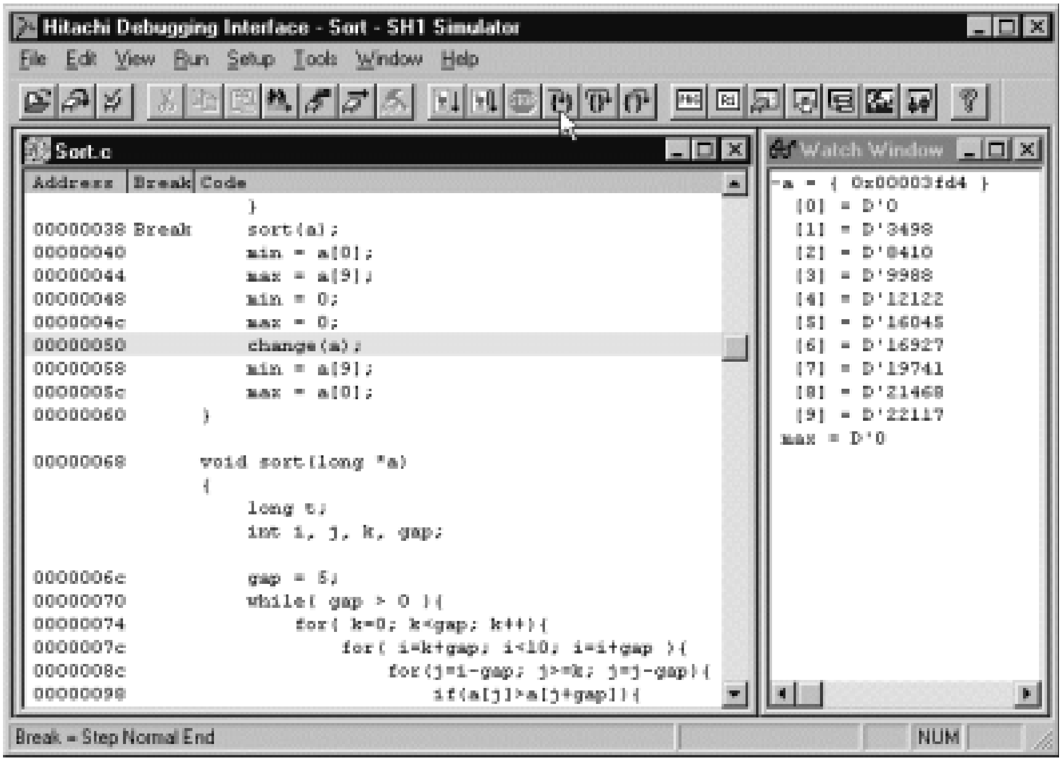
Figure 4.35 Program Window (Step Out –> Step In)

- The value of `max` displayed in the **Watch** window is modified to the maximum data value.

4.17.3 Executing [Step Over]

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

- To demonstrate **[Step Over]** command, execute two steps to reach the `change` function statement.


Hitachi Debugging Interface - Sort - SH1 Simulator
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window
NUM

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

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

0000006c             gap = 5;
00000070             while( gap > 0 ){
00000074                 for( k=0; k<gap; k++){
0000007c                     for( i=k+gap; i<10; i=i+gap ){
0000008c                         for(j=i-gap; j>=k; j=j-gap){
00000098                             if(a[j]>a[j+gap]){
```

```
-a = { 0x00003fd4 }
  [0] = D'0
  [1] = D'3498
  [2] = D'8410
  [3] = D'9988
  [4] = D'12122
  [5] = D'16045
  [6] = D'16927
  [7] = D'19741
  [8] = D'21468
  [9] = D'22117
 max = D'0
```

```
Break = Step Normal End
```

Figure 4.36 Program Window (Before Step Over Execution)

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

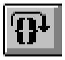
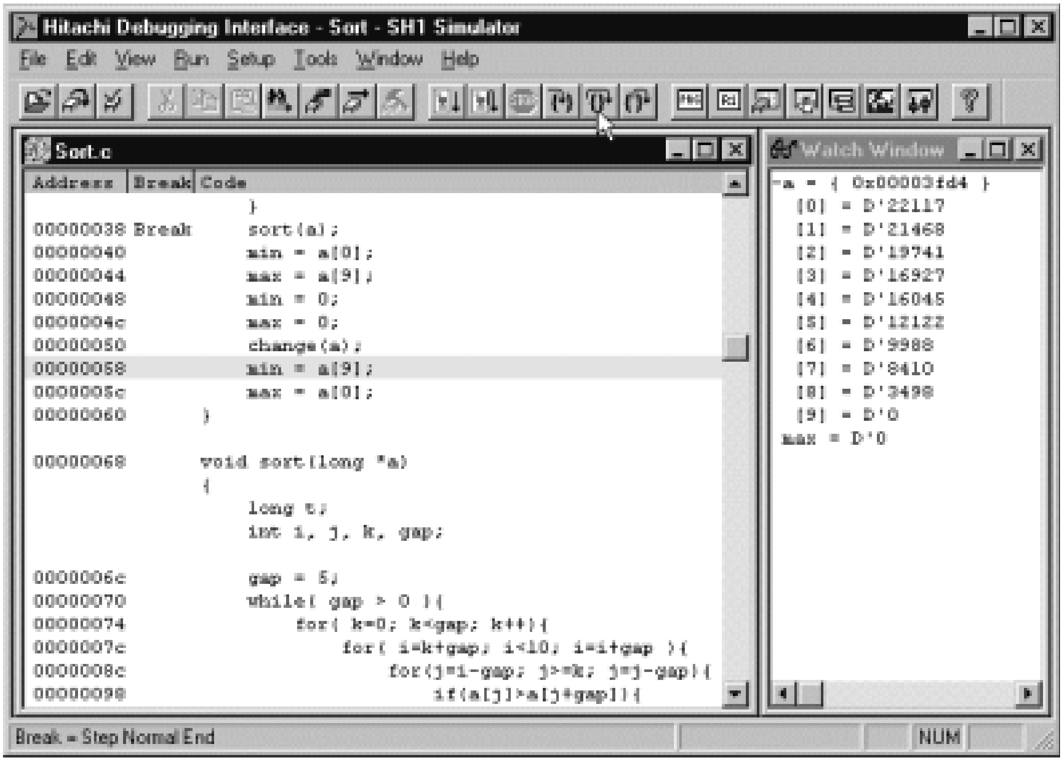


Figure 4.37 Step Over Button


Hitachi Debugging Interface - Sort - SH1 Simulator
File Edit View Run Setup Tools Window Help
Sort.c
Watch Window

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

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

0000006c                gap = 5;
00000070                while( gap > 0 ){
00000074                    for( k=0; k<gap; k++){
0000007c                        for( i=k+gap; i<10; i=i+gap ){
0000008c                            for(j=i-gap; j>=k; j=j-gap){
00000098                                if(a[j]>a[j+gap]){
```

```
-a = { 0x00003fd4 }
  [0] = D'22117
  [1] = D'21468
  [2] = D'19741
  [3] = D'16927
  [4] = D'16045
  [5] = D'12122
  [6] = D'9988
  [7] = D'8410
  [8] = D'3498
  [9] = D'0
 max = D'0
```

```
Break = Step Normal End                NUM
```

Figure 4.38 Program Window (Step Over)

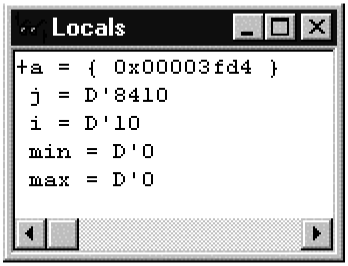
When the last statement of the `change` function is executed, the data of variable “`a`”, which is displayed in the **Watch** window, is sorted in descending order.

RENESAS

4.18 Displaying Local Variables

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

- Open the **Locals** window by choosing **[Local Variable Window]** from the **[View]** menu.
  Initially, the **Locals** window is empty because local variables have not yet been declared.
- Choose **[Step In]** from the **[Run]** menu to execute a single step.
  The **Locals** window will now show the local variables and their values.



```
+a = { 0x00003fd4 }
 j = D'8410
 i = D'10
 min = D'0
 max = D'0
```

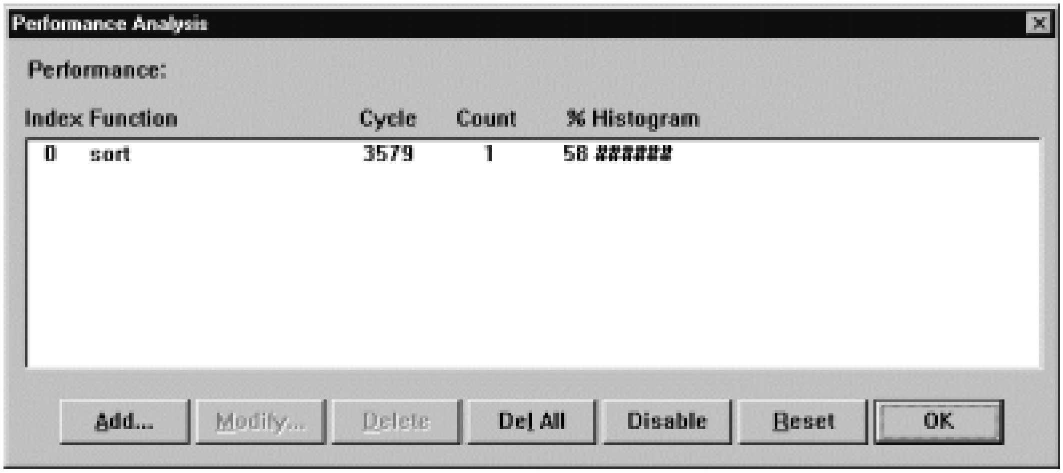
Figure 4.39 Locals Window

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

4.19 Reviewing the Performance Analysis

You can see the performance analysis results of functions in the **Performance Analysis** dialog box.

- Choose **[Performance Analysis Window]** from the **[View]** menu.



Index	Function	Cycle	Count	% Histogram
0	sort	3579	1	58 ######

Figure 4.40 Performance Analysis Dialog Box (Viewing)

The **Performance Analysis** dialog box allows you to enable and disable the performance analysis, define new functions for performance analysis, and delete functions.

- Close the **Performance Analysis** dialog box.

4.20 Saving the Session

Before exiting, it is a good practice to save your session, so that you can resume with the same conditions in your next debugging session.

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

Section 5 Windows and Dialog Boxes

Table 5.1 lists the menu bars and pull-down menus of the Hitachi Debugging Interface window. Note, however, that **[Localized Dump Window]** is supported only in the Japanese environment, and **[TLB Window]**, **[Cache Window]**, and**[Control Register Window]** are supported only by the SH-3, SH-3E, and SH-4 series. In table 5.1, the O mark shows that the menu is described in the Hitachi Debugging Interface User's Manual, and the number indicates the corresponding section(s) in this manual.

Table 5.1 HDI Window Menus and Corresponding Manuals

Menu Bar	Pull-Down Menu	HDI Manual	Sections in This Manual
File menu	Load Program...	O	2.5, 4.5
	Save Memory...	O	—
	Verify Memory...	O	—
	Save Session	O	4.20
	Load Session...	O	—
	Save Session As...	O	—
	Initialise	O	—
	Exit	O	4.20
Edit Menu	Cut	O	—
	Copy	O	—
	Paste	O	—
	Find...	O	2.13
	Set Line...	O	—
	Fill Memory...	O	2.13
	Move Memory...	O	2.5
	Test Memory...	O	—
	Update Memory	O	—
View Menu	Toolbar	O	4.3, 4.11, 4.17
	Status Bar	O	3.3, 4.3, 4.11
	Breakpoint Window	O	4.14, 5.1
	Command Line Window	O	—

Table 5.1 HDI Window Menus and Corresponding Manuals (cont)

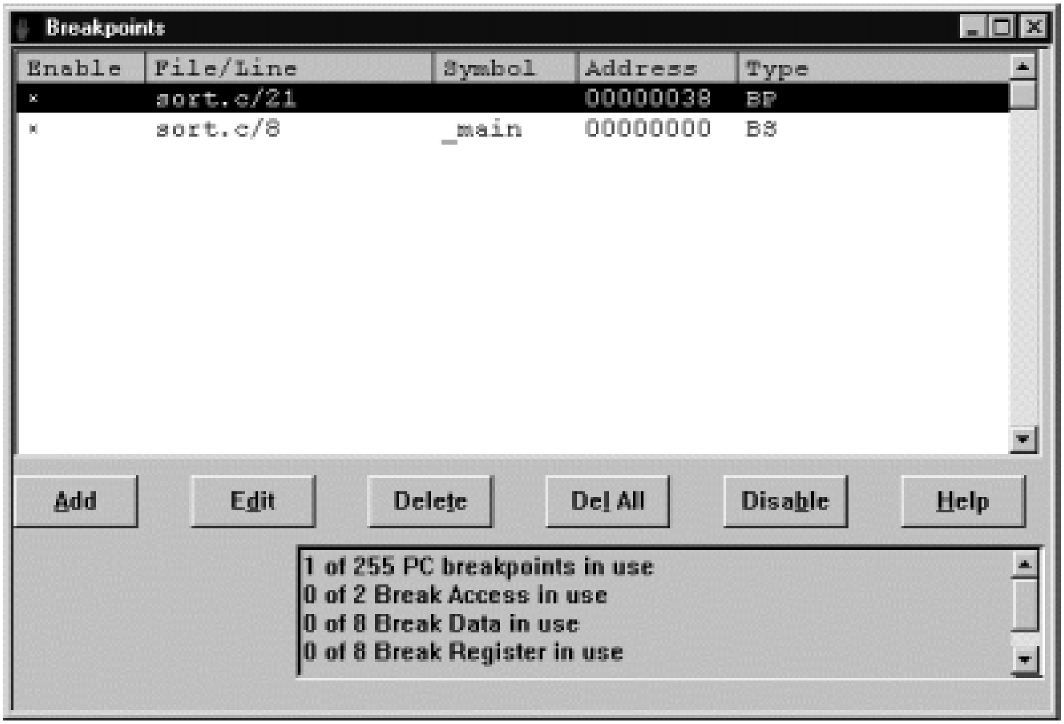
Menu Bar	Pull-Down Menu	HDI Manual	Sections in This Manual
View Menu	I/O Register Window	O	—
	Local Variable Window	O	4.18
	Memory Mapping Window	O	4.4, 5.7
	Memory Window...	O	2.5, 2.13, 4.15
	Performance Analysis Window	O	4.9, 4.19, 5.9
	Program Window...	O	4.6, 4.7, 4.11, 4.16, 4.17
	Register Window	O	2.13, 4.10, 4.11
	Status Window	O	2.5, 4.11
	Text Window	O	—
	Trace Window	O	2.10, 4.8, 4.12, 4.13, 5.11
	Watch Window	O	4.16, 4.17
	TLB Window...	—	2.4, 5.40, 5.43, 5.44, 5.47
	Cache Window...	—	2.5, 5.50, 5.52, 5.55
	Localized Dump Window	O	—
	Simulated I/O Window	—	2.11, 5.57
	Control Register Window	—	2.9, 5.14
	Stack Trace Window	—	2.14, 5.58
Run Menu	Go	O	4.11
	Go Reset	O	—
	Go to Cursor	O	—
	Run...	O	—
	Step In	O	4.17
	Step Over	O	4.17
	Step Out	O	4.17
	Step...	O	4.17
	Halt Program	O	—
	Set PC to Cursor	O	—
	Reset CPU	O	—

Table 5.1 HDI Window Menus and Corresponding Manuals (cont)

Menu Bar	Pull-Down Menu	HDI Manual	Sections in This Manual
Setup Menu	Options...	O	—
	Radix	O	—
	Customise	O	4.6
	Select Platform...	O	3.3, 4.3
	Configure Platform...	O	2.1, 2.2, 2.8, 2.11, 2.12, 2.13, 5.4
	Overlay...	O	—
Tools Menu	Symbols...	O	—
	Evaluate...	O	—
Window Menu	Cascade	O	—
	Tile	O	—
	Arrange Icons	O	—
	Close All	O	—
Help Menu	Index	O	—
	Using Help	O	—
	Search for Help on	O	—
	About HDI	O	—

This section describes windows and dialog boxes displayed by the simulator/debugger of SuperH RISC engine series microprocessors.

5.1 Breakpoints Window



Enable	File/Line	Symbol	Address	Type
×	sort.c/21		00000038	BP
×	sort.c/8	_main	00000000	BS

```
1 of 255 PC breakpoints in use
0 of 2 Break Access in use
0 of 8 Break Data in use
0 of 8 Break Register in use
```

Figure 5.1 Breakpoints Window

This window displays all of the specified breakpoints.

[Enable] Displays whether the breakpoint is enabled or disabled. Breakpoints with X are enabled.

[File/Line] Displays file names and line numbers where breakpoints are specified.

[Symbol] Displays symbols that correspond to breakpoint setting addresses. When no symbol exists, nothing is displayed.

[Address] Displays addresses where breakpoints are specified.

[Type] Displays break types.

BP: PC break
BA: Break access
BD: Break data
BR: Break register (A register name is parenthesized.)
BS: Break sequence

The text box below the buttons displays the current break-setting conditions.

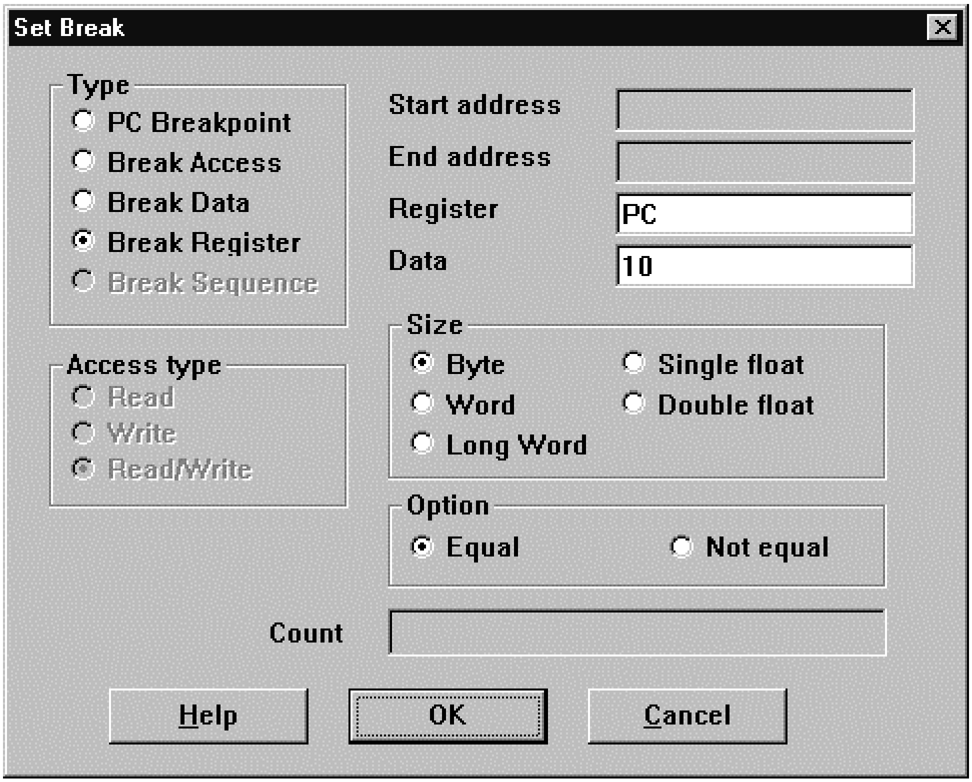
The window buttons have the following features:

[Add]	Sets breakpoints. Clicking this button will open the **Set Break** dialog box and break conditions can be specified.
[Edit]	Edits breakpoints. Select breakpoints to be edited and click this button. The **Set Break** dialog box will open and break conditions can be modified. Note that if a break sequence is selected for editing, the **Break Sequence** dialog box will open.
[Delete]	Deletes specified breakpoints. Select breakpoints to be deleted and click this button.
[Del All]	Deletes all breakpoints.
[Disable] ([Enable])	Reverses the enable/disable status of a breakpoint. Select breakpoints to be modified and click this button.
[Help]	Displays help information.

Clicking anywhere in the view area in the top part of the window with the right mouse button displays the following pop-up menu:

[Help]	Displays help information.
[Add]	Sets breakpoints. Clicking this button will open the **Set Break** dialog box and break conditions can be specified.
[Edit]	Edits breakpoints. Select breakpoints to be edited and click this button. The **Set Break** dialog box will open and break conditions can be changed. Note that if a break sequence is selected for editing, the **Break Sequence** dialog box will open.
[Delete]	Deletes specified breakpoints. Select breakpoints to be deleted and click this button.
[Delete All]	Deletes all breakpoints.
[Disable] ([Enable])	Reverses the enable/disable status of a breakpoint. Select breakpoints to be modified and click this button.

5.2 Set Break Dialog Box


Set Break

Type: PC Breakpoint / Break Access / Break Data / [✓ Break Register] / Break Sequence

Access type: Read / Write / [✓ Read/Write]

Start address:

End address:

Register: PC

Data: 10

Size: [✓ Byte] / Word / Long Word / Single float / Double float

Option: [✓ Equal] / Not equal

Count:

Figure 5.2 Set Break Dialog Box

This dialog box specifies break conditions.

A break type is specified using the radio buttons in the **[Type]** box. Items that can be specified are listed below. Specifying an item and clicking the **[OK]** button can set break conditions.

[PC Breakpoint]	[Start address]	Address where a break occurs
	[Count]	Number of times that a specified instruction is fetched (default: 1)
[Break Access]	[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

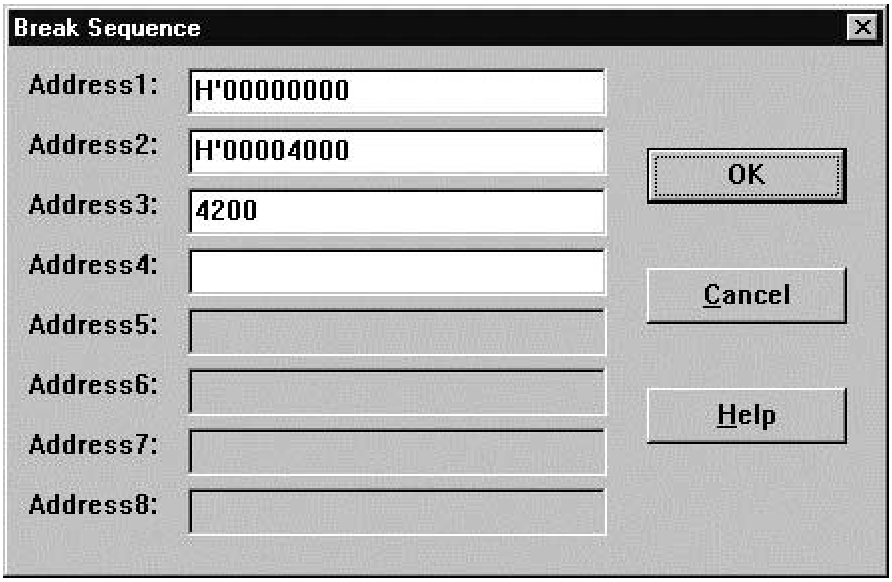
[Break Data]	[Start Address]	Address of memory where a break occurs
	[Data]	Data value that causes a break
	[Size]	Data size
	[Option]	Data match/mismatch
[Break Register]	[Register]	Register name where break conditions are specified
	[Data]	Data value that causes a break (If no data is input, a break occurs whenever data is written to the register)
	[Size]	Data size
	[Option]	Data match/mismatch

Note that when **[Break Sequence]** is selected under **[Type]**, the **Break Sequence** dialog box opens.

When **[PC Breakpoint]** is selected, if an overloaded function or class name including a member function is specified in **[Start Address]**, the **Select Function** dialog box opens. In the dialog box, select a function. For details, refer to the Hitachi Debugging Interface User's Manual.

Clicking the **[Cancel]** button closes the **Set Break** dialog box.

5.3 Break Sequence Dialog Box



Address1: H'00000000
Address2: H'00004000
Address3: 4200
Address4:
Address5:
Address6:
Address7:
Address8:

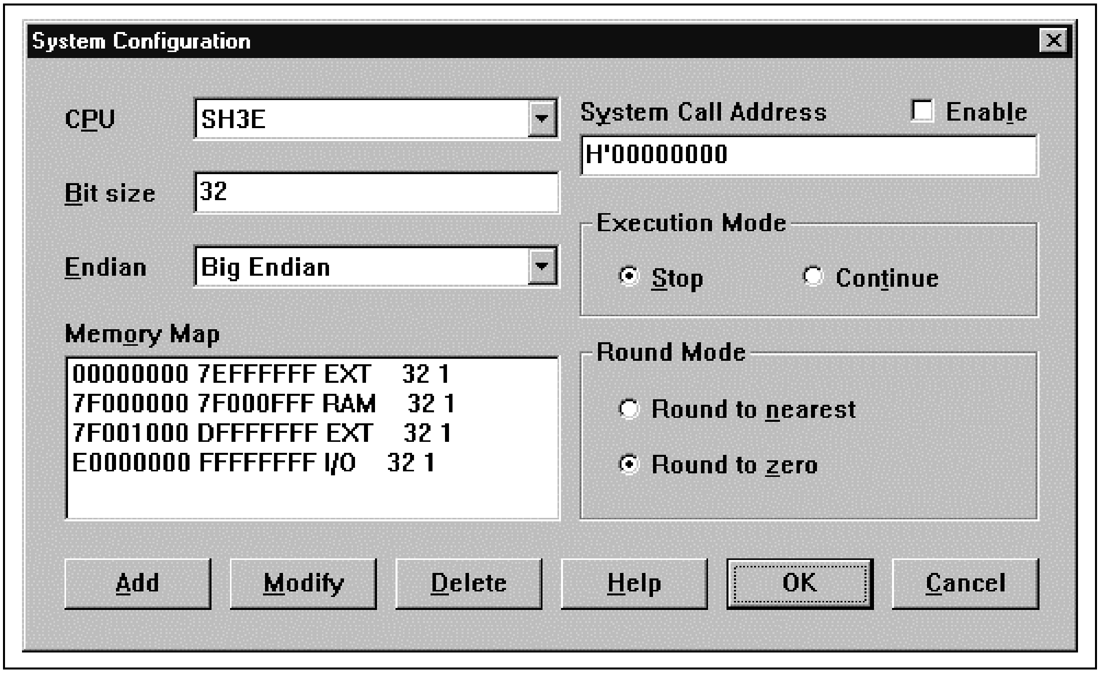
Figure 5.3 Break Sequence Dialog Box

This dialog box specifies the pass addresses as break conditions.

Specify addresses in **[Address1]** to **[Address8]**. Not all eight addresses need to be specified. When an overloaded function or a class name including a member function is specified as a pass address, the **Select Function** dialog box will open; select the function name in the dialog box. For details, refer to the HDI User's Manual.

Clicking the **[OK]** button sets the pass addresses. Clicking the **[Cancel]** button closes this dialog box without adding a new pass address.

5.4 System Configuration Dialog Box


System Configuration

CPU: SH3E

Bit size: 32

Endian: Big Endian

System Call Address: H'00000000 [ ] Enable

Execution Mode: [● Stop] ○ Continue

Round Mode: ○ Round to nearest [● Round to zero]

Memory Map

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

Figure 5.4 System Configuration Dialog Box

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

[CPU] [Bit size] Display the CPU name and bit size. These items cannot be modified.

[Endian] Specifies big endian or little endian (for the SH-1, SH-2, SH-2E, and SH-DSP series, this item is fixed as Big Endian)

[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 when an error occurs.
[Continue]	Continues the simulation when an error occurs.

[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.

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

- SH-1, SH-2, SH-2E, SH-3, and SH-3E Series

  ROM (internal ROM), RAM (internal RAM), EXT (external bus space), I/O (internal I/O area)
- SH-DSP Series

  XROM, YROM (internal ROM), XRAM, YRAM (internal RAM), EXT (external bus space), I/O (internal I/O area)
- SH-4 Series

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

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

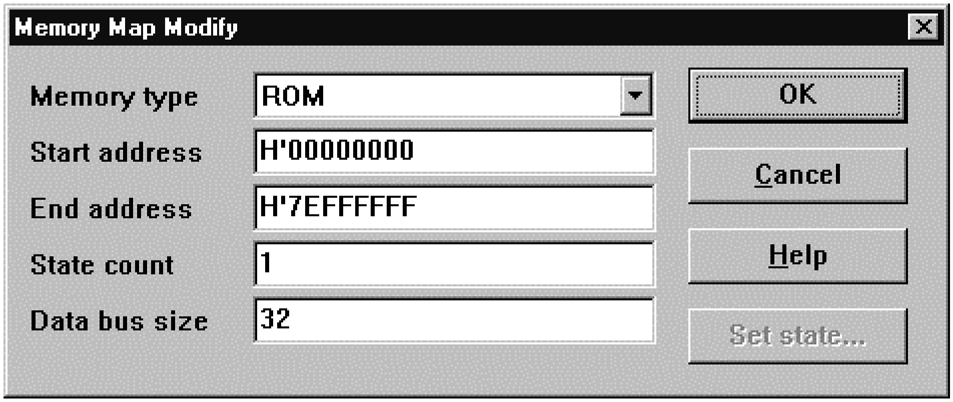
[Add]	Specifies **[Memory Map]** items. Clicking this button opens the **Memory Map Modify** dialog box, and **Memory Map** items can be specified.
[Modify]	Modifies **[Memory Map]** items. Select an item to be modified in the list box and click the **Modify** button. The **Memory Map Modify** dialog 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.

Note: For the SH-4 series, 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.

5.5 Memory Map Modify Dialog Box


Memory Map Modify

Memory type	ROM
Start address	H'00000000
End address	H'7EFFFFFF
State count	1
Data bus size	32

Figure 5.5 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]**, **[End address]**, and **[State count]** cannot be modified. Specify **[Memory type]** and **[Data 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.

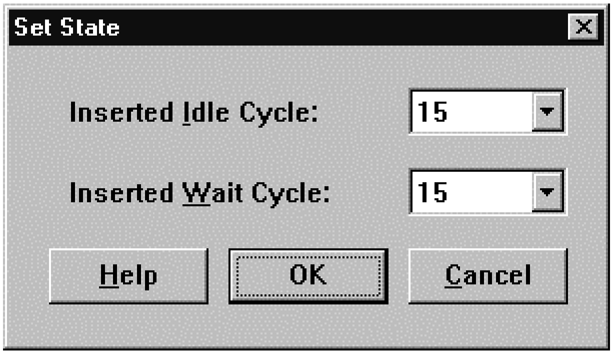
Clicking the **[OK]** button stores the settings. Clicking the **[Cancel]** button closes this dialog box without modifying the settings.

5.6 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.

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:


Set State

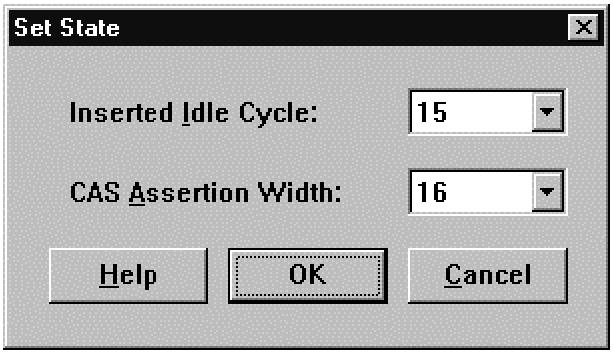
Inserted Idle Cycle: 15

Inserted Wait Cycle: 15

Figure 5.6 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:


Set State

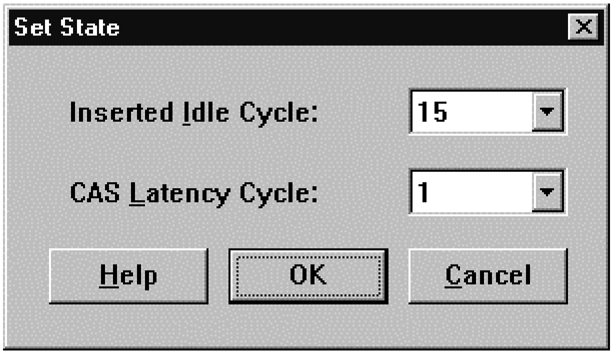
Inserted Idle Cycle: 15

CAS Assertion Width: 16

Figure 5.7 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 period (corresponds to the AnW in WCR2).

SDRAM:


Set State

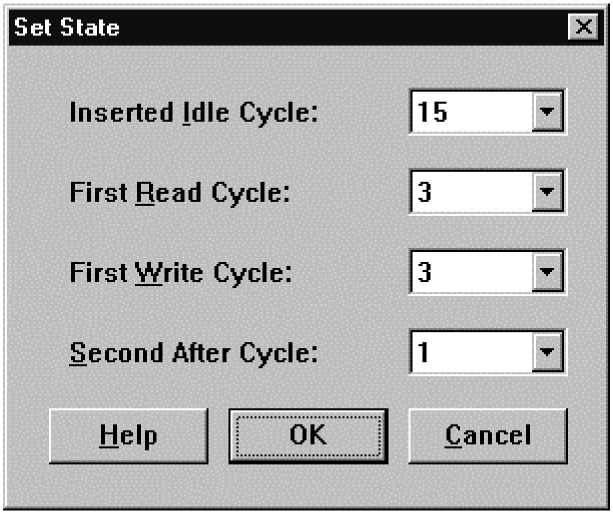
Inserted Idle Cycle: 15

CAS Latency Cycle: 1

Figure 5.8 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:



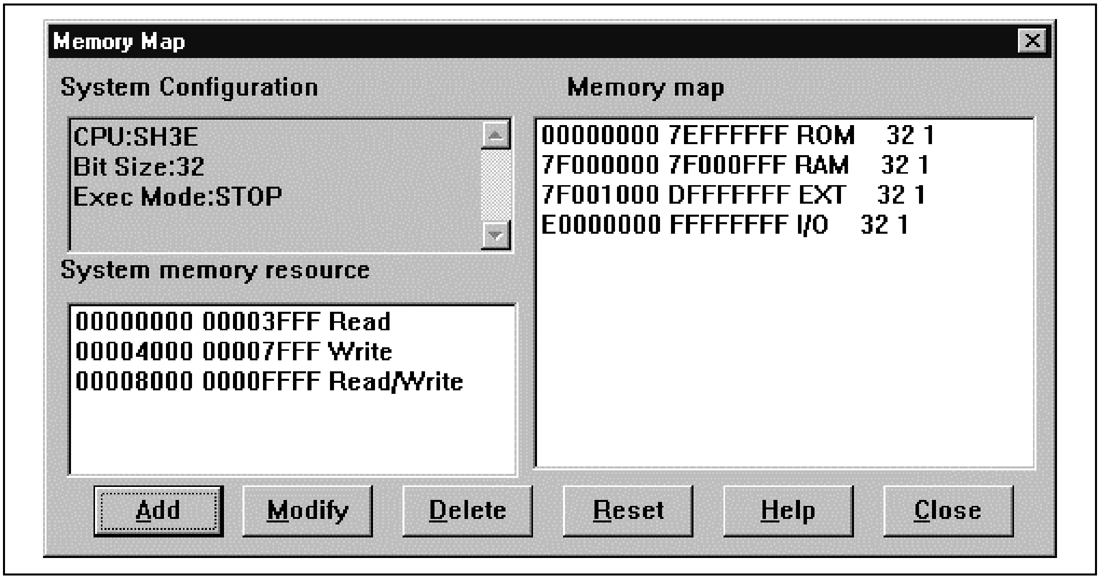
Inserted Idle Cycle: 15
First Read Cycle: 3
First Write Cycle: 3
Second After Cycle: 1

Figure 5.9 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.

5.7 Memory Map Dialog Box



```
CPU:SH3E
Bit Size:32
Exec Mode:STOP
```

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

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

Figure 5.10 Memory Map Dialog Box

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

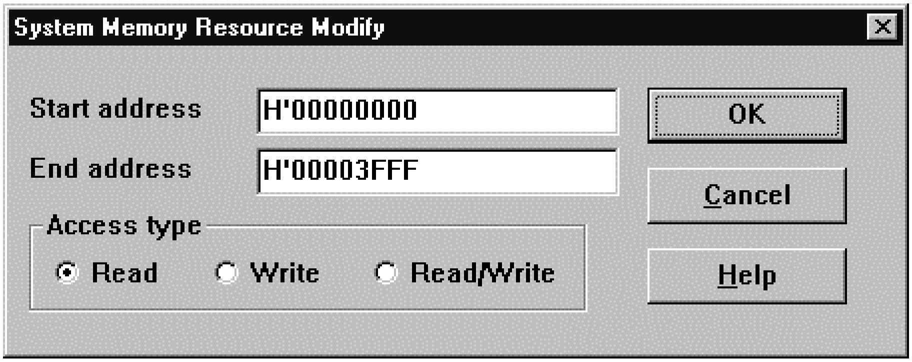
[System Configuration]	Displays the target CPU, bit size, and execution mode of the simulator/debugger.
[System memory resource]	Displays the access type, start address, and end address of the current memory.
[Memory map]	Displays the memory type, start address, end address, data bus width, and access 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 the **Modify** 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 the **Delete** button.

Note that **[Memory map]** and **[System memory resource]** can be reset to the default values using the **[Reset]** button. Clicking the **[Close**] button closes this dialog box.

5.8 System Memory Resource Modify Dialog Box


System Memory Resource Modify

Start address: H'00000000

End address: H'00003FFF

Access type: [✓ Read] Write Read/Write

Figure 5.11 System Memory Resource Modify Dialog Box

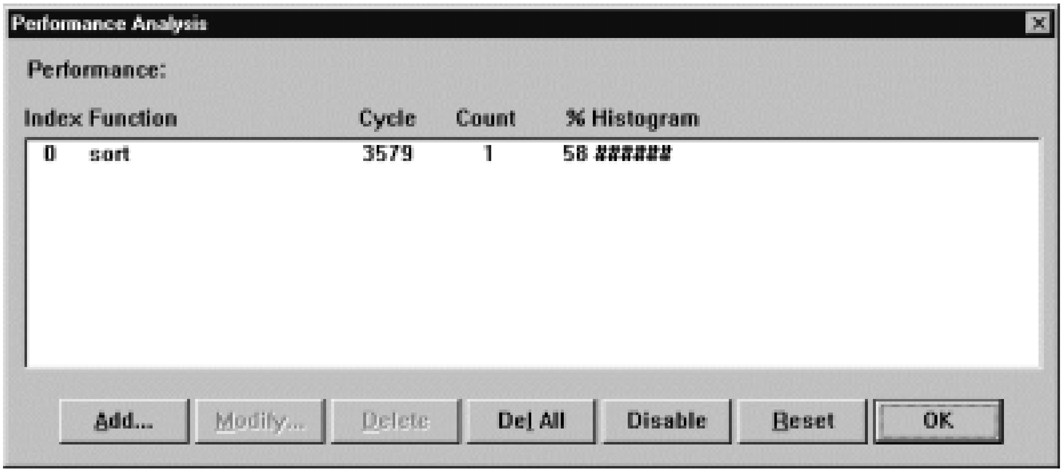
This dialog box specifies or modifies memory resource settings.

[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

Clicking the **[OK]** button after specifying the above items stores the memory resource settings. Clicking the **[Cancel]** button closes this dialog box without modifying the settings.

5.9 Performance Analysis Dialog Box



```
Index Function              Cycle    Count    % Histogram
0     sort                  3579       1      58 ######
```

Figure 5.12 Performance Analysis Dialog Box

This dialog box 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 execution cycles required for the function
[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

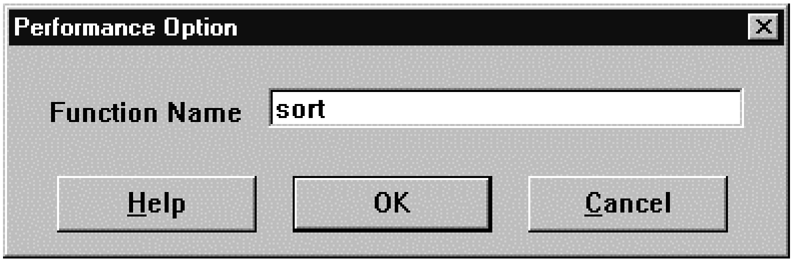
Functions to be evaluated can be added, modified, deleted, or reset using the following buttons. Up to 255 functions can be specified.

[Add...] Adds functions to be evaluated. Clicking this button opens the **Performance Option** dialog box, and functions can be added.

[Modify...]	Modifies function to be evaluated. Select a function to be modified in the list box and click the **[Modify]** button. The **Performance Option** dialog box opens and the function can be modified.
[Delete]	Deletes function to be evaluated. Select a function to be deleted in the list box and click the **[Delete]** button.
[Del All]	Deletes all functions to be evaluated.
[Disable] ([Enable])	Reverses the stop/restart status of the performance analysis.
[Reset]	Resets performance analysis information.

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

5.10 Performance Option Dialog Box



Function Name: sort

Figure 5.13 Performance Option Dialog Box

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

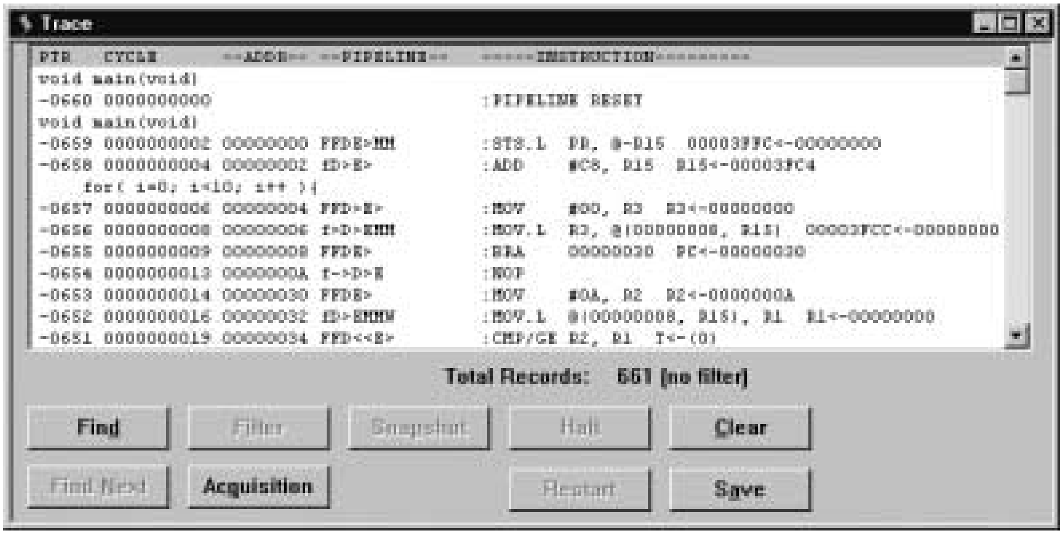
Note that when an overloaded function or a class name including a member function is specified, the **Select Function** dialog box opens. In the dialog box, select a function. For details, refer to the Hitachi Debugging Interface User's Manual.

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

5.11 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 SH-DSP Series:



```
PTR   CYCLE       --ADDR-- --PIPELINE--    -----INSTRUCTION---------
void main(void)
-0660 0000000000                           :PIPELINE RESET
void main(void)
-0659 0000000002 00000000 FFDE>MM          :STS.L  PR, @-R15  00003FFC<-00000000
-0658 0000000004 00000002 fD>E>            :ADD    #C8, R15  R15<-00003FC4
     for( i=0; i<10; i++ ){
-0657 0000000006 00000004 FFD>E>           :MOV    #00, R3  R3<-00000000
-0656 0000000008 00000006 f>D>EMM          :MOV.L  R3, @(00000008, R15)  00003FCC<-00000000
-0655 0000000009 00000008 FFDE>            :BRA    00000030  PC<-00000030
-0654 0000000013 0000000A f->D>E           :NOP
-0653 0000000014 00000030 FFDE>            :MOV    #0A, R2  R2<-0000000A
-0652 0000000016 00000032 fD>EMMW          :MOV.L  @(00000008, R15), R1  R1<-00000000
-0651 0000000019 00000034 FFD<<E>          :CMP/GE R2, R1  T<-(0)
```

Total Records: 661 [no filter]

Figure 5.14 Trace Window (for SH-1, SH-2, SH-2E, and SH-DSP Series)

This window displays the following trace information items:

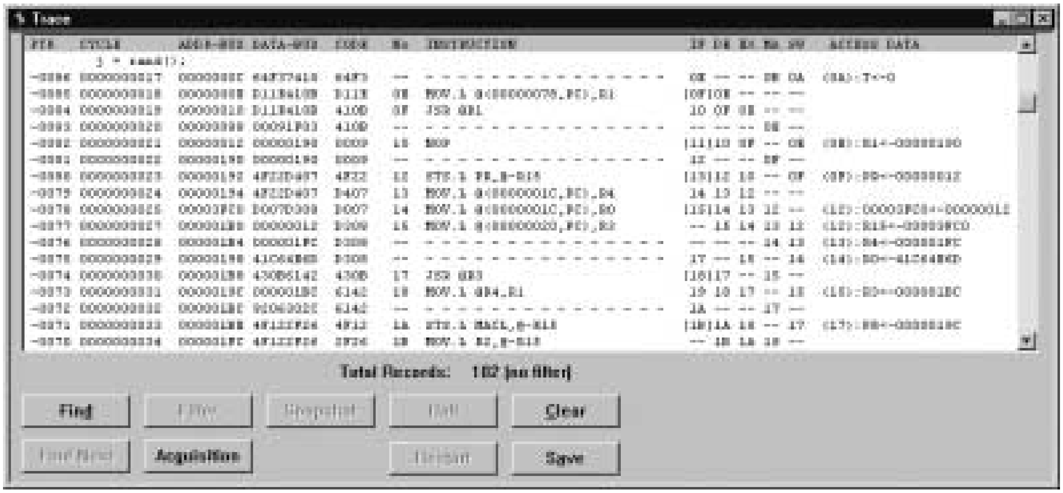
[PTR]	Pointer in the trace buffer (0 for the last executed instruction)
[CYCLE]	Total number of instruction execution cycles (cleared by pipeline reset)
[ADDR]	Instruction address
[PIPELINE]	Pipeline execution status Each symbol has the following meaning: F: Instruction fetch (with memory access) f: 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 an instruction

>: Split
<: Stall due to conflict

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

[INSTRUCTION]	Instruction mnemonic and data access (displayed in the form of [Transfer destination <— Transfer data])

SH-3 and SH-3E Series:



```
PTR    CYCLE        ADDR-BUS DATA-BUS  CODE  No  INSTRUCTION                    IF DE EX MA WB   ACCESS DATA
         i = rand();
-0086  0000000017   0000000C 84F37418  84F3  --  - - - - - - - - - - - - - -    0E -- -- 0B 0A  (0A):T<-0
-0085  0000000018   0000000E D11B410B  D11B  0E  MOV.L @(00000078,PC),R1       |0F|0E -- -- --
-0084  0000000019   00000010 D11B410B  410B  0F  JSR @R1                        10 0F 0E -- --
-0083  0000000020   00000088 00091F03  410B  --  - - - - - - - - - - - - - -    -- -- -- 0E --
-0082  0000000021   00000012 00000190  0009  10  NOP                           |11|10 0F -- 0E  (0E):R1<-00000190
-0081  0000000022   00000190 00000190  0009  --  - - - - - - - - - - - - - -    12 -- -- 0F --
-0080  0000000023   00000192 4F22D407  4F22  12  STS.L PR,@-R15                |13|12 10 -- 0F  (0F):PR<-00000012
-0079  0000000024   00000194 4F22D407  D407  13  MOV.L @(0000001C,PC),R4        14 13 12 -- --
-0078  0000000025   00003FC0 D007D308  D007  14  MOV.L @(0000001C,PC),R0       |15|14 13 12 --  (12):00003FC0<-00000012
-0077  0000000027   000001B0 00000012  D308  15  MOV.L @(00000020,PC),R3        -- 15 14 13 12  (12):R15<-00003FC0
-0076  0000000028   000001B4 000001FC  D308  --  - - - - - - - - - - - - - -    -- -- -- 14 13  (13):R4<-000001FC
-0075  0000000029   00000198 41C64E6D  D308  --  - - - - - - - - - - - - - -    17 -- 15 -- 14  (14):R0<-41C64E6D
-0074  0000000030   000001B8 430B6142  430B  17  JSR @R3                       |18|17 -- 15 --
-0073  0000000031   0000019C 000001BC  6142  18  MOV.L @R4,R1                   19 18 17 -- 15  (15):R3<-000001BC
-0072  0000000032   000001BE 92060020  6142  --  - - - - - - - - - - - - - -    1A -- -- 17 --
-0071  0000000033   000001BE 4F122F26  4F12  1A  STS.L MACL,@-R15              |1B|1A 18 -- 17  (17):PR<-000001BC
-0070  0000000034   000001FC 4F122F26  2F26  1B  MOV.L R2,@-R15                 -- 1B 1A 18 --
```

Total Records: 182 (no filter)

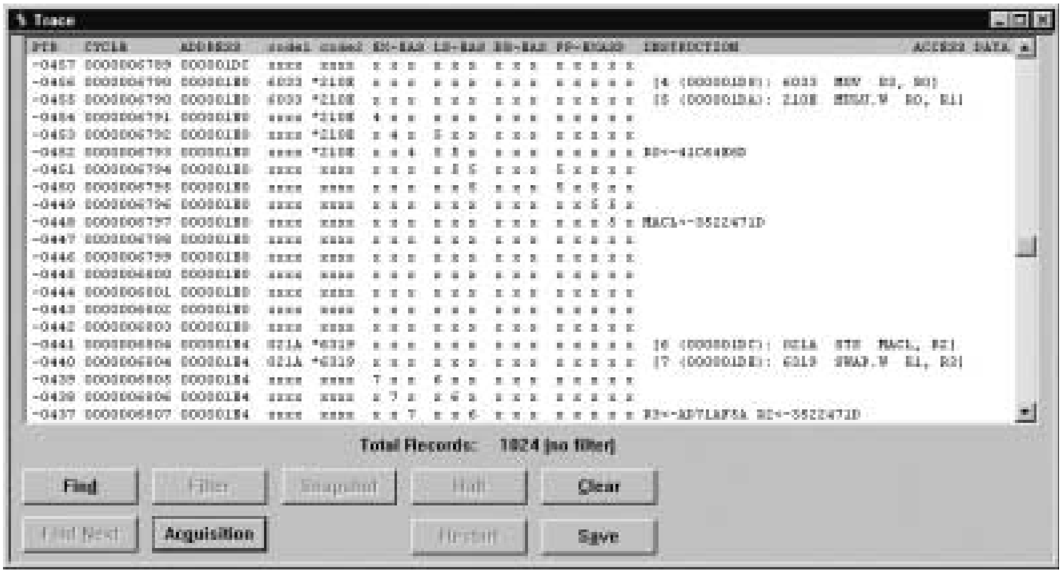
Figure 5.15   Trace Window (for SH-3 and SH-3E Series)

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)
[ADDR-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
[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)

SH-4 Series:



```
PTR    CYCLE      ADDRESS   code1 code2  EX-EAS LS-EAS BR-EAS FP-EXASD  INSTRUCTION                          ACCESS DATA
-0457 0000006789 000001DC  xxxx  xxxx   x x x  x x x  x x x  x x x x x
-0456 0000006790 000001E0  4033 *210E   x x x  x x x  x x x  x x x x x   [4 (000001D8): 4033  MOV   R3, R0]
-0455 0000006790 000001E0  4033 *210E   x x x  x x x  x x x  x x x x x   [5 (000001DA): 210E  MULU.W  R0, R1]
-0454 0000006791 000001E0  xxxx *210E   4 x x  x x x  x x x  x x x x x
-0453 0000006792 000001E0  xxxx *210E   x 4 x  5 x x  x x x  x x x x x
-0452 0000006793 000001E0  xxxx *210E   x x 4  5 5 x  x x x  x x x x x R0<-41C64E6D
-0451 0000006794 000001E0  xxxx  xxxx   x x x  x 5 5  x x x  5 x x x x
-0450 0000006795 000001E0  xxxx  xxxx   x x x  x x 5  x x x  5 x 5 x x
-0449 0000006796 000001E0  xxxx  xxxx   x x x  x x x  x x x  x x 5 5 x
-0448 0000006797 000001E0  xxxx  xxxx   x x x  x x x  x x x  x x x 5 x MACL<-3522471D
-0447 0000006798 000001E0  xxxx  xxxx   x x x  x x x  x x x  x x x x x
-0446 0000006799 000001E0  xxxx  xxxx   x x x  x x x  x x x  x x x x x
-0445 0000006800 000001E0  xxxx  xxxx   x x x  x x x  x x x  x x x x x
-0444 0000006801 000001E0  xxxx  xxxx   x x x  x x x  x x x  x x x x x
-0443 0000006802 000001E0  xxxx  xxxx   x x x  x x x  x x x  x x x x x
-0442 0000006803 000001E0  xxxx  xxxx   x x x  x x x  x x x  x x x x x
-0441 0000006804 000001E4  021A *6319   x x x  x x x  x x x  x x x x x   [6 (000001DC): 021A  STS   MACL, R2]
-0440 0000006804 000001E4  021A *6319   x x x  x x x  x x x  x x x x x   [7 (000001DE): 6319  SWAP.W  R1, R3]
-0439 0000006805 000001E4  xxxx  xxxx   7 x x  6 x x  x x x  x x x x x
-0438 0000006806 000001E4  xxxx  xxxx   x 7 x  x 6 x  x x x  x x x x x
-0437 0000006807 000001E4  xxxx  xxxx   x x 7  x x 6  x x x  x x x x x R3<-AD71AF5A R2<-3522471D

Total Records:    1824 (no filter)
```

Figure 5.16   Trace Window (for SH-4 Series)

This window displays the following trace information items:

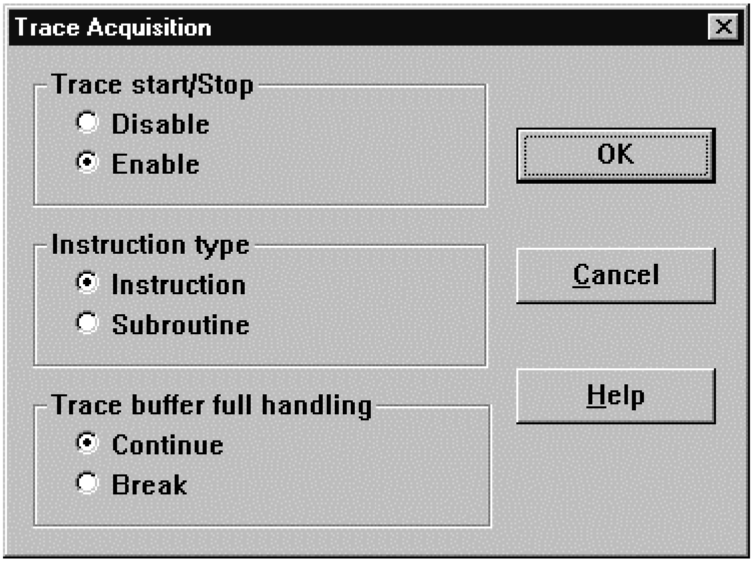
[PTR]	Pointer in the trace buffer (The latest instruction is 0)
[CYCLE]	Total number of instruction execution cycles (cleared by pipeline reset). The CPU internal clock cycles are counted as execution cycles. One execution cycle is equivalent to three external cycles.
[ADDRESS]	Program counter value
[code1]	Fetched code 1
[code2]	Fetched code 2
[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)
[INSTRUCTION]	Instruction number assigned to the instruction to be executed, memory address, instruction code, and mnemonic of that instruction.
[ACCESS DATA]	Data access information (display format: destination <- accessed data)

The buttons and pop-up menus, which will be displayed by clicking the view area with the right mouse button, are the same regardless of the CPU type. The window has the following buttons and pop-up menus.

[Find]	Searches trace information. Clicking this button opens the **Trace Search** dialog box, and search conditions can be specified.
[Find Next]	Searches trace information for the next occurrence of the item specified by **[Find]**.
[Filter]	Not supported by this simulator/debugger.
[Acquisition]	Specifies the conditions of trace information acquisition. Clicking this button opens the **Trace Acquisition** dialog box, and acquisition conditions can be specified.
[Snapshot]	Not supported by this simulator/debugger.
[Halt]	Not supported by this simulator/debugger.
[Restart]	Not supported by this simulator/debugger.
[Clear]	Clears the contents in the trace buffer.
[Save]	Saves trace information into a file.

5.12 Trace Acquisition Dialog Box


Trace Acquisition

Trace start/Stop
- ( ) Disable
- (●) Enable

Instruction type
- (●) Instruction
- ( ) Subroutine

Trace buffer full handling
- (●) Continue
- ( ) Break

Figure 5.17 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]

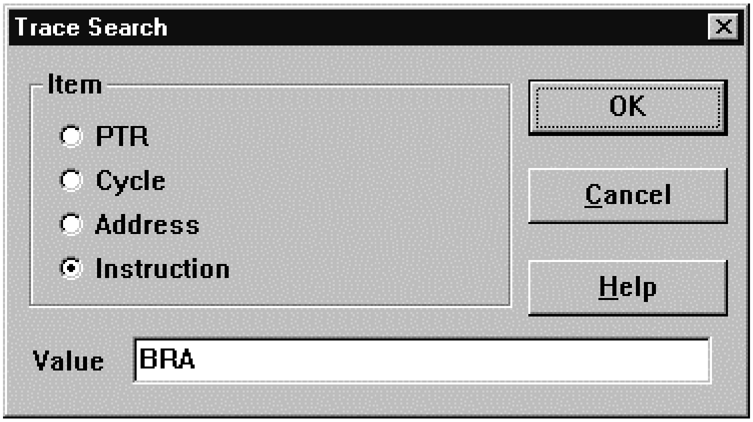
[Instruction]	Acquires trace information for all instructions.
[Subroutine]	Acquires trace information for the subroutine instructions only.

[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.

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

5.13 Trace Search Dialog Box



Item: ○ PTR ○ Cycle ○ Address [✓ Instruction]

Value: BRA

Figure 5.18 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]**.

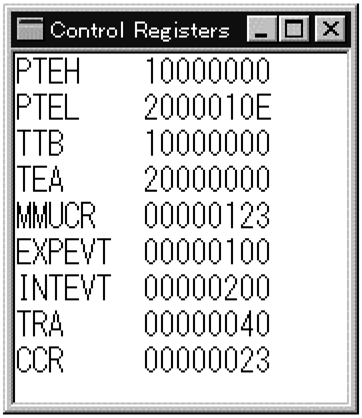
[PTR]	Pointer in the trace buffer (0 for the last executed instruction, specify in the form of -nnn)
[Cycle]	Total number of instruction execution cycles
[Address]	Instruction address
[Instruction]	Instruction mnemonic

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

5.14 Control Registers Window

This window displays the following control register values. This window is provided only for the SH-3, SH-3E, and SH-4 series.

SH-3 and SH-3E Series:

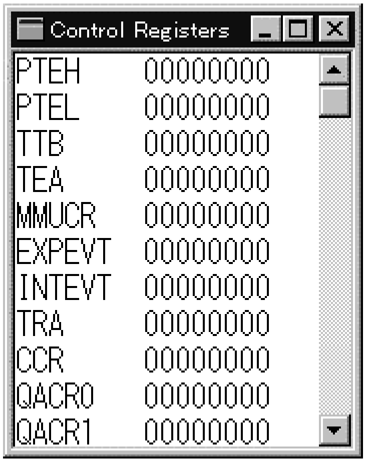


```
PTEH     10000000
PTEL     2000010E
TTB      10000000
TEA      20000000
MMUCR    00000123
EXPEVT   00000100
INTEVT   00000200
TRA      00000040
CCR      00000023
```

Figure 5.19 Control Registers Window (for SH-3 and SH-3E Series)

[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
[EXPEVT]	Exception event register
[INTEVT]	Interrupt event register
[TRA]	TRAPA exception register
[CCR]	Cache control register

SH-4 Series:



```
PTEH     00000000
PTEL     00000000
TTB      00000000
TEA      00000000
MMUCR    00000000
EXPEVT   00000000
INTEVT   00000000
TRA      00000000
CCR      00000000
QACR0    00000000
QACR1    00000000
```

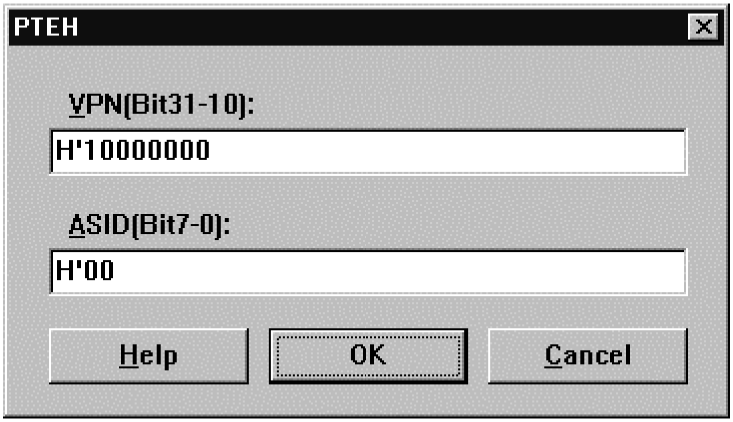
Figure 5.20 Control Registers Window (for SH-4 Series)

In addition to the registers displayed for the SH-3 and SH-3E series, the following registers are displayed for the SH-4 series.

[QACR0 and QACR1]	Queue address control registers 0 and 1
[SAR0 to SAR3]	DMA source address registers 0 to 3
[DAR0 to DAR3]	DMA destination address registers 0 to 3
[DMATCR0 to DMATCR3]	DMA transfer count registers 0 to 3
[CHCR0 to CHCR3]	DMA channel control registers 0 to 3
[DMAOR]	DMA operation register
[MCR]	Individual memory control register
[BCR1 and BCR2]	Bus control registers 1 and 2
[WCR1 to WCR3]	Wait state control registers 1 to 3
[RTCSR]	Refresh timer control/status register
[RTCNT]	Refresh timer counter
[RTCOR]	Refresh time constant counter
[RFCR]	Refresh counter register

The control register values can be directly modified in the window. Double-clicking a register opens the corresponding dialog box. In this dialog box, the register value can be modified in bit or field units.

5.15 PTEH Dialog Box


PTEH

VPN(Bit31-10):

H'10000000

ASID(Bit7-0):

H'00

Figure 5.21 PTEH Dialog Box

This dialog box specifies the following values of the page table entry high register (PTEH). This dialog box is provided only for the SH-3, SH-3E, and SH-4 series.

[VPN]	Virtual page number in longword size.
[ASID]	Address space ID.

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

5.16 PTEL Dialog Box

This dialog box specifies the following values of the page table entry low register (PTEL). This dialog box is provided only for the SH-3, SH-3E, and SH-4 series. The displayed contents depend on the target CPU.

For SH-3 and SH-3E Series:

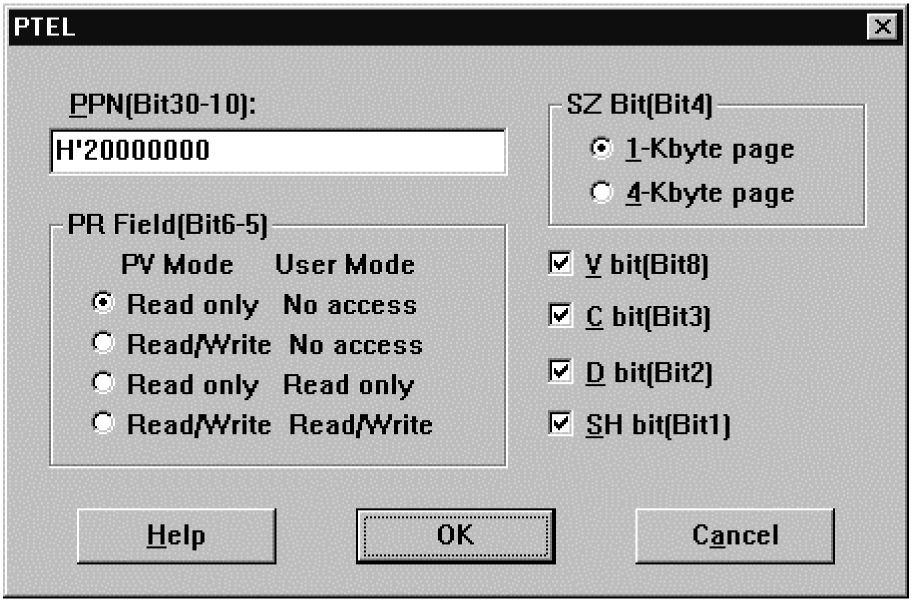
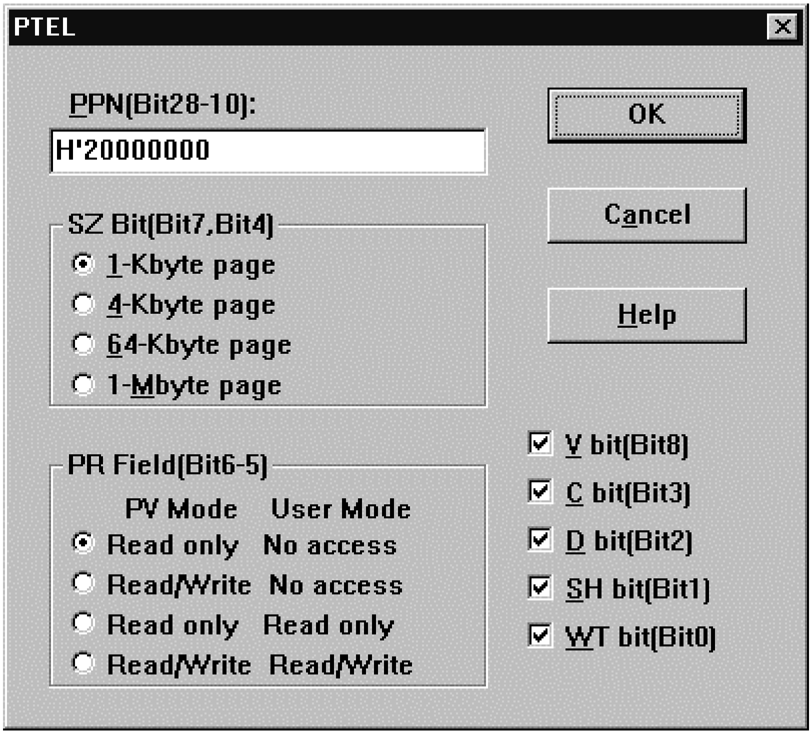


Figure 5.22 PTEL Dialog Box (for SH-3 and SH-3E Series)

[PPN]	Physical page number in longword size.		
[PR Field]	Specifies page protection status.		
	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.
[SZ Bit]	Page size bit.		
[V bit]	Valid bit.		

[C bit]	Cacheable bit.
[D bit]	Dirty bit.
[SH bit]	Sharing bit.

For SH-4 Series:


PTEL

PPN(Bit28-10): H'20000000

SZ Bit(Bit7,Bit4)
- (●) 1-Kbyte page
- ( ) 4-Kbyte page
- ( ) 64-Kbyte page
- ( ) 1-Mbyte page

PR Field(Bit6-5)

	PV Mode	User Mode
(●)	Read only	No access
( )	Read/Write	No access
( )	Read only	Read only
( )	Read/Write	Read/Write

- [✓] V bit(Bit8)
- [✓] C bit(Bit3)
- [✓] D bit(Bit2)
- [✓] SH bit(Bit1)
- [✓] WT bit(Bit0)

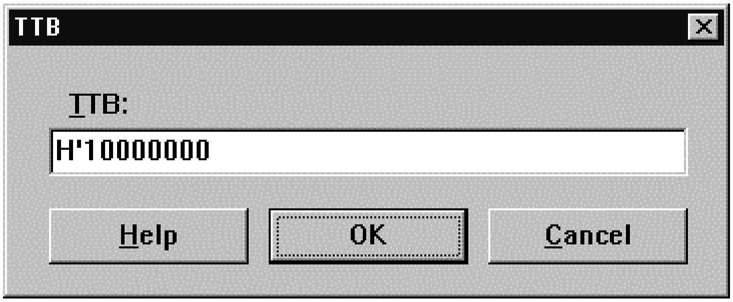
Figure 5.23 PTEL Dialog Box (for SH-4 Series)

In addition to the items set for the SH-3 and SH-3E series, the following item must be specified for the SH-4 series.

[WT bit]	Write-through bit. Specifies the writing mode for the cache.

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

5.17 TTB Dialog Box



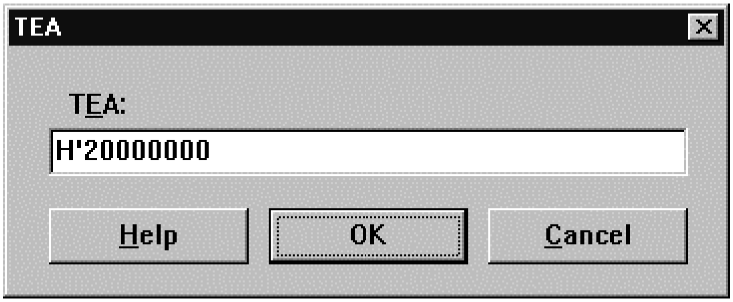
H'10000000

Figure 5.24 TTB Dialog Box

This dialog box specifies the value of the translation table base register (TTB) in longword size. This dialog box is provided only for the SH-3, SH-3E, and SH-4 series.

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

5.18 TEA Dialog Box



H'20000000

Figure 5.25 TEA Dialog Box

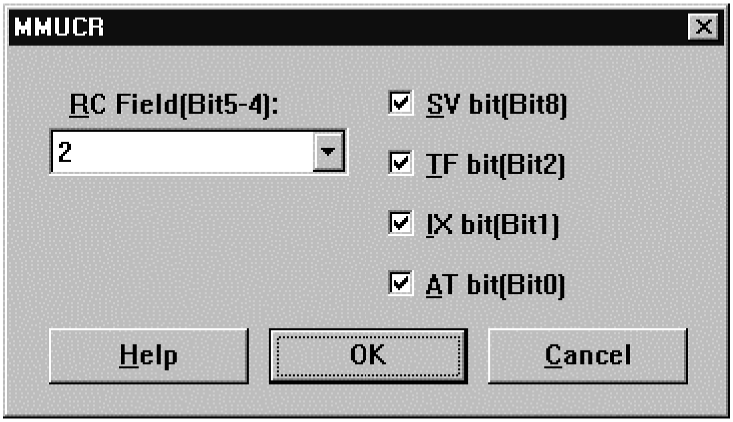
This dialog box specifies the value of the TLB exception address register (TEA) in longword size. This dialog box is provided only for the SH-3, SH-3E, and SH-4 series.

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

5.19 MMUCR Dialog Box

This dialog box specifies the following values of the MMU control register (MMUCR). This dialog box is provided only for the SH-3, SH-3E, and SH-4 series. The displayed contents depend on the target CPU.

For SH-3 and SH-3E Series:


MMUCR

RC Field(Bit5-4): 2

☑ SV bit(Bit8)

☑ TF bit(Bit2)

☑ IX bit(Bit1)

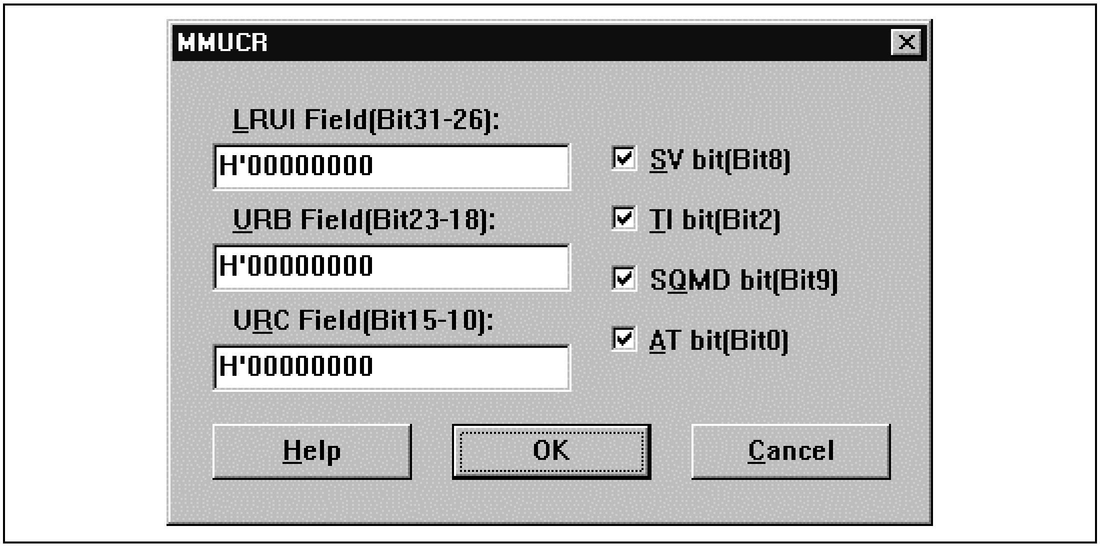
☑ AT bit(Bit0)

Figure 5.26 MMUCR Dialog Box (for SH-3 and SH-3E Series)

The following items must be specified. Selecting each item turns the setting on.

[RC Field]	Random counter.
[SV bit]	Single virtual memory mode bit.
[TF bit]	TLB flush bit. Selecting this box and clicking the **[OK]** button flushes TLB.
[IX bit]	Index mode bit.
[AT bit]	Address translation bit. Specifies whether or not to enable the MMU.

For SH-4 Series:



LRUI Field[Bit31-26]: H'00000000
URB Field[Bit23-18]: H'00000000
URC Field[Bit15-10]: H'00000000
☑ SV bit[Bit8]
☑ TI bit[Bit2]
☑ SQMD bit[Bit9]
☑ AT bit[Bit0]

Figure 5.27 MMUCR Dialog Box (for SH-4 Series)

The following items must be specified. Selecting each item turns the setting on.

[LRUI Field]	Number indicating the ITLB entry to be replaced when an ITLB miss occurs.
[URB Field]	Boundary value of the UTLB entry to be replaced.
[URC Field]	Random counter value, which indicates the UTLB entry to be replaced by the LDTLB instruction.
[SV bit]	Single virtual memory mode bit.
[TI bit]	TLB invalidating bit. Selecting this box and clicking the **[OK]** button flushes UTLB and ITLB.
[SQMD bit]	Store queue mode bit. Specifies the access right to the store queue.
[AT bit]	Address translation bit. Specifies whether or not to enable the MMU.

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

5.20 EXPEVT Dialog Box

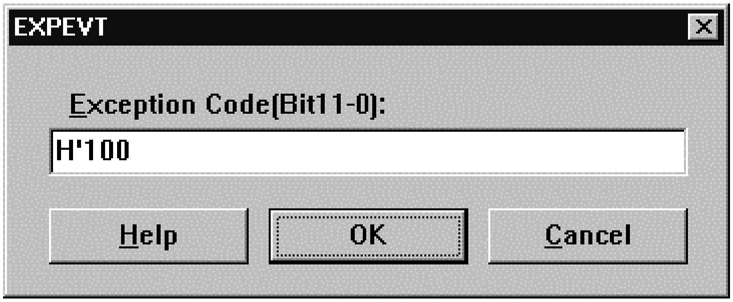


Figure 5.28 EXPEVT Dialog Box

This dialog box specifies the value of the exception event register (EXPEVT). This dialog box is provided only for the SH-3, SH-3E, and SH-4 series.

[Exception Code] specifies an exception code. (H'0 to H'FFF)

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

5.21 INTEVT Dialog Box

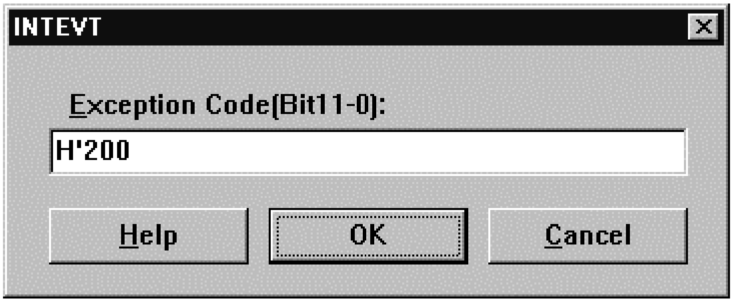


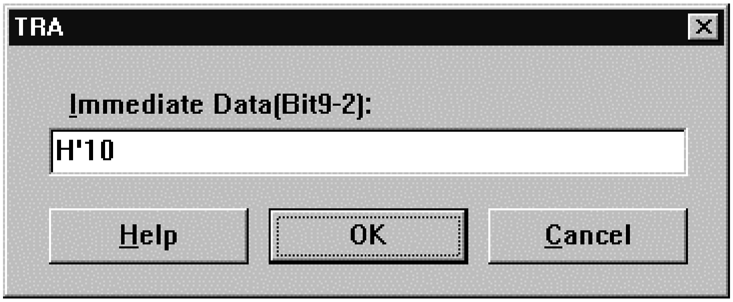
Figure 5.29 INTEVT Dialog Box

This dialog box specifies the value of the interrupt event register (INTEVT). This dialog box is provided only for the SH-3, SH-3E, and SH-4 series.

[Exception Code] specifies an exception code. (H'0 to H'FFF)

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

5.22 TRA Dialog Box



H'10

Figure 5.30 TRA Dialog Box

This dialog box specifies the value of the TRAPA exception register (TRA). This dialog box is provided only for the SH-3, SH-3E, and SH-4 series.

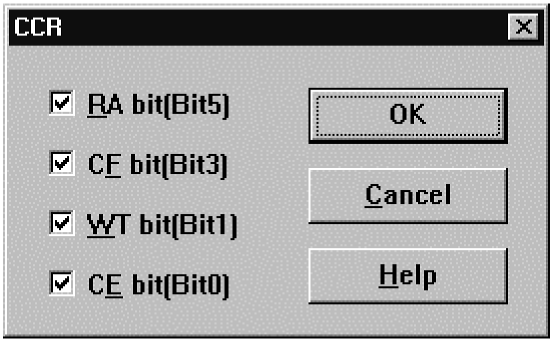
[Immediate Data] specifies an immediate value (H'0 to H'FF). The specified value is multiplied by four, and the result is set in the TRA register.

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

5.22 CCR Dialog Box

This dialog box specifies the following values of the cache control register (CCR). This dialog box is provided only for the SH-3, SH-3E, and SH-4 series. The displayed contents depend on the target CPU.

For SH-3 and SH-3E Series:


CCR

☑ RA bit[Bit5]

☑ CF bit[Bit3]

☑ WT bit[Bit1]

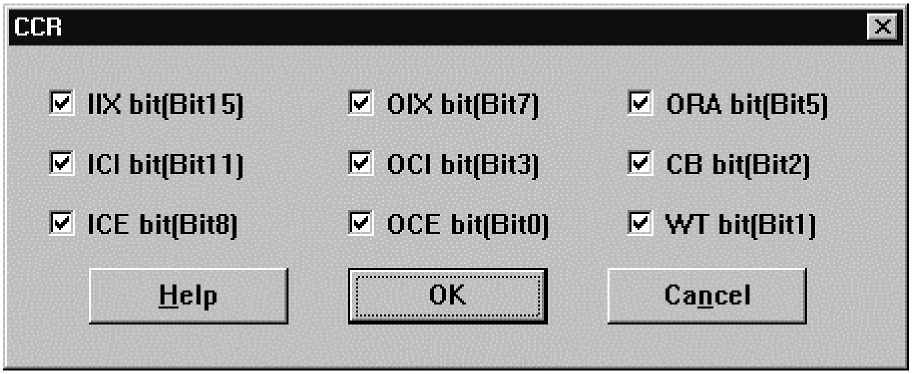
☑ CE bit[Bit0]

Figure 5.31 CCR Dialog Box (for SH-3 and SH-3E Series)

The following items must be specified. Selecting each item turns the setting on.

[RA bit]	RAM bit. Specifies the cache operating mode.
[CF bit]	Cache flush bit. Selecting this box and clicking the **[OK]** button flushes the V, U, and LRU bits of all entries in the cache.
[WT bit]	Write-through bit. Specifies the cache operating mode in the P0, U0, and P3 areas.
[CE bit]	Cache enable bit.

For SH-4 Series:



✓ IIX bit[Bit15]	✓ OIX bit[Bit7]	✓ ORA bit[Bit5]
✓ ICI bit[Bit11]	✓ OCI bit[Bit3]	✓ CB bit[Bit2]
✓ ICE bit[Bit8]	✓ OCE bit[Bit0]	✓ WT bit[Bit1]

Figure 5.32 CCR Dialog Box (for SH-4 Series)

The following items must be specified. Selecting each box turns the setting on.

[IIX bit]	IC index enable bit.
[ICI bit]	IC disable bit. Selecting this box and clicking the **[OK]** button flushes the V bits of all entries in the IC.
[ICE bit]	IC enable bit. Specifies whether or not to use the IC.
[OIX bit]	OC index enable bit.
[OCI bit]	OC disable bit. Selecting this box and clicking the **[OK]** button clears the V and the U bits of all entries in the OC to zero.
[OCE bit]	OC enable bit. Specifies whether or not to use the OC.
[ORA bit]	OC RAM bit.
[CB bit]	Copy-back bit. Specifies the cache writing mode in the P1 area.
[WT bit]	Write-through bit. Specifies the cache writing mode in the P0, U0, and P3 areas.

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

5.24 QACR0 and QACR1 Dialog Boxes

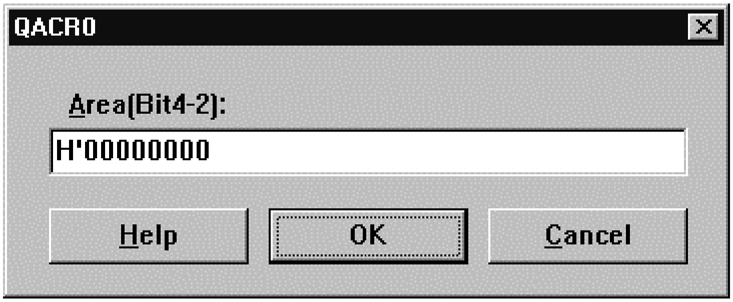


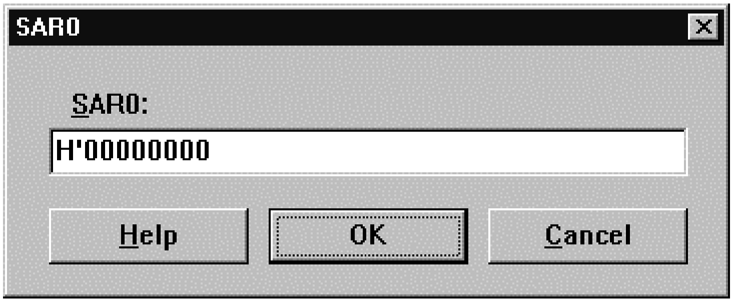
Figure 5.33 QACR0 Dialog Box

These dialog boxes specify the values of the queue address control registers 0 and 1 (QACR0 and QACR1). The QACR1 dialog box has the same functions as the QACR0 dialog box shown in figure 5.33. These dialog boxes are provided only for the SH-4 series.

In these dialog boxes, specify the areas where the store queues (0 and 1) are mapped when the MMU is disabled.

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

5.25 SAR0 to SAR3 Dialog Boxes



H'00000000

Figure 5.34 SAR0 Dialog Box

These dialog boxes specify the values of the DMA source address registers 0 to 3 (SAR0 to SAR3). The SAR1 to SAR3 dialog boxes have the same functions as the SAR0 dialog box shown in figure 5.34. These dialog boxes are provided only for the SH-4 series.

In these dialog boxes, specify the DMA transfer source addresses corresponding to channels 0 to 3.

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

5.26 DAR0 to DAR3 Dialog Boxes

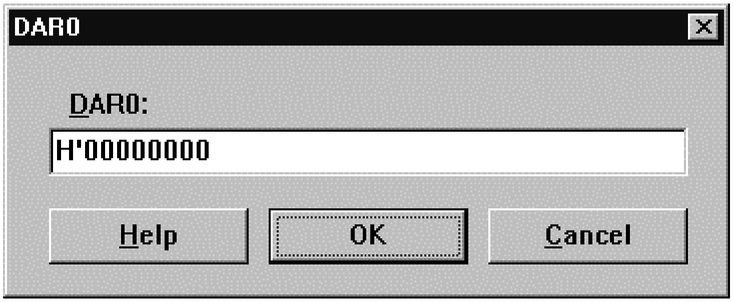


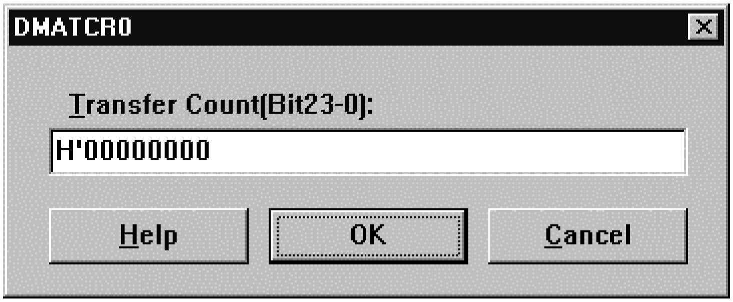
Figure 5.35 DAR0 Dialog Box

These dialog boxes specify the values of the DMA destination address registers 0 to 3 (DAR0 to DAR3). The DAR1 to DAR3 dialog boxes have the same functions as the DAR0 dialog box shown in figure 5.35. These dialog boxes are provided only for the SH-4 series.

In these dialog boxes, specify the DMA transfer destination addresses corresponding to channels 0 to 3.

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

5.27 DMATCR0 to DMATCR3 Dialog Boxes



Transfer Count(Bit23-0): H'00000000

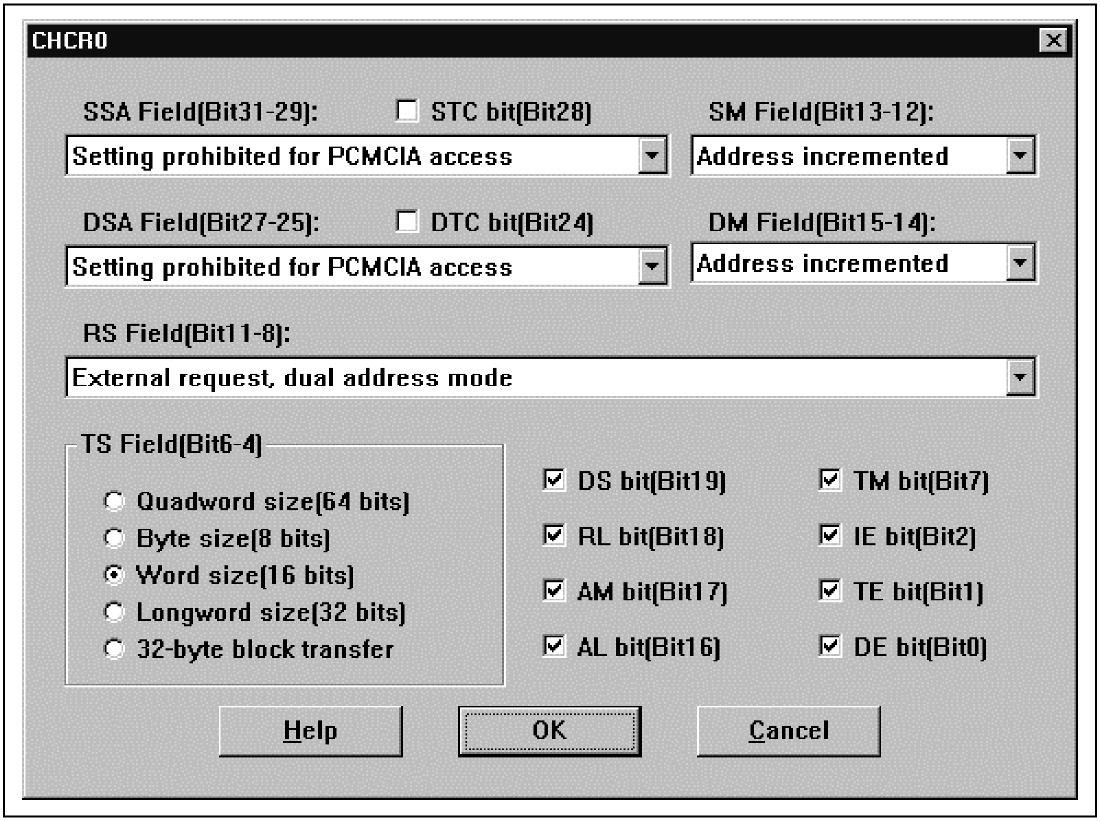
Figure 5.36 DMATCR0 Dialog Box

These dialog boxes specify the values of the DMA transfer count registers 0 to 3 (DMATCR0 to DMATCR3). The DMATCR1 to DMATCR3 dialog boxes have the same functions as the DMATCR0 dialog box shown in figure 5.36. These dialog boxes are provided only for the SH-4 series.

In these dialog boxes, specify the transfer count corresponding to channels 0 to 3.

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

5.28 CHCR0 to CHCR3 Dialog Boxes


CHCR0

SSA Field(Bit31-29): Setting prohibited for PCMCIA access

☐ STC bit(Bit28)

SM Field(Bit13-12): Address incremented

DSA Field(Bit27-25): Setting prohibited for PCMCIA access

☐ DTC bit(Bit24)

DM Field(Bit15-14): Address incremented

RS Field(Bit11-8): External request, dual address mode

TS Field(Bit6-4)

- ○ Quadword size(64 bits)
- ○ Byte size(8 bits)
- ● Word size(16 bits)
- ○ Longword size(32 bits)
- ○ 32-byte block transfer

☑ DS bit(Bit19)

☑ RL bit(Bit18)

☑ AM bit(Bit17)

☑ AL bit(Bit16)

☑ TM bit(Bit7)

☑ IE bit(Bit2)

☑ TE bit(Bit1)

☑ DE bit(Bit0)

Figure 5.37 CHCR0 Dialog Box

These dialog boxes specify the values of the DMA channel control registers 0 to 3 (CHCR0 to CHCR3). The CHCR1 to CHCR3 dialog boxes have the same functions as the CHCR0 dialog box shown in figure 5.37. These dialog boxes are provided only for the SH-4 series.

In these dialog boxes, specify the following values. Selecting each box turns the setting on.

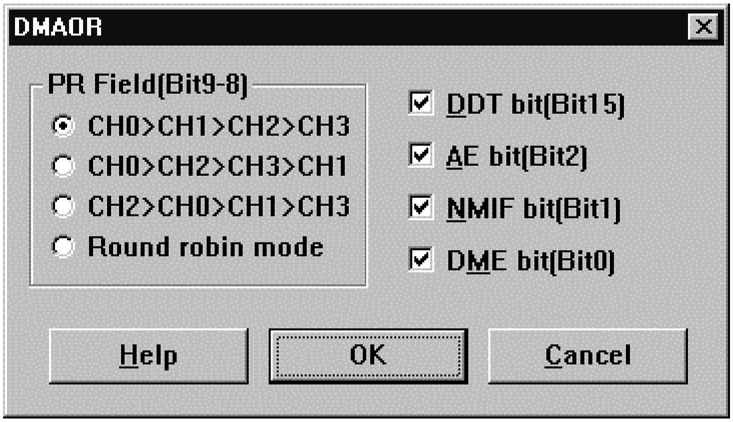
[SSA Field]	Specifies attributes for the source address space.
[STC bit]	Specifies wait control for the source address space.
[DSA Field]	Specifies attributes for the destination address space.
[DTC bit]	Specifies wait control for the destination address space.
[SM Field]	Source address mode. Specifies whether the DMA transfer source address is incremented or decremented.

[DM Field]	Destination address mode. Specifies whether the DMA transfer destination address is incremented or decremented.
[RS Field]	Resource select. Specifies the transfer request source.
[TS Field]	Transmit size. Specifies the transfer data size.
[DS bit]	$\overline{\text{DREQ}}$ select bit.
[RL bit]	Request check level bit.
[AM bit]	Acknowledge mode bit.
[AL bit]	Acknowledge level bit.
[TM bit]	Transmit mode bit. Specifies the bus mode for transfer.
[IE bit]	Interrupt enable bit.
[TE bit]	Transfer end bit. This bit is set when transfer has been completed for the count specified in the DMATCR.
[DE bit]	DMAC enable bit. This bit is enabled when the user program execution starts.

This simulator/debugger does not support PCMCIA. For the [RS Field] setting, only the automatic request and the external area can be selected.

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

5.29 DMAOR Dialog Box


DMAOR

PR Field(Bit9-8)

(●) CH0>CH1>CH2>CH3

( ) CH0>CH2>CH3>CH1

( ) CH2>CH0>CH1>CH3

( ) Round robin mode

[✓] DDT bit(Bit15)

[✓] AE bit(Bit2)

[✓] NMIF bit(Bit1)

[✓] DME bit(Bit0)

Figure 5.38 DMAOR Dialog Box

This dialog box specifies the values of the DMA operation register (DMAOR). This dialog box is provided only for the SH-4 series.

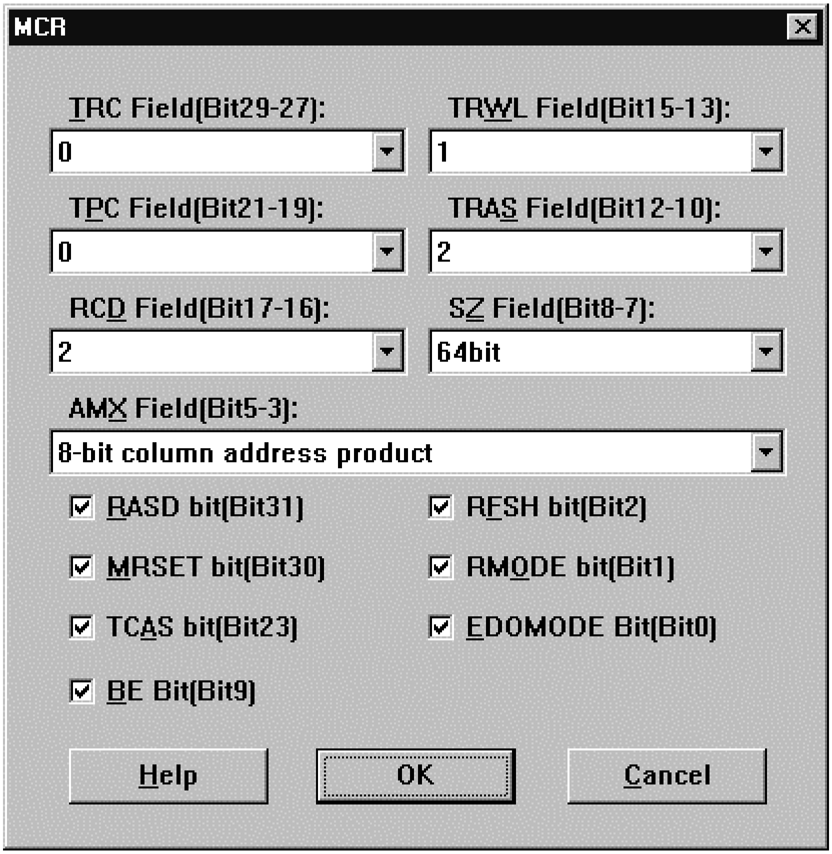
In this dialog box, specify the following values. Selecting each box turns the setting on.

[PR Field]	Priority mode. Specifies the priority of the channels when transfer is requested to two or more channels at the same time.
[DDT bit]*	On-demand data transfer bit.
[AE bit]	Address error flag.
[NMIF bit]*	NMI flag.
[DME bit]	DMAC master enable bit. Enables whole DMAC operations. This bit becomes valid when the user program execution starts.

The simulator/debugger does not support the functions marked with *.

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

5.30 MCR Dialog Box



TRC Field(Bit29-27): 0

TRWL Field(Bit15-13): 1

TPC Field(Bit21-19): 0

TRAS Field(Bit12-10): 2

RCD Field(Bit17-16): 2

SZ Field(Bit8-7): 64bit

AMX Field(Bit5-3): 8-bit column address product

✓ RASD bit(Bit31)

✓ RFSH bit(Bit2)

✓ MRSET bit(Bit30)

✓ RMODE bit(Bit1)

✓ TCAS bit(Bit23)

✓ EDOMODE Bit(Bit0)

✓ BE Bit(Bit9)

Figure 5.39 MCR Dialog Box

This dialog box specifies the values of the individual memory control register (MCR). This dialog box is provided only for the SH-4 series.

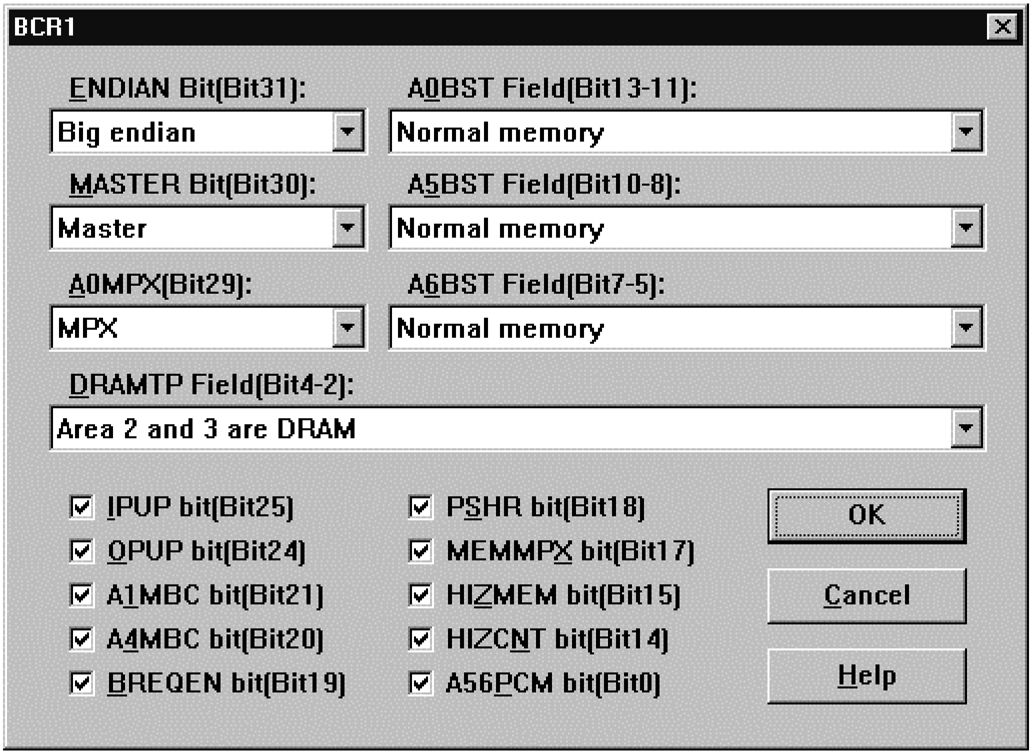
In this dialog box, specify the following values. Selecting each box turns the setting on.

[TRC Field]	Specifies the $\overline{\text{RAS}}$ precharge period after refresh.
[TPC Field]	$\overline{\text{RAS}}$ precharge period.
[RCD Field]	$\overline{\text{RAS}}$-$\overline{\text{CAS}}$ delay.
[TRWL Field]	Write precharge delay.
[TRAS Field]	$\overline{\text{RAS}}$ assertion period for $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh.

[SZ Field]	Memory data size.
[AMX Field]	Address multiplexing.
[RASD bit]	$\overline{\text{RAS}}$ down mode bit.
[MRSET]	Mode register set.
[TCAS bit]	$\overline{\text{CAS}}$ negation period.
[BE Bit]	Burst enable bit.
[RFSH bit]	Refresh control bit.
[RMODE bit]	Refresh mode bit.
[EDOMODE bit]	EDO mode bit.

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

5.31 BCR1 Dialog Box


BCR1

ENDIAN Bit(Bit31): Big endian

A0BST Field(Bit13-11): Normal memory

MASTER Bit(Bit30): Master

A5BST Field(Bit10-8): Normal memory

A0MPX(Bit29): MPX

A6BST Field(Bit7-5): Normal memory

DRAMTP Field(Bit4-2): Area 2 and 3 are DRAM

[✓] IPUP bit(Bit25)
[✓] OPUP bit(Bit24)
[✓] A1MBC bit(Bit21)
[✓] A4MBC bit(Bit20)
[✓] BREQEN bit(Bit19)
[✓] PSHR bit(Bit18)
[✓] MEMMPX bit(Bit17)
[✓] HIZMEM bit(Bit15)
[✓] HIZCNT bit(Bit14)
[✓] A56PCM bit(Bit0)

Figure 5.40 BCR1 Dialog Box

This dialog box specifies the values of the bus control register 1 (BCR1). This dialog box is provided only for the SH-4 series.

In this dialog box, specify the following values. Selecting each box turns the setting on.

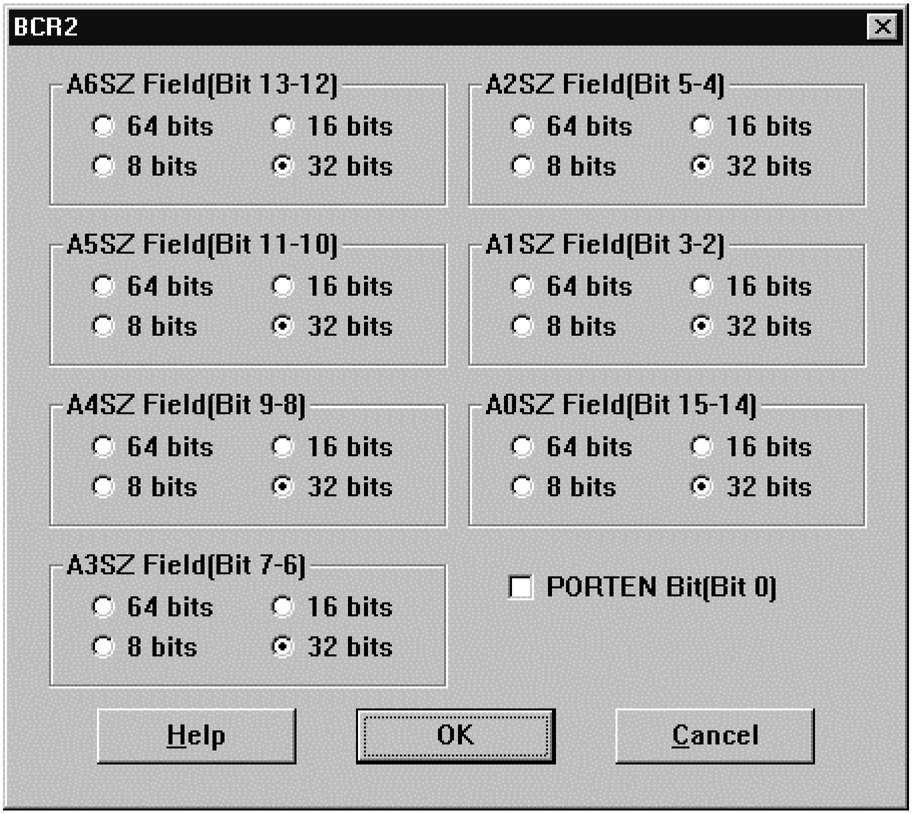
[ENDIAN Bit]	Endian flag.
[MASTER Bit]	Master/slave flag.
[A0MPX]	Memory type of area 0.
[A0BST Field]	Burst ROM control in area 0. When using the burst ROM, specify the burst count together.
[A5BST Field]	Burst ROM control in area 5. When using the burst ROM, specify the burst count together.

[A6BST Field]	Burst ROM control in area 6. When using the burst ROM, specify the burst count together.
[DRAMTP Field]	Memory type for areas 2 and 3.
[IPUP bit]*	Pull-up resistor state for control input pins.
[OPUP bit]*	Pull-up resistor state for control output pins.
[A1MBC bit]	SRAM byte control mode for area 1.
[A4MBC bit]	SRAM byte control mode for area 4.
[BREQEN bit]*	BREQ enable bit.
[PSHR bit]*	Partial sharing mode bit.
[MEMMPX bit]	MPX bus bit for areas 1 to 6.
[HIZMEM bit]*	High-impedance control bit.
[HIZCNT bit]*	High-impedance control bit.
[A56PCM bit]	Bus type for areas 5 and 6.

The simulator/debugger does not support the functions marked with *, and does not support PCMCIA.

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

5.32 BCR2 Dialog Box


BCR2

A6SZ Field[Bit 13-12]: 64 bits, 16 bits, 8 bits, [✓ 32 bits]

A2SZ Field[Bit 5-4]: 64 bits, 16 bits, 8 bits, [✓ 32 bits]

A5SZ Field[Bit 11-10]: 64 bits, 16 bits, 8 bits, [✓ 32 bits]

A1SZ Field[Bit 3-2]: 64 bits, 16 bits, 8 bits, [✓ 32 bits]

A4SZ Field[Bit 9-8]: 64 bits, 16 bits, 8 bits, [✓ 32 bits]

A0SZ Field[Bit 15-14]: 64 bits, 16 bits, 8 bits, [✓ 32 bits]

A3SZ Field[Bit 7-6]: 64 bits, 16 bits, 8 bits, [✓ 32 bits]

☐ PORTEN Bit[Bit 0]

Figure 5.41 BCR2 Dialog Box

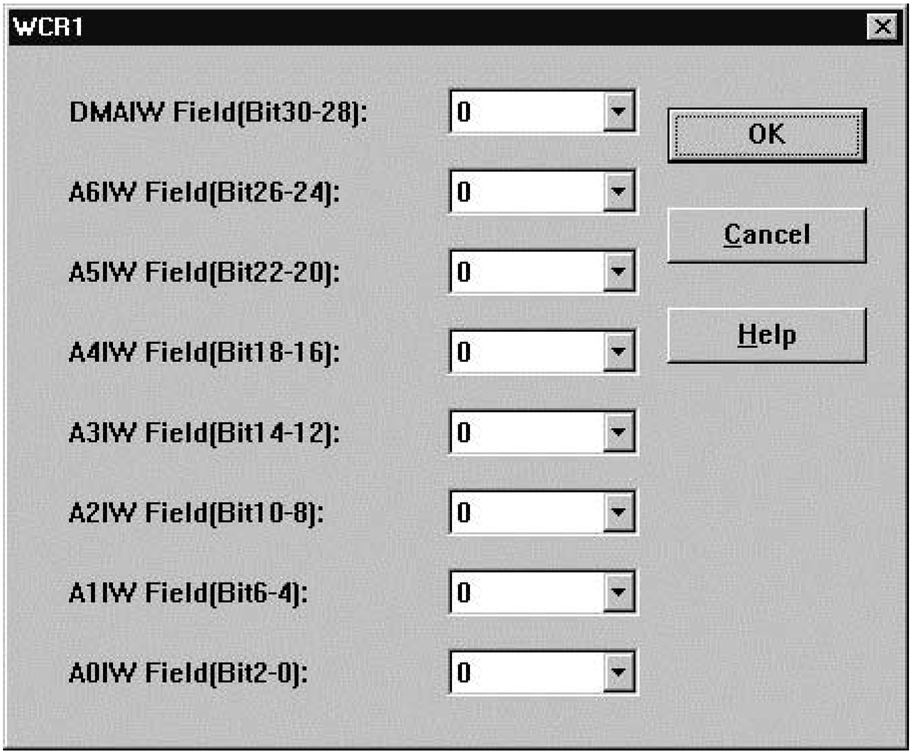
This dialog box specifies the values of the bus control register 2 (BCR2). This dialog box is provided only for the SH-4 series.

In this dialog box, specify the following values:

[A0SZ Field] to [A6SZ Field]	Bus width of the corresponding area (0 to 6).
[PORTEN Bit]	Port function enable bit. When this box is selected, the pins are used as ports.

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

5.33 WCR1 Dialog Box



WCR1

DMAIW Field(Bit30-28): 0
A6IW Field(Bit26-24): 0
A5IW Field(Bit22-20): 0
A4IW Field(Bit18-16): 0
A3IW Field(Bit14-12): 0
A2IW Field(Bit10-8): 0
A1IW Field(Bit6-4): 0
A0IW Field(Bit2-0): 0

Figure 5.42 WCR1 Dialog Box

This dialog box specifies the values of the wait control register 1 (WCR1). This dialog box is provided only for the SH-4 series.

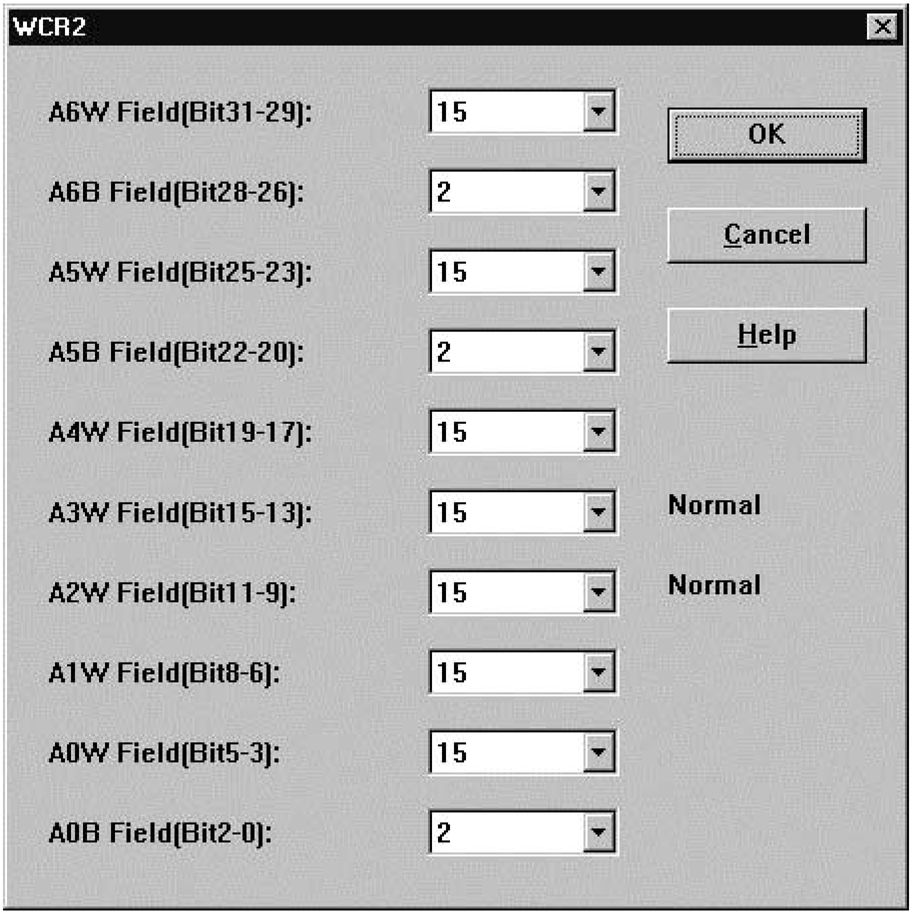
In this dialog box, specify the following values:

[DMAIW Field]*	Idle cycle count specification for the DMAIW-DACK devices.
[A0IW Field] to [A6IW Field]	Idle cycle count specification for areas 0 to 6.

The simulator/debugger does not support the function marked with *.

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

5.34 WCR2 Dialog Box



A6W Field(Bit31-29):	15
A6B Field(Bit28-26):	2
A5W Field(Bit25-23):	15
A5B Field(Bit22-20):	2
A4W Field(Bit19-17):	15
A3W Field(Bit15-13):	15 Normal
A2W Field(Bit11-9):	15 Normal
A1W Field(Bit8-6):	15
A0W Field(Bit5-3):	15
A0B Field(Bit2-0):	2

Figure 5.43 WCR2 Dialog Box

This dialog box specifies the values of the wait control register 2 (WCR2). This dialog box is provided only for the SH-4 series.

In this dialog box, specify the following values:

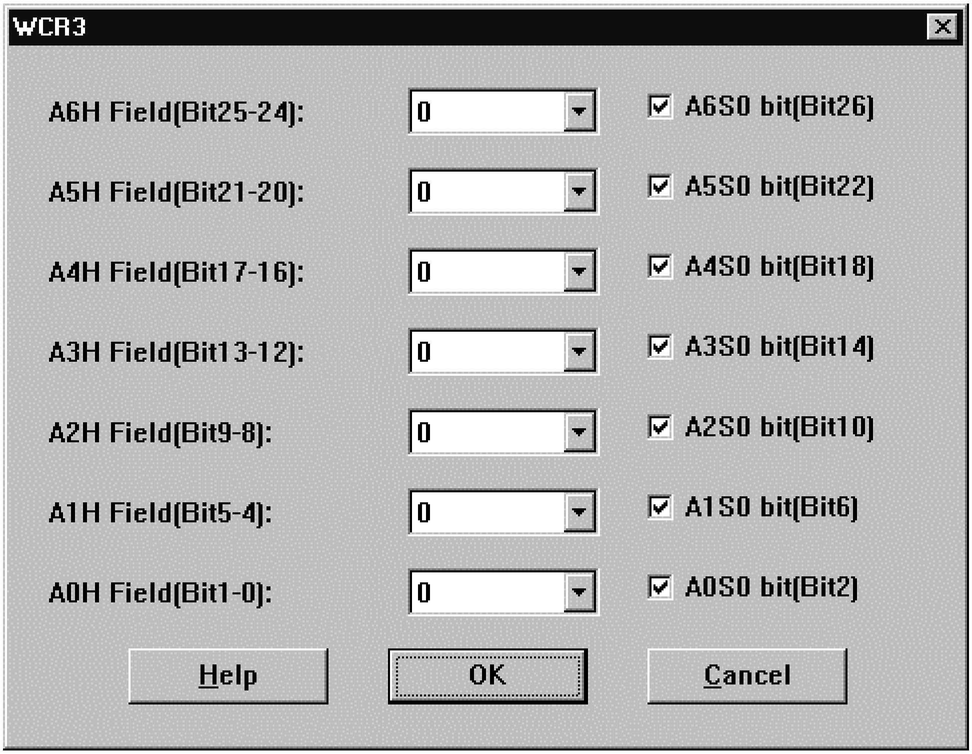
[A0W Field] to [A6W Field]	Wait state control for the corresponding area (0 to 6). Note that the specifications for areas 2 and 3 are valid only for normal memory. Memory types for areas 2 and 3 are indicated on the right side of [A3W Field] and [A2W Field].
[A6B Field]*	Burst pitch count for the burst transfer in area 6.

[A5B Field]*	Burst pitch count for the burst transfer in area 5.
[A0B Field]*	Burst pitch count for the burst transfer in area 0.

This simulator/debugger does not support the functions marked with *.

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

5.35 WCR3 Dialog Box



WCR3

A6H Field[Bit25-24]: 0 ☑ A6S0 bit[Bit26]

A5H Field[Bit21-20]: 0 ☑ A5S0 bit[Bit22]

A4H Field[Bit17-16]: 0 ☑ A4S0 bit[Bit18]

A3H Field[Bit13-12]: 0 ☑ A3S0 bit[Bit14]

A2H Field[Bit9-8]: 0 ☑ A2S0 bit[Bit10]

A1H Field[Bit5-4]: 0 ☑ A1S0 bit[Bit6]

A0H Field[Bit1-0]: 0 ☑ A0S0 bit[Bit2]

Figure 5.44 WCR3 Dialog Box

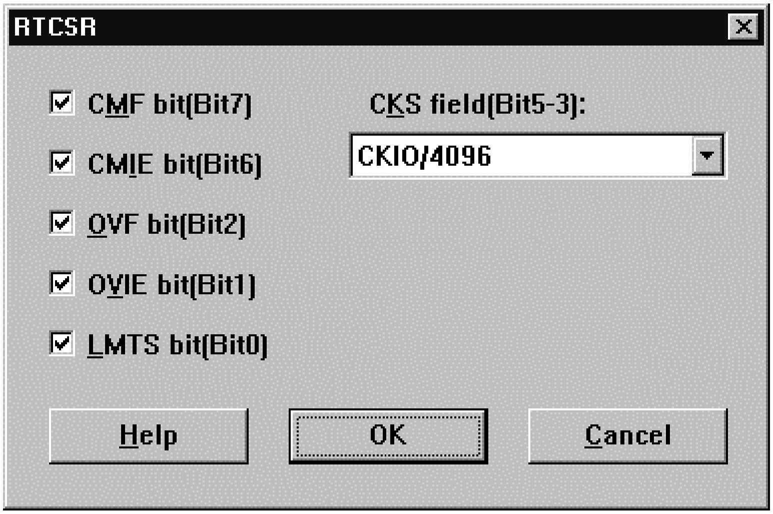
This dialog box specifies the values of the wait control register 3 (WCR3). This dialog box is provided only for the SH-4 series.

In this dialog box, specify the following values:

[A0H Field] to [A6H Field]	Data hold time for the corresponding area (0 to 6).
[A0S0 bit] to [A6S0 bit]	Setup time of the write strobe signal for the corresponding area (0 to 6).

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

5.36 RTCSR Dialog Box



☑ CMF bit(Bit7)
☑ CMIE bit(Bit6)
☑ OVF bit(Bit2)
☑ OVIE bit(Bit1)
☑ LMTS bit(Bit0)
CKS field(Bit5-3): CKIO/4096

Figure 5.45 RTCSR Dialog Box

This dialog box specifies the values of the refresh timer control/status register (RTCSR). This dialog box is provided only for the SH-4 series.

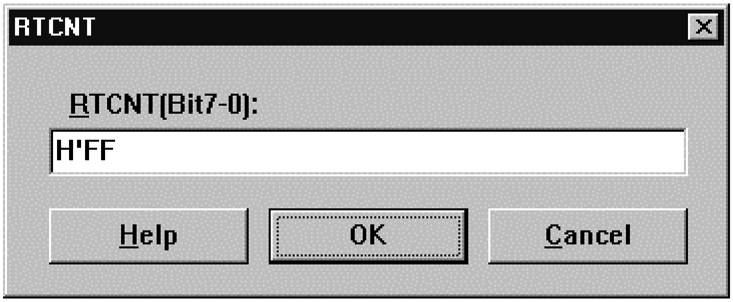
In this dialog box, specify the following values. Selecting each box turns the setting on.

[CMF bit]	Compare match flag.
[CMIE bit]*	Compare match interrupt enable bit.
[OVF bit]*	Refresh count overflow flag.
[OVIE bit]*	Refresh count overflow interrupt enable bit.
[LMTS bit]	Refresh count overflow limit select bit.
[CKS Field]	Clock select bit.

The simulator/debugger does not support the functions marked with *.

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

5.37 RTCNT Dialog Box



RTCNT(Bit7-0):

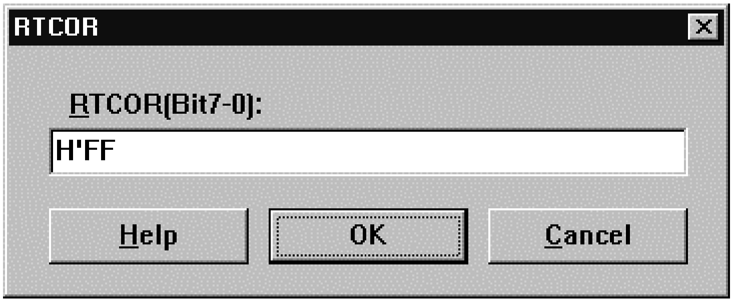
H'FF

Figure 5.46 RTCNT Dialog Box

This dialog box specifies the values of the refresh timer counter (RTCNT). This dialog box is provided only for the SH-4 series.

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

5.38 RTCOR Dialog Box



RTCOR(Bit7-0):

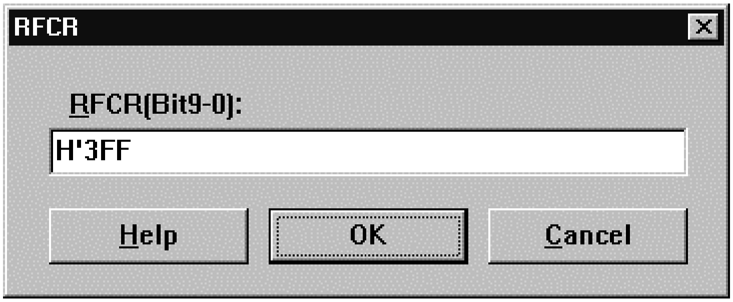
H'FF

Figure 5.47 RTCOR Dialog Box

This dialog box specifies the values of the refresh time constant register (RTCOR). This dialog box is provided only for the SH-4 series.

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

5.39 RFCR Dialog Box


RFCR

RFCR(Bit9-0):

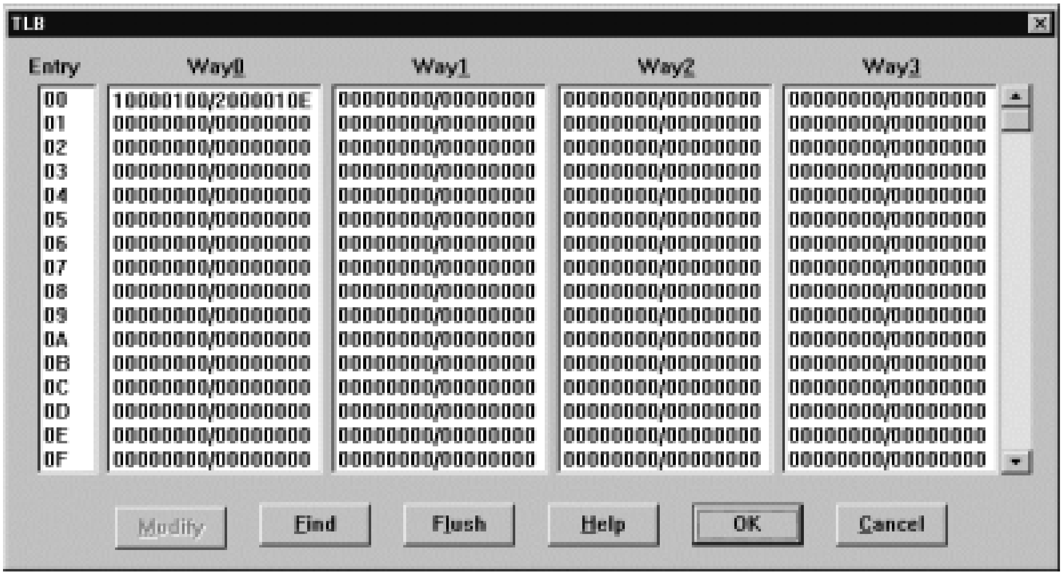
H'3FF

Figure 5.48 RFCR Dialog Box

This dialog box specifies the values of the refresh count register (RFCR). This dialog box is provided only for the SH-4 series.

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

5.40 TLB Dialog Box



Entry	Way0	Way1	Way2	Way3
00	10000100/2000010E	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 5.49 TLB Dialog Box

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

The following items are displayed.

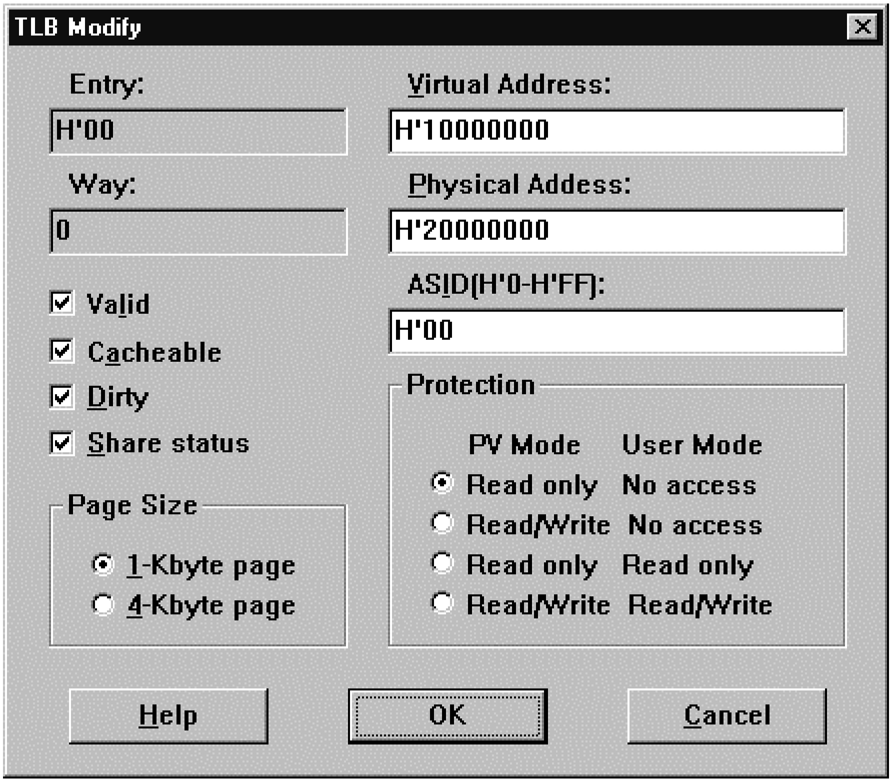
[Entry]	Entry number in the TLB (H'00 to H'1F)
[Way0]-[Way3]	Address array and data array in each way

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.

5.41 TLB Modify Dialog Box


TLB Modify

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 5.50 TLB Modify Dialog Box

This dialog box specifies the TLB contents of the displayed entry and way. This dialog box is provided only for the SH-3 and SH-3E series.

The following items can be specified.

[Virtual Address]	Virtual address in longword size. Bits 16 to 12 are set to 0 regardless of the input value. Bits 31 to 17, 11, and 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. Checking this box makes the entry valid.

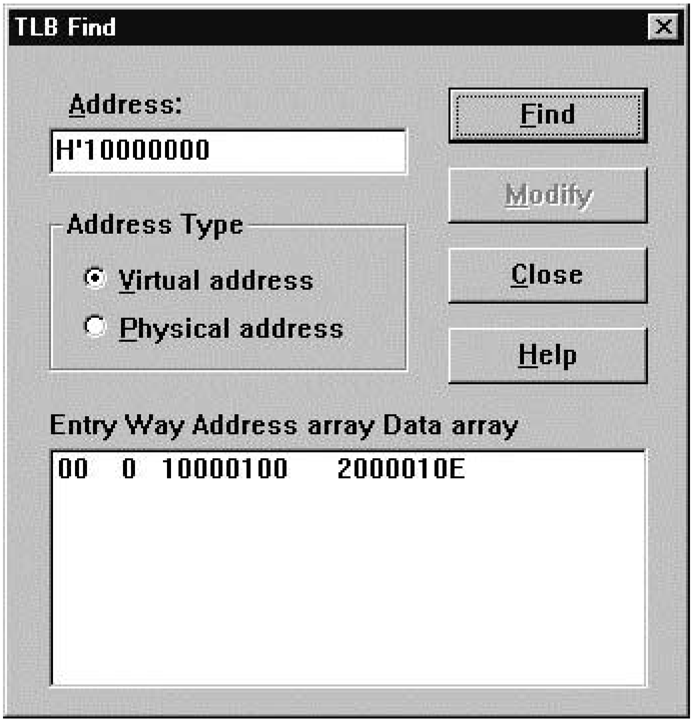
[Cacheable]	Enables or disables caching. Checking this box enables caching of the page.
[Dirty]	Specifies whether or not the page has been written to. Checking this box assumes that the page has been written to.
[Share status]	Specifies whether or not to share the page with multiple processes. Checking 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 displaying the modified contents in the **TLB** dialog box.

5.42 TLB Find Dialog Box



Address: H'10000000

Address Type: [✓ Virtual address] Physical address

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

Figure 5.51 TLB Find Dialog Box

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

5.43 Open TLB Dialog Box

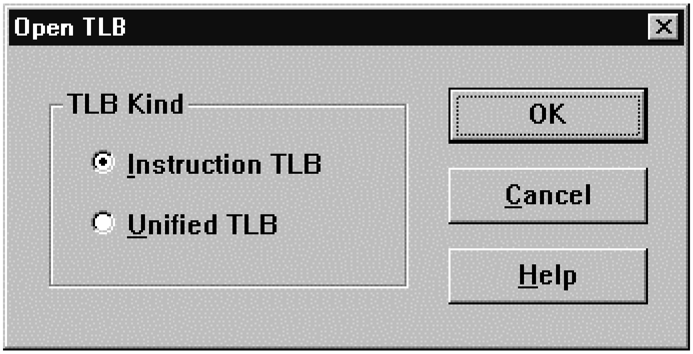


Figure 5.52 Open TLB Dialog Box

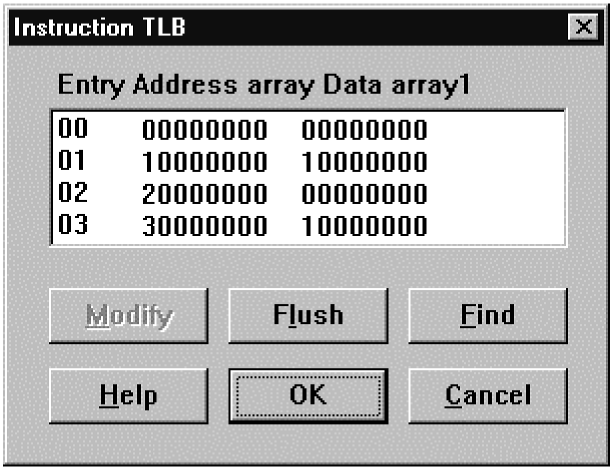
This dialog box selects the TLB to be displayed. This dialog box is provided only for the SH-4 series.

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.

5.44 Instruction TLB Dialog Box



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

Figure 5.53 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.

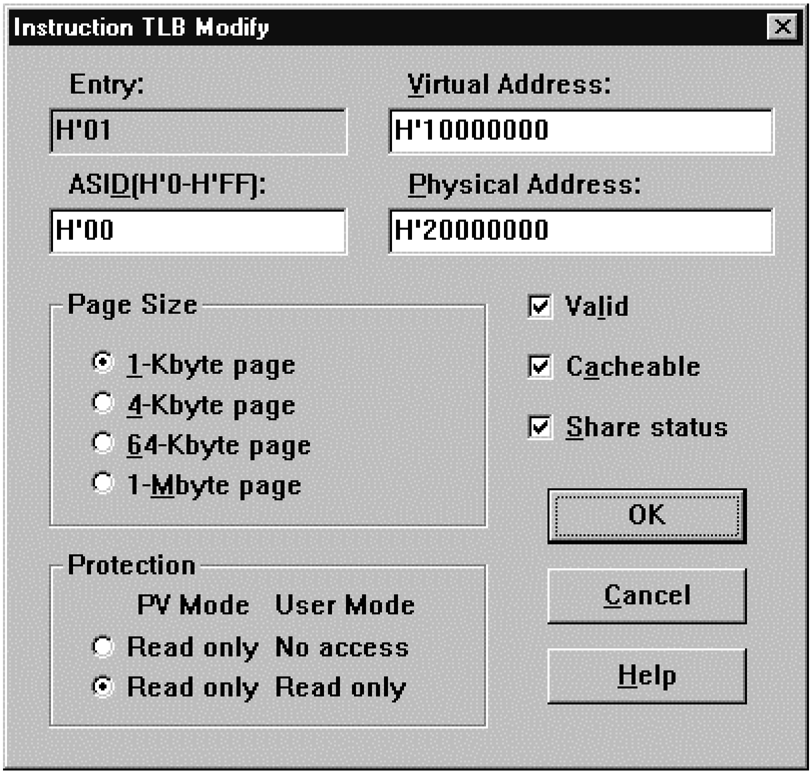
[Entry]	Entry number in the ITLB (H'00 to H'03)
[Address array]	Address array of the ITLB
[Data array1]	Data array 1 of the ITLB

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.

5.45 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

[✓ Valid]

[✓ Cacheable]

[✓ Share status]

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

Figure 5.54 Instruction TLB Modify Dialog Box

This dialog box modifies the ITLB contents of the displayed entry. This dialog box is provided only for the SH-4 series.

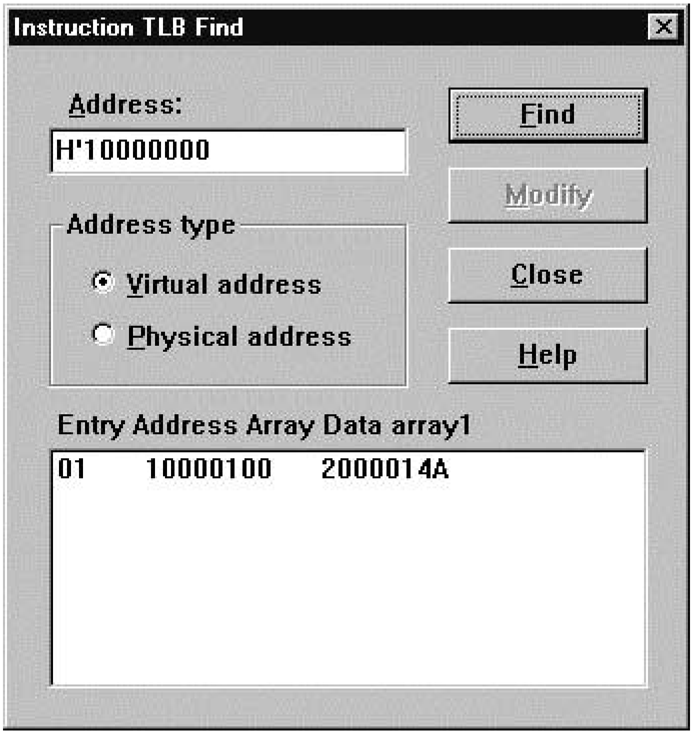
The following items can be specified.

[Entry]	Entry number selected by the Instruction TLB dialog box.
[ASID]	Address space ID, which specifies the process that can access the virtual page.
[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.
[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.
[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.		

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

5.46 Instruction TLB Find Dialog Box



Address: H'10000000

Address type: [✓ Virtual address] Physical address

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

Figure 5.55 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.

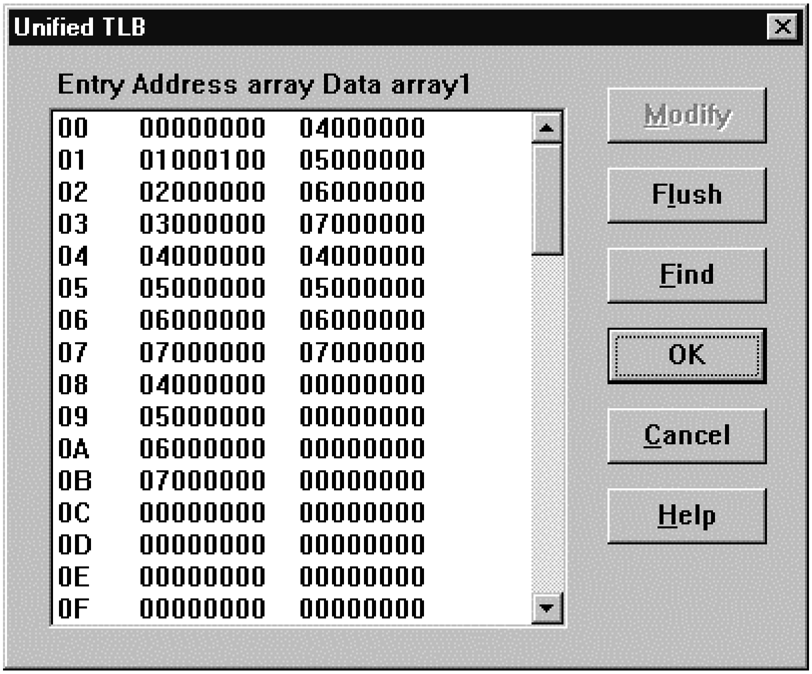
[Address]	Specifies the address to be searched for.
[Address type]	Specifies whether the address to be searched for is virtual or physical.

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.

5.47 Unified TLB Dialog Box



```
Entry Address array Data array1
00    00000000   04000000
01    01000100   05000000
02    02000000   06000000
03    03000000   07000000
04    04000000   04000000
05    05000000   05000000
06    06000000   06000000
07    07000000   07000000
08    04000000   00000000
09    05000000   00000000
0A    06000000   00000000
0B    07000000   00000000
0C    00000000   00000000
0D    00000000   00000000
0E    00000000   00000000
0F    00000000   00000000
```

Figure 5.56 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 way of the UTLB
[Data array1]	Data array 1 in each way of the UTLB

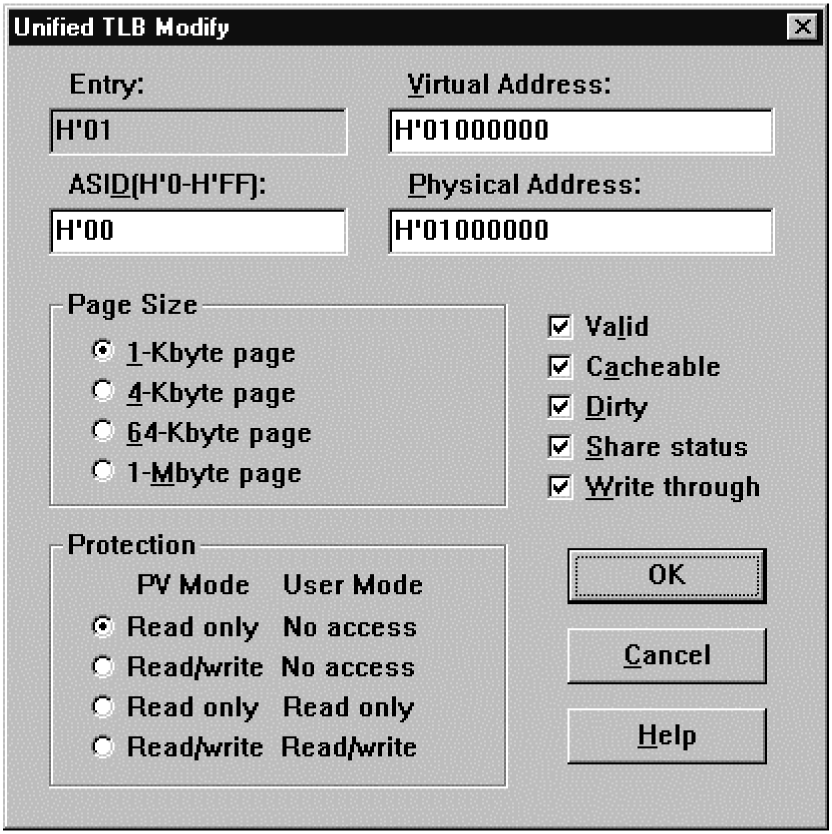
The UTLB contents can be modified, flushed, and searched using the following buttons.

[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.

5.48 Unified TLB Modify Dialog Box


Unified TLB Modify

Entry:	H'01
Virtual Address:	H'01000000
ASID(H'0-H'FF):	H'00
Physical Address:	H'01000000

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

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

Protection

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

Figure 5.57 Unified TLB Modify Dialog Box

This dialog box modifies the UTLB contents of the displayed entry. 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.
[ASID]	Address space ID, which specifies the process that can access the virtual page.
[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.
[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.

[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.

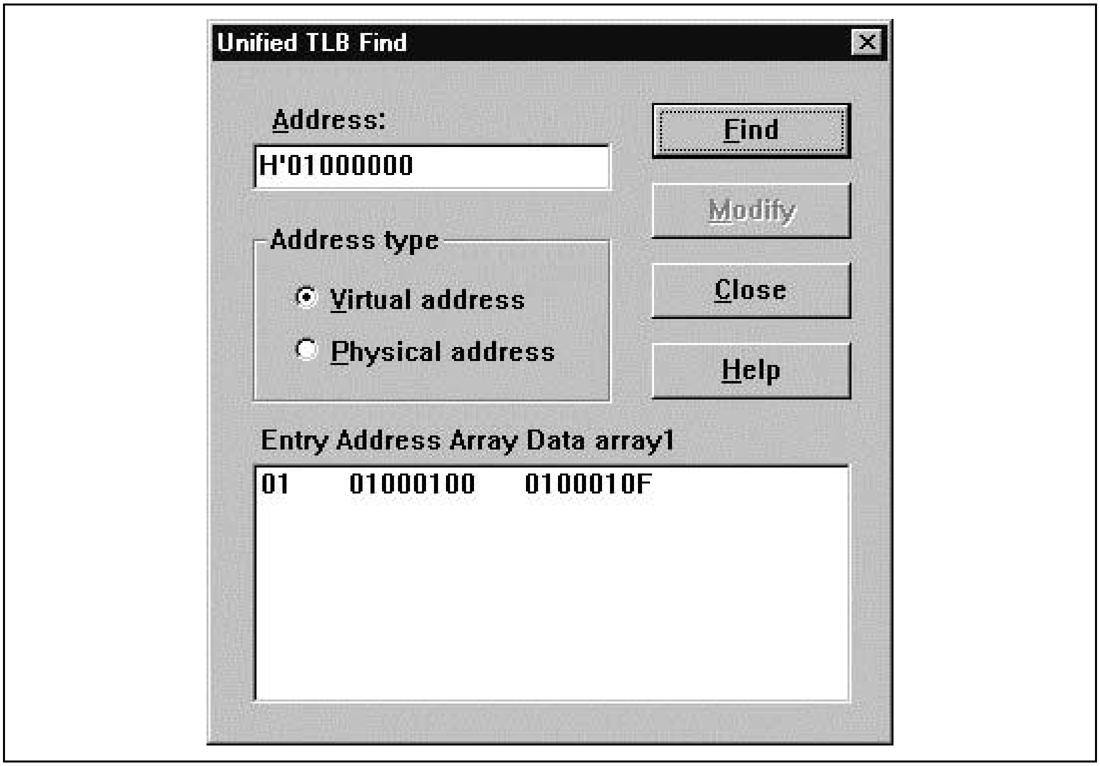
[Dirty] Specifies whether or not the page has been written to. Selecting this box makes the simulator/debugger assume that the page has been written to.

[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.

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

5.49 Unified TLB Find Dialog Box



Address: H'01000000

Address type
- (●) Virtual address
- ( ) Physical address

Entry Address Array Data array1

```
01      01000100     0100010F
```

Figure 5.58 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.

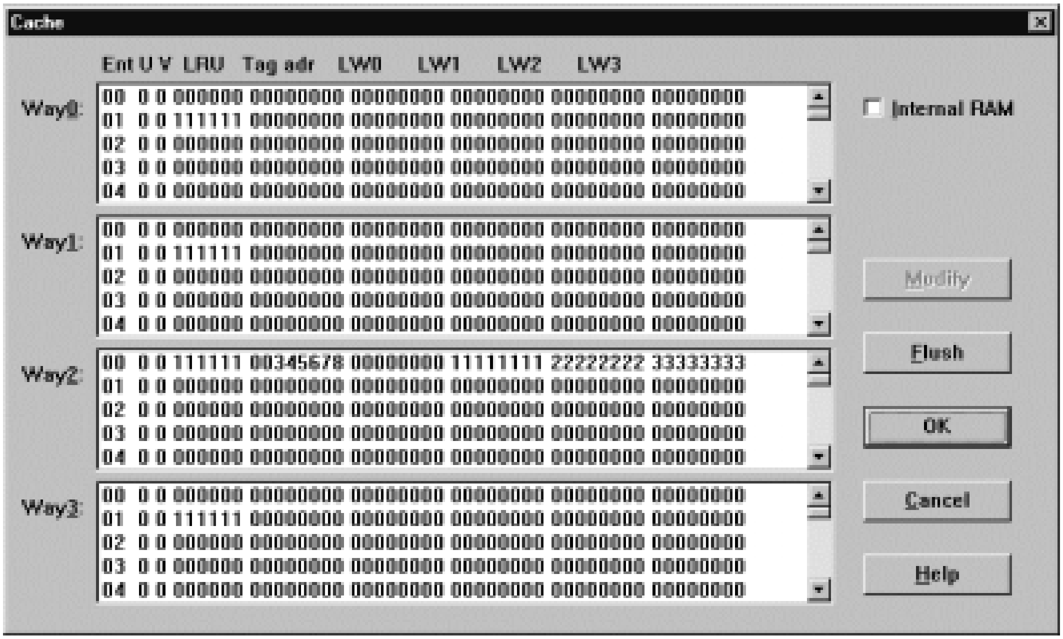
[Address]	Specifies the address to be searched for.
[Address type]	Specifies whether the address to be searched for is virtual or physical.

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.

5.50 Cache Dialog Box


Cache

```
       Ent U V  LRU    Tag adr   LW0       LW1       LW2       LW3
Way0:  00  0 0 000000 00000000 00000000 00000000 00000000 00000000
       01  0 0 111111 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 00000000 00000000 00000000 00000000 00000000
       01  0 0 111111 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 111111 00345678 00000000 11111111 22222222 33333333
       01  0 0 000000 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 111111 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
```

Figure 5.59 Cache Dialog Box

This dialog box displays the cache contents in ways 0 to 3. This dialog box is provided only for the SH-3 and SH-3E series.

The following items are displayed.

[Ent]	Entry number in the cache (H'00 to H'7F).
[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 entries.

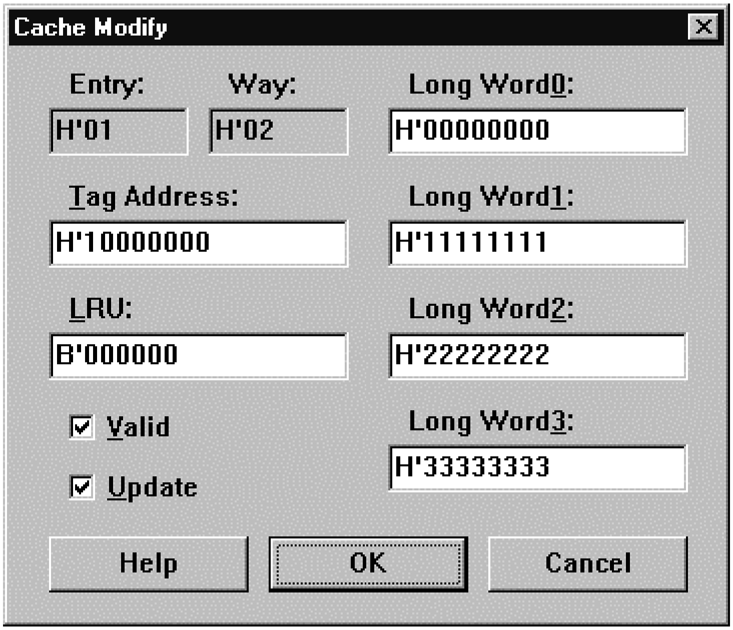
The cache contents can be modified and flushed using the following buttons.

[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.

Selecting the **[Internal RAM]** enables a half of the cache to be used as the internal RAM.

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

5.51 Cache Modify Dialog Box



Entry: H'01
Way: H'02
Long Word0: H'00000000
Tag Address: H'10000000
Long Word1: H'11111111
LRU: B'000000
Long Word2: H'22222222
☑ Valid
☑ Update
Long Word3: H'33333333

Figure 5.60 Cache Modify Dialog Box

This dialog box modifies the cache contents of the displayed way and entry. This dialog box is provided only for the SH-3 and SH-3E series.

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 physical address (longword) 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.

5.52 Open Cache Dialog Box

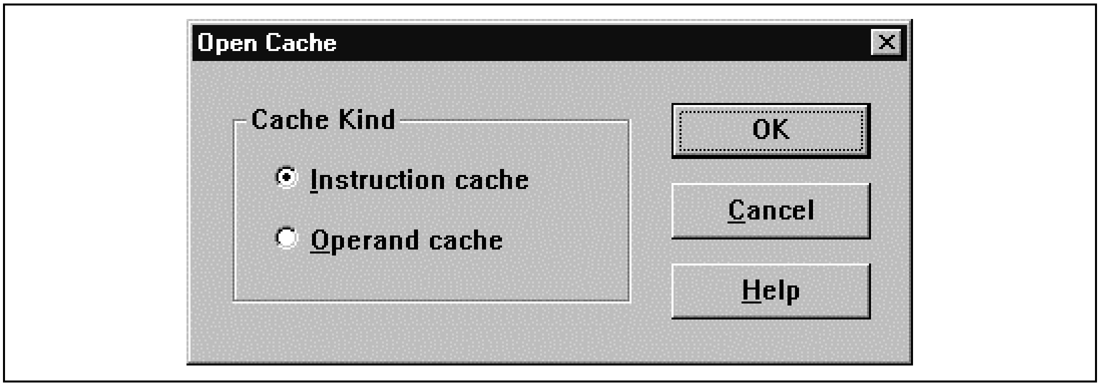


Figure 5.61 Open Cache Dialog Box

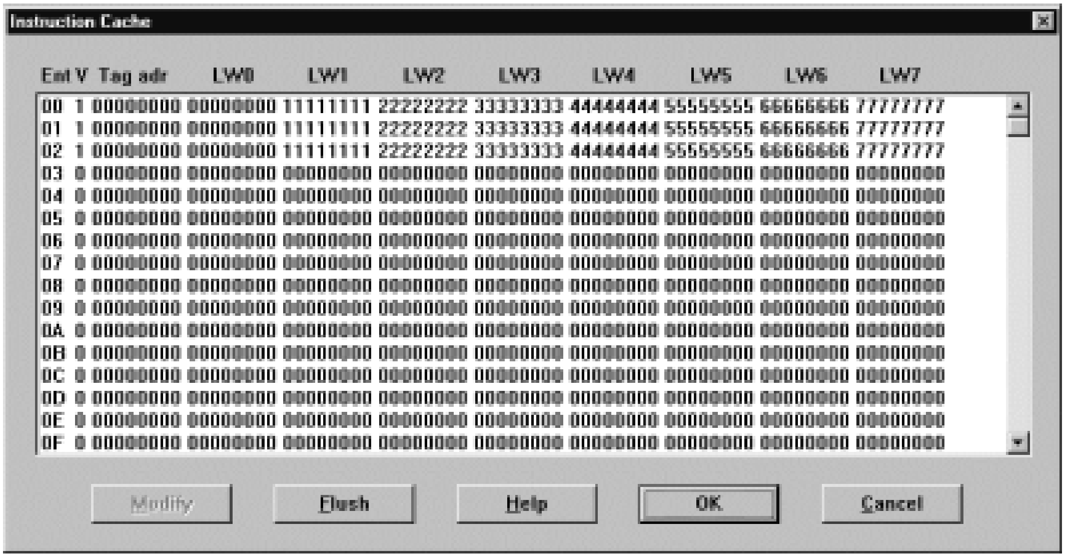
This dialog box selects the cache to be displayed. This dialog box is provided only for the SH-4 series.

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.

5.53 Instruction Cache Dialog Box



```
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 5.62 Instruction Cache Dialog Box

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

The following items are displayed.

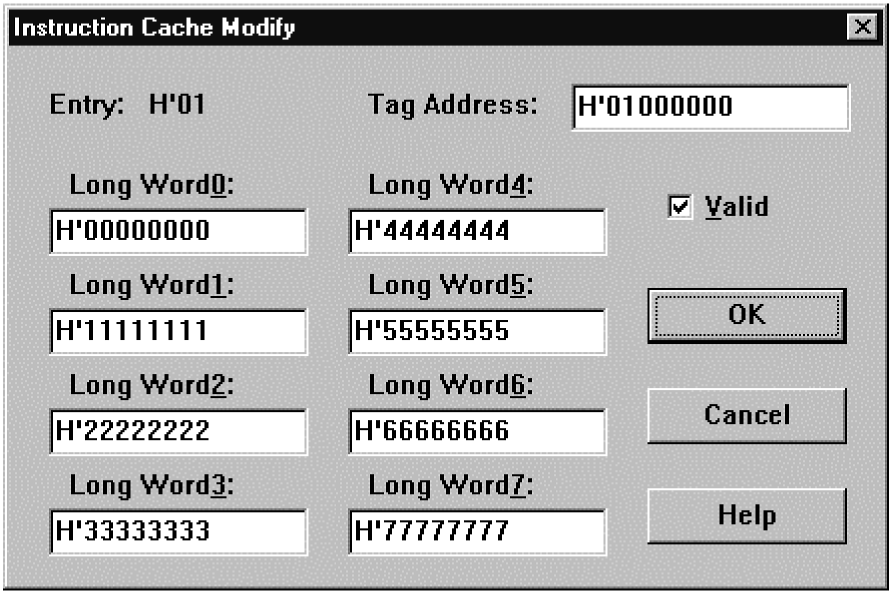
[Ent]	Entry number in the IC (H'00 to H'FF).
[V]	Validity bit. When this bit is 1, the entry is valid.
[Tag adr]	Tag address.
[LW0] to [LW7]	Longword data 0 to 7 stored in IC entries.

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.

5.54 Instruction Cache Modify Dialog Box



Entry: H'01

Tag Address: H'01000000

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 5.63 Instruction Cache Modify Dialog Box

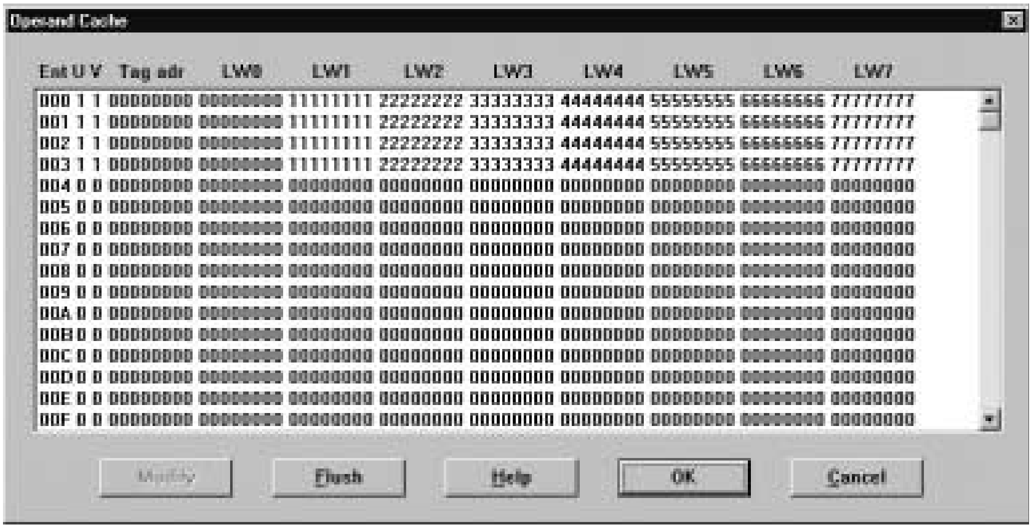
This dialog box modifies the IC contents of the displayed entry. This dialog box is provided only for the SH-4 series.

The following items can be displayed or specified.

[Entry]	Displays the entry number selected by the Instruction Cache dialog box.
[Tag Address]	Tag address. A physical address (longword) 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.

5.55 Operand Cache Dialog Box



```
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 1 1 00000000 00000000 11111111 22222222 33333333 44444444 55555555 66666666 77777777
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 5.64 Operand Cache Dialog Box

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

The following items are displayed.

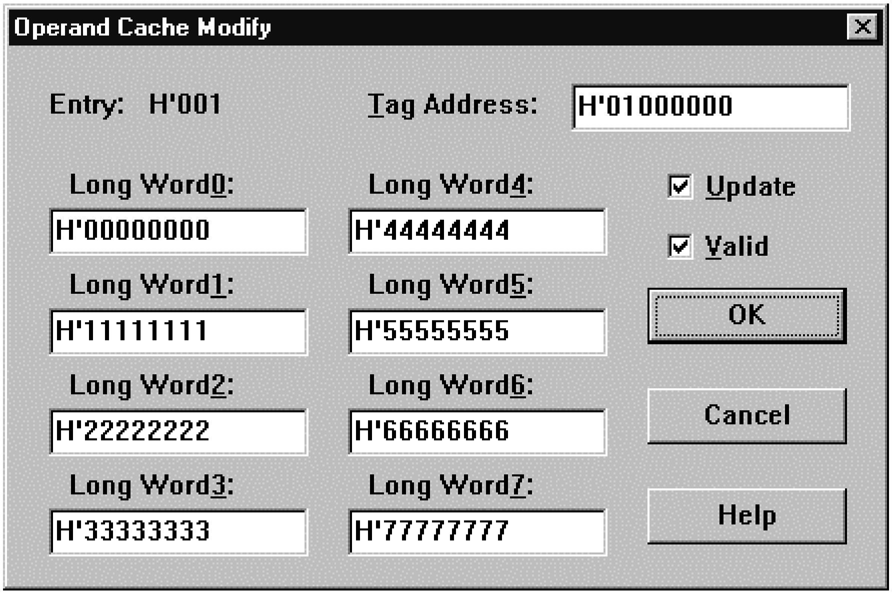
[Ent]	Entry number in the OC (H'000 to H'1FF).
[U]	Update bit. When this bit is 1, the entry has been written to.
[V]	Validity bit. When this bit is 1, the entry is valid.
[Tag adr]	Tag address.
[LW0] to [LW7]	Longword data 0 to 7 stored in OC entries.

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.

5.56 Operand Cache Modify Dialog Box


Operand Cache Modify

Entry: H'001

Tag Address: H'01000000

Long Word0:	Long Word4:
H'00000000	H'44444444

Long Word1:	Long Word5:
H'11111111	H'55555555

Long Word2:	Long Word6:
H'22222222	H'66666666

Long Word3:	Long Word7:
H'33333333	H'77777777

☑ Update

☑ Valid

Figure 5.65 Operand Cache Modify Dialog Box

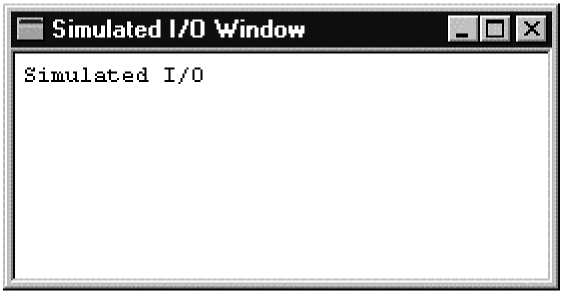
This dialog box modifies the OC contents of the displayed entry. This dialog box is provided only for the SH-4 series.

The following items can be specified.

[Entry]	Entry number selected by the Operand Cache dialog box.
[Tag Address]	Tag address. A physical address (longword) 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.

5.57 Simulated I/O Window



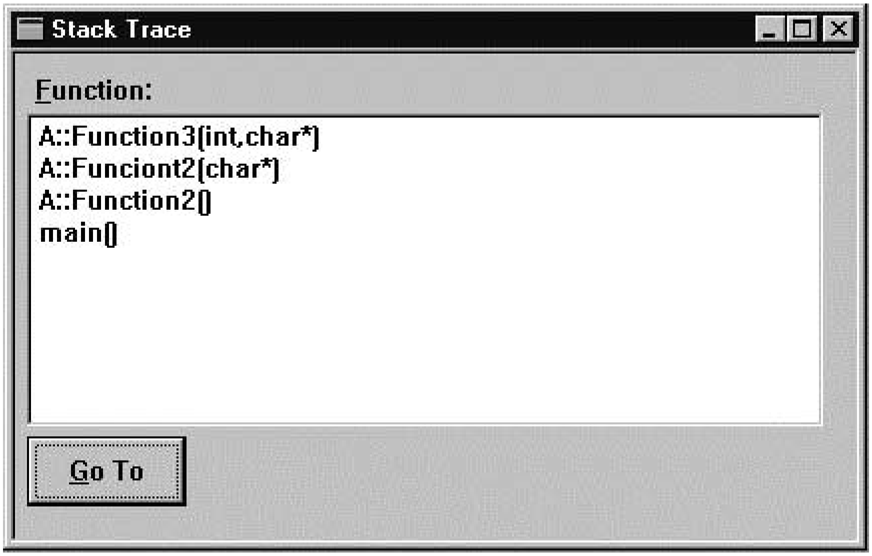
Simulated I/O

Figure 5.66 Simulated I/O Window

This window is for standard I/O and file I/O system calls from the user program.

For the I/O processing, refer to section 2.11, Standard I/O and File I/O Processing.

5.58 Stack Trace Window



A::Function3(int,char*)
A::Funciont2(char*)
A::Function2()
main()

Figure 5.67 Stack Trace Window

This window displays the function call history except the functions called after an interrupt.

Selecting a displayed function and clicking the **[Go To]** button, or double-clicking a displayed function displays the corresponding source program in the **Program** window.

Section 6 Simulator/Debugger Commands

This section describes the commands that can be used only in the simulator/debugger. For other commands, refer to the Hitachi Debugging Interface User's Manual.

Table 6.1 lists the simulator/debugger commands.

Table 6.1 Simulator/Debugger Commands

Command Name	Abbrevi-ation	Function
BREAKPOINT	BP	Sets a breakpoint at an instruction address
BREAK_ACCESS	BA	Specifies a memory range access as a break condition
BREAK_CLEAR	BC	Deletes breakpoints
BREAK_DATA	BD	Specifies a memory data value as a break condition
BREAK_DISPLAY	BI	Displays a list of breakpoints
BREAK_ENABLE	BE	Enables or disables a breakpoint
BREAK_REGISTER	BR	Specifies a register data as a break condition
BREAK_SEQUENCE	BS	Sets sequential breakpoints
MAP_SET	MS	Allocates a memory area
TRACE_ACQUISITION	TA	Enables or disables trace information acquisition
ANALYSIS_RANGE	AR	Specifies or displays functions for performance analysis

Details of each command are provided in the format shown in figure 6.1.

6.1 BREAKPOINT [BP] **Sets a breakpoint at an instruction address**
(1) (2) (3)

Syntax: (4)

Parameters: (5)

Examples: (6)

Figure 6.1 Command Description Format

(1) Command name

(2) Abbreviated command name

(3) Command function

(4) Command input syntax
  Δ: Indicates one or more spaces or tabs
  [A]: Indicates that A can be omitted.

(5) Command parameters

(6) Command usage examples

6.1 BREAKPOINT [BP] Sets a breakpoint at an instruction address

Syntax: BREAKPOINTΔaddress[Δcount]

Parameters:	address	The address of a breakpoint
	count	The number of times the instruction at the specified address is to be fetched. The default is 1.

Examples:	BREAKPOINT 0 2	A break occurs when an attempt is made to execute the instruction at address 0 for the second time.
	BP C0	A break occurs when an attempt is made to execute the instruction at address C0.

6.2 BREAK_ACCESS [BA] Specifies a memory range access as a break condition

Syntax: BREAK_ACCESSΔstart address[Δend address] [Δaccess type]

Parameters:

start address — The start address of the break memory range, to which access stops program execution.

end address — The end address of the break memory range, to which access stops program execution. If omitted, the end address is assumed to be the same as the start address.

access type — Access type

R:	A break occurs when the range is read.
W:	A break occurs when the range is written to.
RW:	A break occurs when the range is read or written to.

The default is RW(read or written to).

Examples:	BREAK_ACCESS 0 1000 W	A break occurs when the range from address 0 to address 1000 is written to.
	BA FFFF	A break occurs when address FFFF is accessed.

6.3 BREAK_CLEAR [BC] Deletes breakpoints

Syntax: BREAK_CLEAR [Δindex]

Parameters:

index	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 canceled.

6.4 BREAK_DATA [BD] Specifies a memory data value as a break condition

Syntax: BREAK_DATAΔaddressΔdata [Δsize] [Δoption]

Parameters:

address — The address where the break condition is checked.

data — Access data

size — Size

B:	Byte
W:	Word
L:	Longword
S:	Single-precision floating-point data
D:	Double-precision floating-point data

The default is L (longword)

option — Match or mismatch of data

EQ:	A break occurs when the data matches the specified value.
NE:	A break occurs when the data does not match the specified value.

The default is EQ (match).

Examples:

BREAK_DATA 0 100 L EQ	A break occurs when 100 is written to memory address 0 in longword.
BD C0 FF B NE	A break occurs when a value other than FF is written to memory address C0 in byte.
BD 4000 1000	A break occurs when 1000 is written to memory address 4000 in longword.

6.5 BREAK_DISPLAY [BI] Displays a list of breakpoints

Syntax:	BREAK_DISPLAY	
Examples:	BREAK_DISPLAY	A list of breakpoints is displayed.

6.6 BREAK_ENABLE [BE] Enables or disables a breakpoint

Syntax: BREAK_ENABLE Δenable/disable flag [Δindex]

Parameters:	enable/disable flag	Enabling or disabling of a breakpoint. E: Enable D: Disable
	index	Index of a breakpoint. If the index is omitted, all breakpoints are enabled or disabled.

Examples:	BREAK_ENABLE D 0	The first breakpoint is disabled.
	BE E	All breakpoints are enabled.

6.7 BREAK_REGISTER [BR] **Specifies a register data as a break condition**

Syntax: BREAK_REGISTER Δregister name [ΔdataΔsize] [Δaccess type]

Parameters:

register name	Register name
data	Data for a break condition If no data is specified, a break occurs when any value is written to the specified register.
size	Access size B: Byte W: Word L: Longword S: Single-precision floating-point data D: Double-precision floating-point data 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.
access type	Data match or mismatch EQ: A break occurs when the data matches the specified value. NE: A break occurs when the data does not match the specified value. The default is EQ (match).

Examples:

BREAK_REGISTER R0 FFFF W EQ	A break occurs when the low-order two bytes of the R0 register change to FFFF.
BR R10	A break occurs when the R10 register is written to.

6.8 BREAK_SEQUENCE [BS] **Sets sequential breakpoints**

Syntax: BREAK_SEQUENCE Δaddress1 [Δaddress2 [Δaddress 3 [...]]]

Parameters:

address1—address8	Addresses of sequential breakpoints. Up to eight addresses can be specified.

Examples:

BREAK_SEQUENCE 1000 2000	A break occurs when addresses 1000 and 2000 are passed.
BS 1000	A break occurs when address 1000 is executed.

6.9 MAP_SET [MS] **Allocates a memory area**

Syntax: MAP_SET Δstart address [Δend address] [Δaccess type]

Parameters:

start address	Start address of a memory area.
end address	End address of a memory area. If omitted, the end address is assumed to be the same as the start address.
access type	Access type R: Read W: Write RW: Read/write The default is RW (Read/write).

Examples:

MAP_SET 0000 3FFF RW	A read/write-enabled area is allocated to addresses 0000 to 3FFF.
MS 5000	A read/write-enabled area is allocated to address 5000.

6.10 TRACE_ACQUISITION [TA] **Enables or disables trace information acquisition**

Syntax: TRACE_ACQUISITION Δenable/disable flag

Parameters:

enable/disable flag	Enabling or disabling 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.

6.11 ANALYSIS_RANGE [AR] Specifies or displays functions for performance analysis

Syntax: ANALYSIS_RANGE [Δfunction]

Parameters:

function	Function to be evaluated. If omitted, the function that has been specified is assumed.

Examples:

ANALYSIS_RANGE sort	The sort function is specified to be evaluated.
AR	The function that has been specified is displayed.

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 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 was normally terminated.
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 halted.

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 2n. • Longword data was accessed to an address other than 4n. • 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.
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: • 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.
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 general invalid instruction. Correct the user program to prevent the error from occurring.
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.

Table 7.2 Error Messages (cont)

Message	Contents
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 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.
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 Multiple Hit	Multiple instruction TLB entries were hit when a virtual address was accessed in memory. TLB is not correctly set. Modify TLB contents and user program (handler routine).
Instruction TLB Protection Violation	An instruction TLB protection exception occurred during memory access. Take necessary procedures such as updating the TLB contents.
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.
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.

Table 7.2 Error Messages (cont)

Message	Contents
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 occurred during simulation.
Unified TLB Miss	A unified TLB miss occurred during memory access. Take necessary procedures such as updating the TLB contents.
Unified TLB Multiple Hit	Multiple unified TLB entries were hit when a virtual address was accessed in memory. 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 TLB contents.

SH Series Simulator/Debugger User’s Manual

Publication Date:	1st Edition, November 1998
Published by:	Electronic Devices Sales & Marketing Group Semiconductor & Integrated Circuits Group Hitachi, Ltd.
Edited by:	Technical Documentation Group UL Media Co., Ltd.

Copyright © Hitachi, Ltd., 1998. All rights reserved. Printed in Japan.