

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

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

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

April 1<sup>st</sup>, 2010  
Renesas Electronics Corporation

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

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

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
  - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
  - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
  - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

# H8/300H Tiny Compact Emulator Debugger

User's Manual

Renesas Microcomputer Development  
Environment System

Active X, Microsoft, MS-DOS, Visual Basic, Visual C++, Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

IBM and AT are registered trademarks of International Business Machines Corporation.

Intel and Pentium are registered trademarks of Intel Corporation.

Adobe and Acrobat are registered trademarks of Adobe Systems Incorporated.

All other brand and product names are trademarks, registered trademarks or service marks of their respective holders.

#### **Keep safety first in your circuit designs!**

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

#### **Notes regarding these materials**

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

For inquiries about the contents of this document or product, fill in the text file the installer generates in the following directory and email to your local distributor.

\\SUPPORT\Product-name\SUPPORT.TXT

Renesas Tools Homepage <http://www.renesas.com/en/tools>

# Overview

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

This help explains the function as a "debugger" of High-performance Embedded Workshop.

# Target System

The Debugger operates on the compact emulator system.

# Supported CPU

This help explains the debugging function corresponding to the following CPUs.

- H8/300H Tiny Series

**Note:**

In this help, the information which depends on this CPU is described as "for H8/300H Tiny".

(blank)

<b>1. Features</b>	<b>3</b>
1.1 Real-Time RAM Monitor Function.....	3
1.2 Break Functions.....	3
1.2.1 Software Break .....	3
1.2.2 Hardware Break .....	3
1.3 Real-Time Trace Function .....	3
1.4 GUI Input/Output Function .....	4
<b>2. About the Compact Emulator</b>	<b>5</b>
2.1 Communication method.....	5
2.2 Function table .....	5
<b>3. Before starting the debugger</b>	<b>6</b>
3.1 Communication method by emulator.....	6
3.1.1 USB Interface .....	6
3.2 Download of Firmware.....	6
3.3 Setting before emulator starts.....	7
3.3.1 USB communication with the Emulator .....	7
3.3.1.1 Install of USB device driver .....	7
<b>4. Preparation before Use</b>	<b>8</b>
4.1 Workspaces, Projects, and Files .....	8
4.2 Starting the High-performance Embedded Workshop .....	9
4.2.1 Creating a New Workspace (Toolchain Used) .....	11
4.2.1.1 Step1 : Creation of a new workspace .....	11
4.2.1.2 Step2 : Setting for the Toolchain .....	12
4.2.1.3 Step 3: Selecting of the Target Platform .....	13
4.2.1.4 Step4 : Setting the Configuration File Name.....	14
4.2.1.5 Step5 : The check of a created file name .....	15
4.2.2 Creating a New Workspace (Toolchain Not Used).....	16
4.2.2.1 Step1 : Creation of a new workspace .....	16
4.2.2.2 Step 2: Selecting of the Target Platform .....	17
4.2.2.3 Step3 : Setting the Configuration File Name.....	18
4.2.2.4 Step4 : Registering the Load modules to be downloaded.....	19
4.3 Starting the Debugger .....	20
4.3.1 Connecting the Emulator .....	20
4.3.2 Ending the Emulator.....	20
<b>5. Setup the Debugger</b>	<b>21</b>
5.1 Init Dialog.....	21
5.1.1 MCU Tab.....	22
5.1.1.1 Specifying the MCU file .....	22
5.1.1.2 Setting of the Communication Interface .....	22
5.1.1.3 Executing Self-Check.....	23
5.1.1.4 Choosing to use or not to use the trace point setting function .....	23
5.1.1.5 Recording only the valid cycles in trace memory .....	24
5.1.2 Debugging Information Tab .....	24
5.1.2.1 display the compiler used and its object format.....	24
5.1.2.2 Specify the Storing of Debugging Information .....	24
5.1.3 Emulator Tab.....	25
5.1.3.1 Specify the Target Clock .....	25
5.2 MCU Setting Dialog (H8/300H Tiny Debugger) .....	26
5.2.1 MCU Tab.....	26

5.2.1.1 Select the Processor Mode .....	27
5.2.1.2 Inspecting the MCU status .....	27

## Tutorial

29

<b>6. Tutorial</b> .....	<b>31</b>
6.1 Introduction .....	31
6.2 Usage .....	32
6.2.1 Step1 : Starting the Debugger .....	32
6.2.1.1 Preparation before Use .....	32
6.2.1.2 Setup the Debugger .....	32
6.2.2 Step2 : Checking the Operation of RAM .....	33
6.2.2.1 Checking the Operation of RAM .....	33
6.2.3 Step3 : Downloading the Tutorial Program .....	34
6.2.3.1 Downloading the Tutorial Program .....	34
6.2.3.2 Displaying the Source Program .....	34
6.2.4 Step4 : Setting a Breakpoint .....	35
6.2.4.1 Setting a Software Breakpoint .....	35
6.2.5 Step5 : Executing the Program .....	36
6.2.5.1 Resetting of CPU .....	36
6.2.5.2 Executing the Program .....	36
6.2.5.3 Reviewing Cause of the Break .....	37
6.2.6 Step6 : Reviewing Breakpoints .....	38
6.2.6.1 Reviewing Breakpoints .....	38
6.2.7 Step7 : Viewing Register .....	39
6.2.7.1 Viewing Register .....	39
6.2.7.2 Setting the Register Value .....	39
6.2.8 Step8 : Viewing Memory .....	40
6.2.8.1 Viewing Memory .....	40
6.2.9 Step9 : Watching Variables .....	41
6.2.9.1 Watching Variables .....	41
6.2.9.2 Registering Variable .....	42
6.2.10 Step10 : Stepping Through a Program .....	43
6.2.10.1 Executing [Step In] Command .....	43
6.2.10.2 Executing [Step Out] Command .....	43
6.2.10.3 Executing [Step Over] Command .....	44
6.2.11 Step11 : Forced Breaking of Program Executions .....	45
6.2.11.1 Forced Breaking of Program Executions .....	45
6.2.12 Step12 : Displaying Local Variables .....	46
6.2.12.1 Displaying Local Variables .....	46
6.2.13 Step13 : Stack Trace Function .....	47
6.2.13.1 Reference the function call status .....	47
6.2.14 What Next? .....	48

## Reference

49

<b>7. Windows/Dialogs</b> .....	<b>51</b>
7.1 RAM Monitor Window .....	52
7.1.1 Extended Menus .....	53
7.1.2 Setting the RAM monitor area .....	54
7.1.2.1 Changing the RAM Monitor Area .....	55
7.1.2.2 Adding RAM Monitor Areas .....	55

7.1.2.3 Deleting RAM Monitor Areas.....	55
7.2 ASM Watch Window .....	56
7.2.1 Extended Menus .....	57
7.3 C Watch Window.....	58
7.3.1 Extended Menus .....	59
7.4 S/W Break Point Setting Window .....	60
7.4.1 Command Button.....	61
7.4.2 Setting and Deleting a Break Points from Editor(Source) Window .....	61
7.5 H/W Break Point Setting Window.....	62
7.5.1 Specify the Break Event.....	63
7.5.2 Specify the Combinatorial Condition.....	65
7.5.3 Command Button.....	66
7.6 Trace Point Setting Window.....	67
7.6.1 Specify the Trace Event.....	68
7.6.2 Specify the Combinatorial Condition.....	70
7.6.3 Specify the Trace Range.....	71
7.6.4 Specify the Trace Write Condition.....	72
7.6.5 Command Button.....	72
7.7 Trace Window.....	73
7.7.1 Configuration of Bus Mode.....	73
7.7.2 Configuration of Disassemble Mode .....	75
7.7.3 Configuration of Data Access Mode .....	76
7.7.4 Configuration of Source Mode.....	77
7.7.5 Extended Menus .....	78
7.8 GUI I/O Window.....	79
7.8.1 Extended Menus .....	80
<b>8. Target Dependent Commands</b>	<b>81</b>
<b>9. Expressions</b>	<b>82</b>
9.1 Expressions .....	82
9.1.1 Constants .....	82
9.1.2 Symbols and Labels .....	82
9.1.3 Register Variables .....	83
9.1.4 String Literals .....	83
9.1.5 Operators .....	83
9.2 Writing C/C++ Expressions .....	84
9.3 Display Format of C/C++ Expressions .....	88
<b>10. Display the Cause of the Program Stoppage</b>	<b>92</b>
<b>11. Attention</b>	<b>93</b>
11.1 Common Attention.....	93
11.1.1 File operation on Windows.....	93
11.1.2 Area where software breakpoint can be set .....	93
11.1.3 Get or set C variables .....	93
11.1.4 Function name in C++.....	94
11.1.5 Debugging multi modules .....	94
11.1.6 Synchronized debugging.....	94
11.1.7 Compact Emulator reset switch.....	94
11.2 Attention of the H8/300H Tiny Debugger.....	95
11.2.1 Map of stack area used by the compact emulator .....	95
11.2.2 Hardware break function .....	95
11.2.3 Hardware Event .....	95

[MEMO]

# **Starting/Setup of Debugger**

(blank)

# 1. Features

This debugger have the following functions.

## 1.1 Real-Time RAM Monitor Function

This function allows you to inspect changes of memory contents without impairing the realtime capability of target program execution.

The compact emulator system has 1 Kbytes of RAM monitor area which can be located in any contiguous address location or in 4 separate blocks comprised of 256 bytes each.

## 1.2 Break Functions

### 1.2.1 Software Break

This function causes the target program to stop immediately before executing the instruction at a specified address.

Up to 64 breakpoints can be set. If multiple breakpoints are set, the program breaks at one of the breakpoints that is reached.

### 1.2.2 Hardware Break

This function causes the target program to stop upon detecting a data read/write to memory, instruction execution, or the rising/falling edge of the input signal fed from an external trace cable.

The contents of events that can be set vary with each target MCU. Specified hardware break events can be used in one of the following combinations:

- Break when all specified break points are effected.(And)
- Break when all specified break points are effected simultaneously.(And(Same Time))
- Break when any one of the specified break points is effected.(Or)

## 1.3 Real-Time Trace Function

This function records a target program execution history.

Up to 64K cycles of execution history can be recorded. This record allows inspecting the bus information, executed instructions, and source program execution path for each cycle.

---

## 1.4 GUI Input/Output Function

This function simulates the user target system's key input panel (buttons) and output panel on a window.

Buttons can be used for the input panel, and labels (strings) and LEDs can be used for the output panel.

## 2. About the Compact Emulator

The compact emulator is a small emulator equipped with the debugging function needed for full-scale development, such as real-time trace and a hardware break, though it is a handy price and a small body.

### 2.1 Communication method

The supported communication methods are as follows.

- The R0E436640CPE00 is supporting USB as a communication interface.

### 2.2 Function table

The supported functions are as follows.

<b>Function</b>	<b>Compact Emulator</b>
SW Break	64 points
HW Break	2 points
Real-Time Trace	64K Cycles
RAM Monitor	1K bytes (256bytes x 4blocks) area
Time Measurement	Go to Stop

---

## 3. Before starting the debugger

Before starting the debugger, check the following contents:  
Communication method by emulator

### 3.1 Communication method by emulator

#### 3.1.1 USB Interface

- The supported host computer OS is Windows Me/98/2000/XP. USB communication cannot be used in any other OS.
- Compliant with USB Standard 1.1.
- Connections via USB hub are not supported.
- By connecting the host computer and the emulator with USB cable, it is possible to install the supported device drivers using a wizard.
- The necessary cable is included with the emulator.

### 3.2 Download of Firmware

It is necessary to download the firmware which corresponds to connected Compact Emulator when the debugger is started to the emulator.

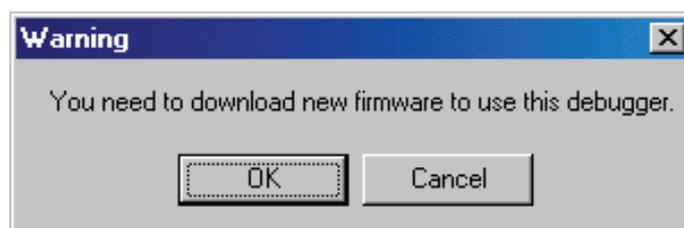
- Your emulator has unexpected firmware.
- You have setup the debugger for the first time.
- You have upgraded emulator debugger.

Press the system reset switch within two seconds after powering up the Compact Emulator to establish the maintenance mode.

This debugger searches the version of the firmware downloaded to the emulator at start. Also when the firmware downloaded to the emulator is of old version, a mode which drives this debugger to download firmware is set.

When this debugger gets started while the emulator is set in the mode which drives the debugger to download firmware forcedly, the following dialog is opened at start.

Click the OK button to download the firmware.



## 3.3 Setting before emulator starts

### 3.3.1 USB communication with the Emulator

Connection of USB devices is detected by Windows' Plug & Play function. The device driver needed for the connected USB device is automatically installed. For details, see "Install of USB Device Driver".

#### 3.3.1.1 Install of USB device driver

The USB devices connected are detected by Windows' Plug & Play function. The installation wizard for USB device drivers starts after the device had been detected. The following shows the procedure for installing the USB device drivers.

1. Connect the host computer and the emulator with USB cable.
2. Set the emulator's communication interface switch to the "USB" position. Then turn on the power to the emulator.
3. The dialog box shown below appears.



Go on following the wizard, and a dialog box for specifying the setup information file (inf file) is displayed. Specify the musbdrv.inf file stored in a location below the directory where this debugger is installed.

#### ATTENTION

- Before the USB device drivers can be installed, the debugger you use must already be installed. Install this debugger first.
- USB communication can be used only in Windows Me/98/2000/XP, and cannot be used in any other OSs.
- When using Windows 2000/XP, a user who install the USB device driver need administrator rights.
- During installation, a message may be output indicating that the device driver proper musbdrv.sys cannot be found. In this case, specify the musbdrv.sys which is stored in the same directory as is the musbdrv.inf file.

---

## 4. Preparation before Use

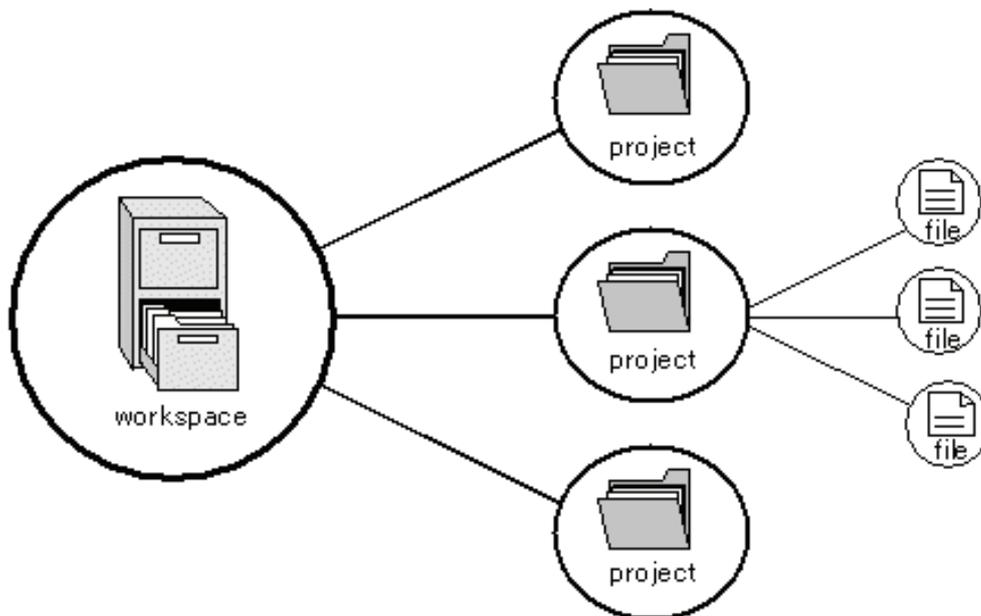
Please run the High-performance Embedded Workshop and connect the emulator. Starting the High-performance Embedded Workshop

In addition, in order to debug with this product, it is necessary to create a workspace.

### 4.1 Workspaces, Projects, and Files

Just as a word processor allows you to create and modify documents, this product allows you to create and modify workspaces.

A workspace can be thought of as a container of projects and, similarly, a project can be thought of as a container of project files. Thus, each workspace contains one or more projects and each project contains one or more files.



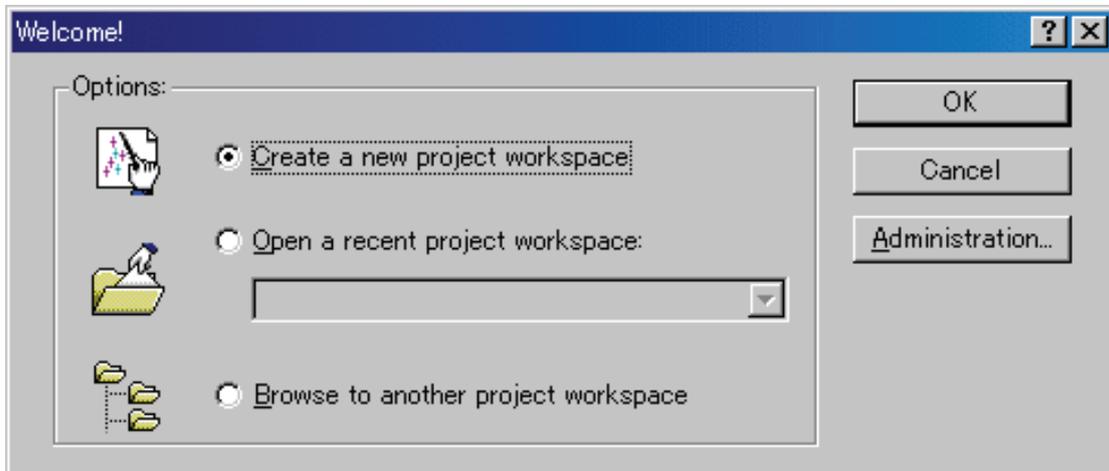
Workspaces allow you to group related projects together. For example, you may have an application that needs to be built for different processors or you may be developing an application and library at the same time. Projects can also be linked hierarchically within a workspace, which means that when one project is built all of its "child" projects are built first.

However, workspaces on their own are not very useful, we need to add a project to a workspace and then add files to that project before we can actually do anything.

## 4.2 Starting the High-performance Embedded Workshop

Activate the High-performance Embedded Workshop from [Programs] in the [Start] menu.

The [Welcome!] dialog box is displayed.



In this dialog box, A workspace is created or displayed.

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

In the case of Selecting an Existing Workspace, select [Open a recent project workspace] or [Browse to another project workspace] radio button and select the workspace file (.hws).

Please refer to the following about the method to create a new workspace.

- Creating a New Workspace (Toolchain Used)  
0

---

- Creating a New Workspace (Toolchain Not Used)

- \* When debugging the existing load module file with this product, a workspace is created by this method.

The method to create a new workspace depends on whether a toolchain is or is not in use. Note that this product does not include a toolchain. Use of a toolchain is available in an environment where the C/C++ compiler package for the CPU which you are using has been installed.

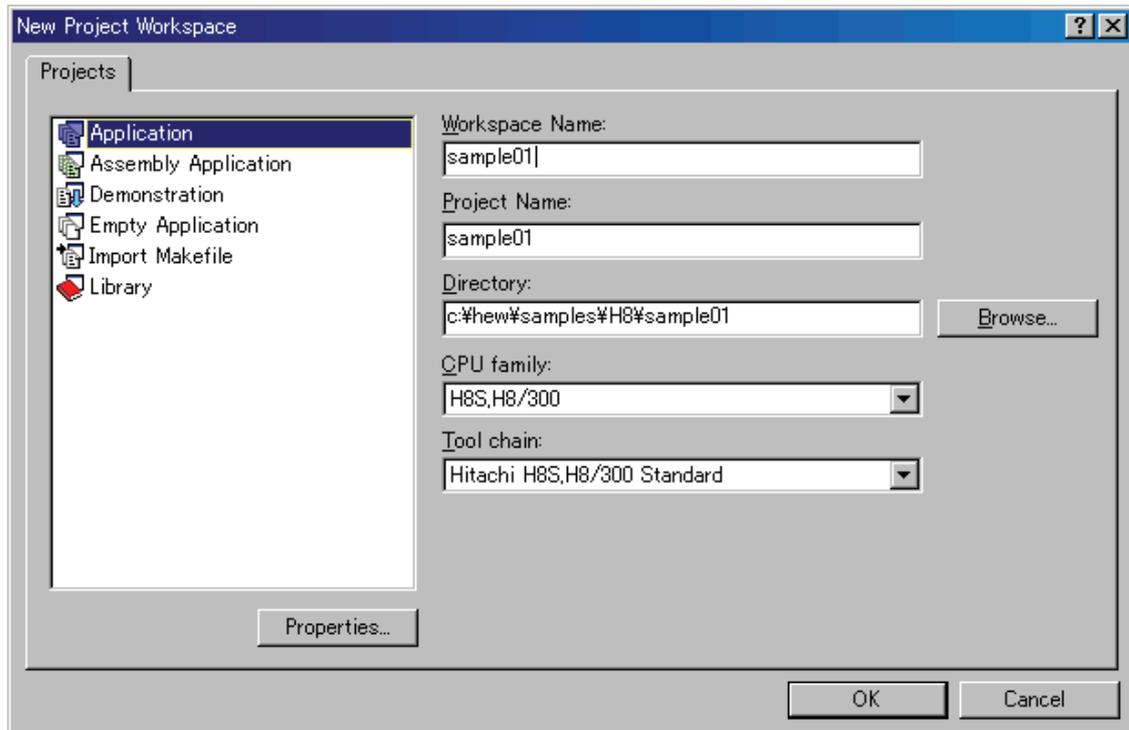
For details on this, refer to the manual attached to your C/C++ compiler package.

## 4.2.1 Creating a New Workspace (Toolchain Used)

### 4.2.1.1 Step1 : Creation of a new workspace

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

Creation of a new workspace is started. The following dialog box is displayed.

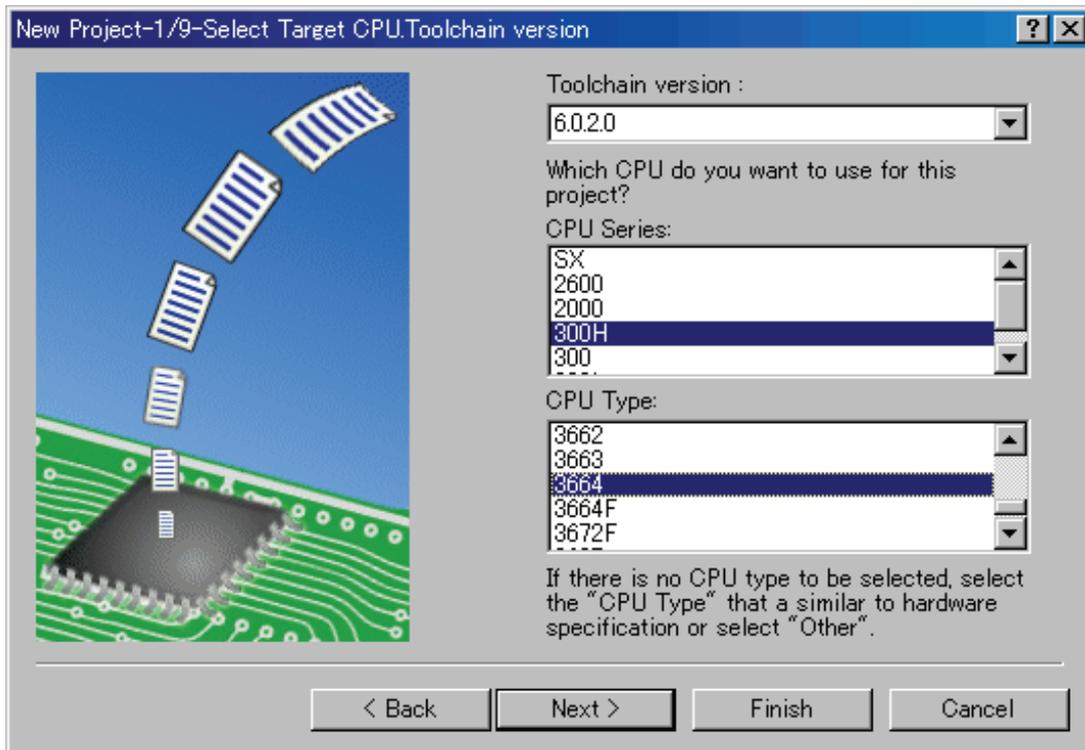


1. Select the target CPU family  
In the [CPU family] combo box, select the target CPU family.
2. Select the target toolchain  
In the [Tool chain] combo box, select the target toolchain name when using the toolchain.
3. Select the project type  
In the [Project type] list box, select the project type to be used. In this case, select "Application".  
(Please refer to the manual attached to your C/C++ compiler package about the details of the project type which can be chosen.)
4. Specify the workspace name and project name
  - In the [Workspace Name] edit box, enter the new workspace name.
  - In the [Project Name] edit box, enter the project name. When the project name is the same as the workspace name, it needs not be entered.
  - In the [Directory] edit box, enter the directory name in which the workspace will be created. Click the [Browse...] button to select a directory.

After a setting, click the [OK] button.

#### 4.2.1.2 Step2 : Setting for the Toolchain

A wizard for the project creation starts.



Here, the following contents are set.

- toolchain
- the setting for the real-time OS (when using)
- the setting for the startup file, heap area, stack area, and so on

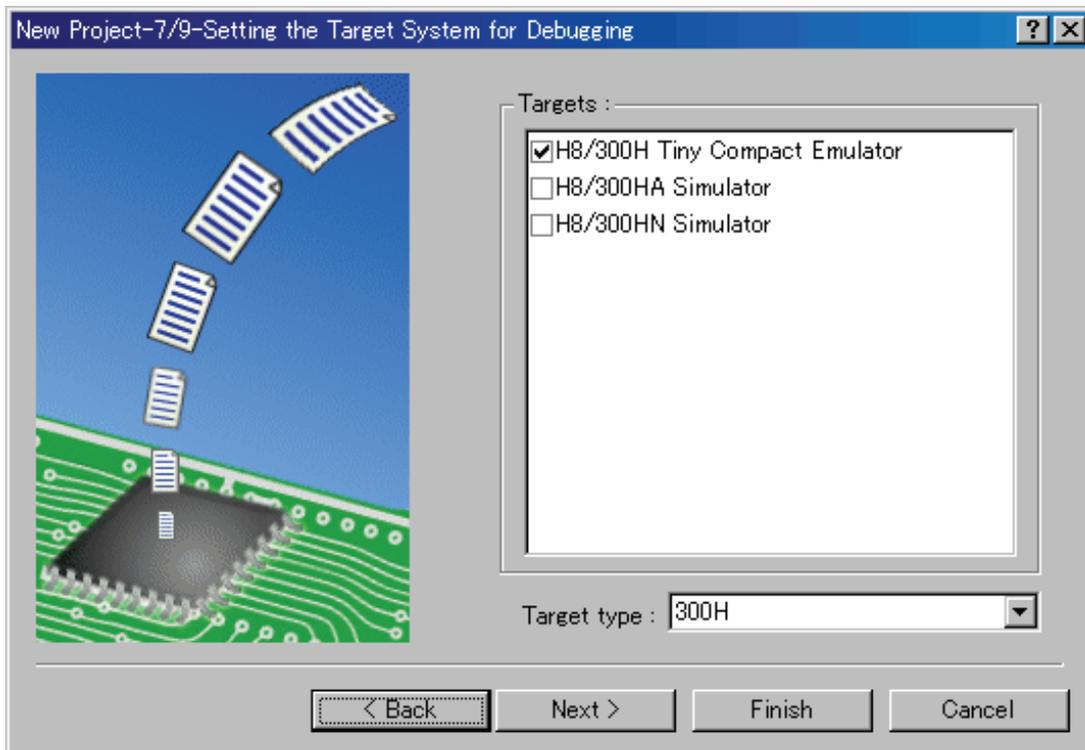
Please set required information and click the [Next] button.

The contents of a setting change with C/C++ compiler packages of use. Please refer to the manual attached to your C/C++ compiler package about the details of the contents of a setting.

#### 4.2.1.3 Step 3: Selecting of the Target Platform

Select the target system used for your debugging (emulator, simulator).

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



1. Selecting of the Target type  
In the [Target type] list box, select the target CPU type.
2. Selecting of the Target Platform  
In the [Targets] area, the target for the session file used when this debugger is activated must be selected here. Check the box against the target platform. (And choose other target as required.)

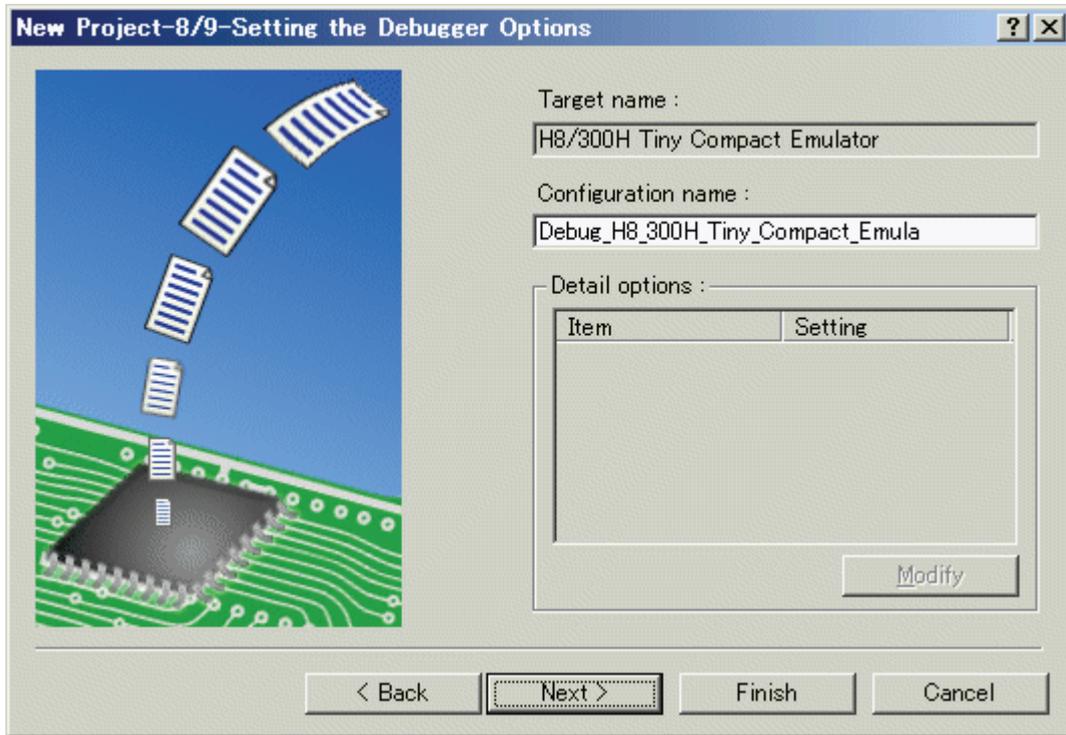
And click the [Next] button.

---

#### 4.2.1.4 Step4 : Setting the Configuration File Name

Set the configuration file name for each of the all selected target.

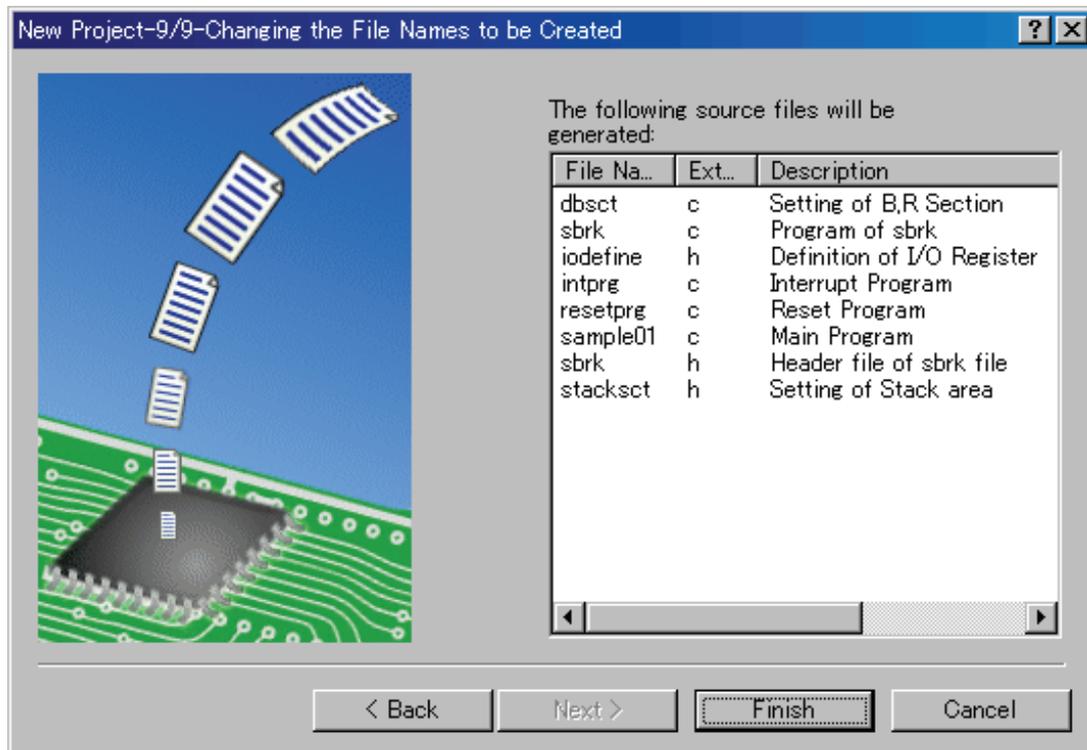
The configuration file saves the state of High-performance Embedded Workshop except for the target (emulator, simulator).



The default name is already set. If it is not necessary to change, please click the [next] button as it is.

#### 4.2.1.5 Step5 : The check of a created file name

Finally, confirm the file name you create. The files which will be generated by the High-performance Embedded Workshop are displayed. If you want to change the file name, select and click it then enter the new name.



This is the end of the emulator settings.

Exit the Project Generator following the instructions on the screen.

---

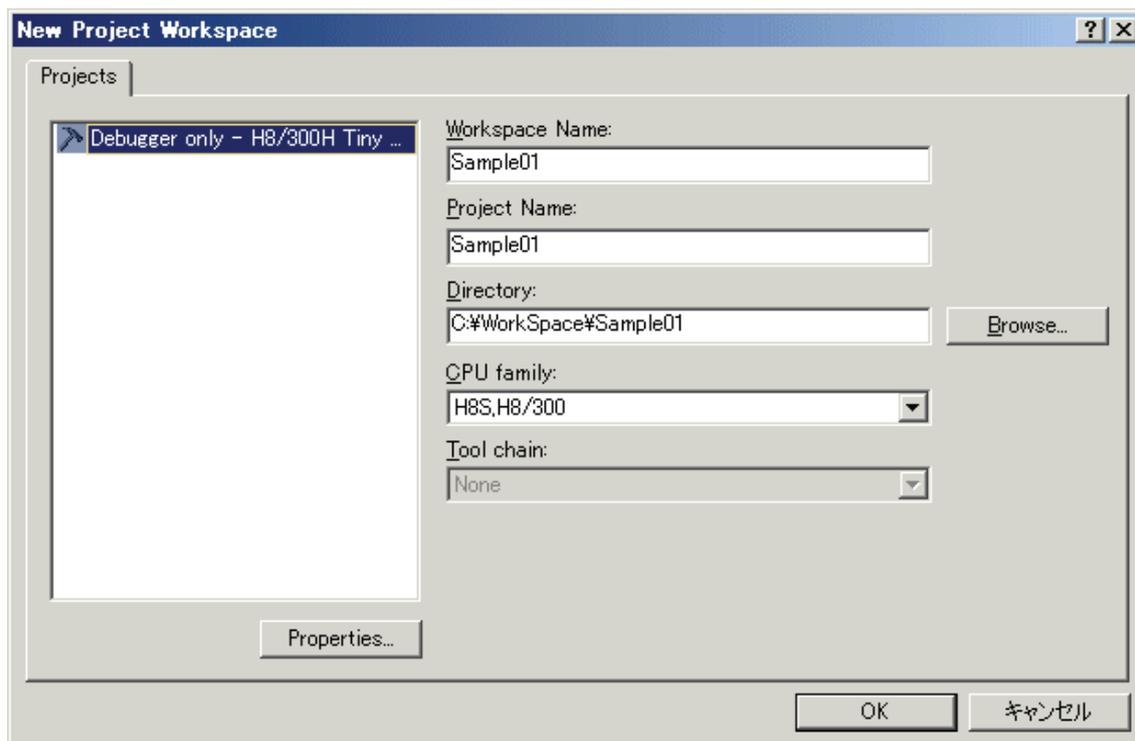
## 4.2.2 Creating a New Workspace (Toolchain Not Used)

When debugging the existing load module file with this product, a workspace is created by this method.

### 4.2.2.1 Step1 : Creation of a new workspace

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

Creation of a new workspace is started. The following dialog box is displayed.

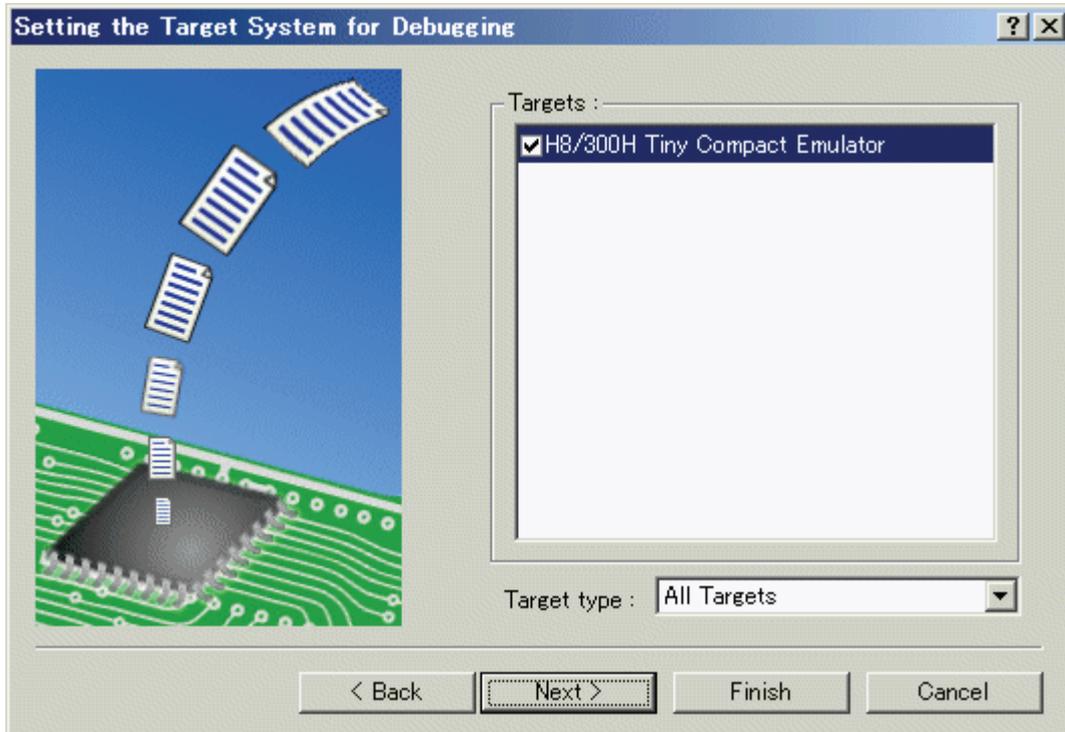


1. Select the target CPU family  
In the [CPU family] combo box, select the target CPU family.
2. Select the target toolchain  
In the [Tool chain] combo box, select "None". In this case, toolchain is not used. (When the toolchain has not been installed, the fixed information is displayed in this combo box.)
3. Select the project type  
When the toolchain is not used, it is displayed on a [Project Type] list box as "Debugger only - Target Name". Select it. (When two or more project types are displayed, please select one of them.)
4. Specify the workspace name and project name  
In the [Workspace Name] edit box, enter the new workspace name.
  - In the [Project Name] edit box, enter the project name. When the project name is the same as the workspace name, it needs not be entered.
  - In the [Directory] edit box, enter the directory name in which the workspace will be created. Click the [Browse...] button to select a directory.

After a setting, click the [OK] button.

#### 4.2.2.2 Step 2: Selecting of the Target Platform

Select the target system used for your debugging (emulator, simulator).  
A wizard starts and the following dialog box is displayed.



1. Selecting of the Target type  
In the [Target type] list box, select the target CPU type.
2. Selecting of the Target Platform  
In the [Targets] area, the target for the session file used when this debugger is activated must be selected here. Check the box against the target platform. (And choose other target as required.)

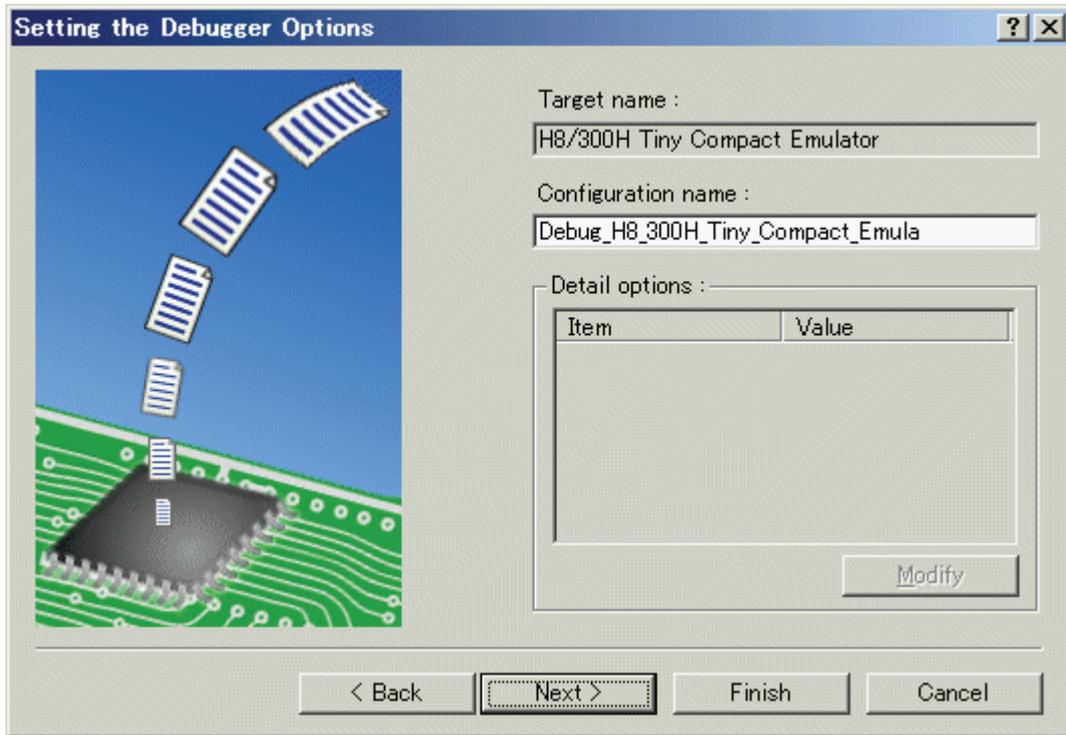
And click the [Next] button.

---

#### 4.2.2.3 Step3 : Setting the Configuration File Name

Set the configuration file name for each of the all selected target.

The configuration file saves the state of High-performance Embedded Workshop except for the target (emulator, simulator).



The default name is already set. If it is not necessary to change, please click the [next] button as it is.

This is the end of the emulator settings.

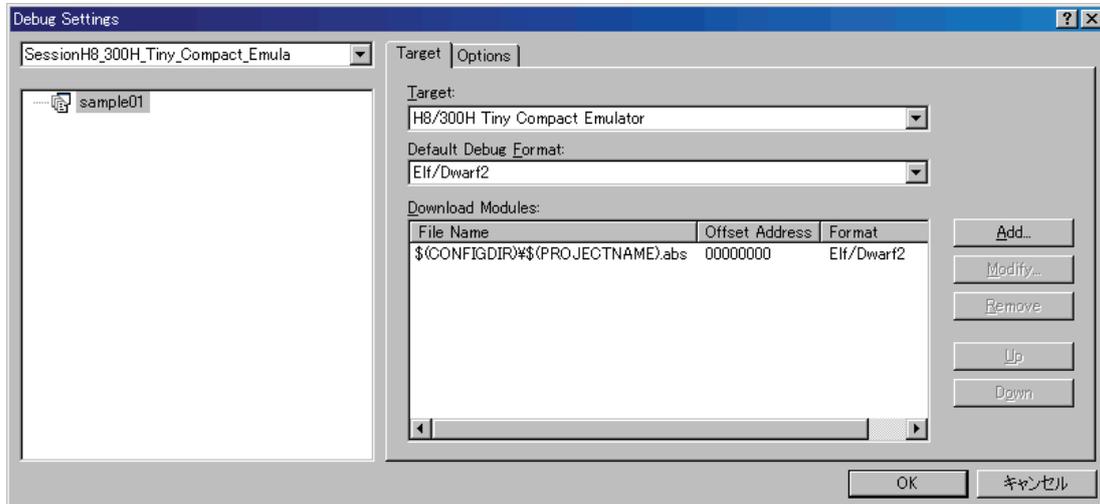
Exit the Project Generator following the instructions on the screen.

And the dialog for the setup of a debugger is also displayed at this time. If preparation of an emulator is completed, set up the debugger in this dialog box and connect with an emulator.

#### 4.2.2.4 Step4 : Registering the Load modules to be downloaded

Finally, register the load module file to be used.

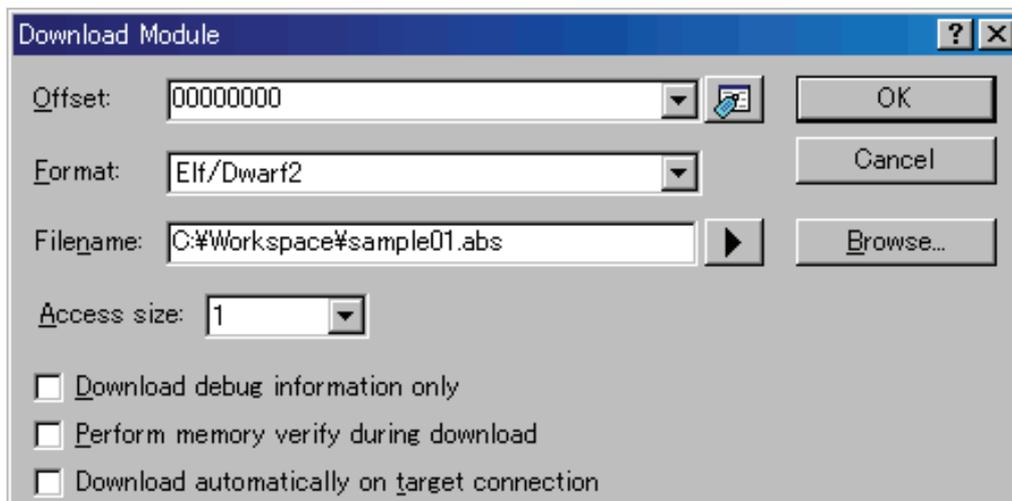
Select [Debug Settings...] from the [Debug] menu to open the [Debug Settings] dialog box.



1. Select the product name to be connected in the [Target] drop-down list box.
2. Select the format of the load module to be downloaded in the [Default Debug Format] drop-down list box.

Format Name	Contents
ELF/DWARF2	ELF/DWARF2 format file (When Using H8C)

3. Then register the corresponding download module in the [Download Modules] list box. A download module can be specified in the dialog opened with a [Add...] button.



- Enter the offset at which to load the download module in the [Offset] edit box.
- Select the format of the download module in the [Format] edit box. Please refer to the upper table about the format name of a download module.
- Enter the full path and filename of the download module in the [Filename] edit box.
- Specifies the access size for the current download module in the [Access size] list box.

After that, click the [OK] button.

---

## 4.3 Starting the Debugger

The debugging can be started by connecting with an emulator.

### 4.3.1 Connecting the Emulator

Connect the emulator by simply switching the session file to one in which the setting for the emulator use has been registered.

The session file is created by default. The session file has information about the target selected when a project was created.

In the circled list box in the following tool bars, select the session name including the character string of the target to connect.



After the session name is selected, the dialog box for setting the debugger is displayed and the emulator will be connected. Details)

### 4.3.2 Ending the Emulator

The emulator can be exited by using the following methods:

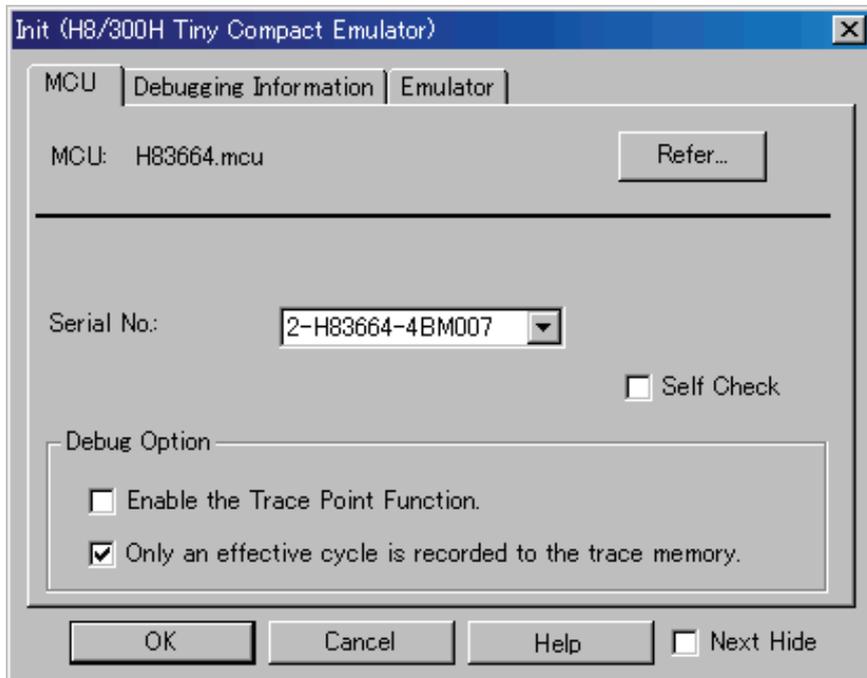
1. Selecting the "DefaultSession"  
Select the "DefaultSession" in the list box that was used at the time of emulator connection.
2. Exiting the High-performance Embedded Workshop  
Select [Exit] from the [File] menu. High-performance Embedded Workshop will be ended.

The message box, that asks whether to save a session, will be displayed when an emulator is exited. If necessary to save it, click the [Yes] button. If not necessary, click the [No] button.

## 5. Setup the Debugger

### 5.1 Init Dialog

The Init dialog box is provided for setting the items that need to be set when the debugger starts up. The contents set from this dialog box are also effective the next time the debugger starts. The data set in this dialog remains effective for the next start.



The tabs available on this dialog box vary with each product used. For details, click the desired tab name shown in the table below.

Tab Name	Contents
MCU	Specifies a MCU File, communication interface and debug options.
Debugging Information	Specifies compiler you use and debugging information handling.
Emulator	Specifies target clocks.

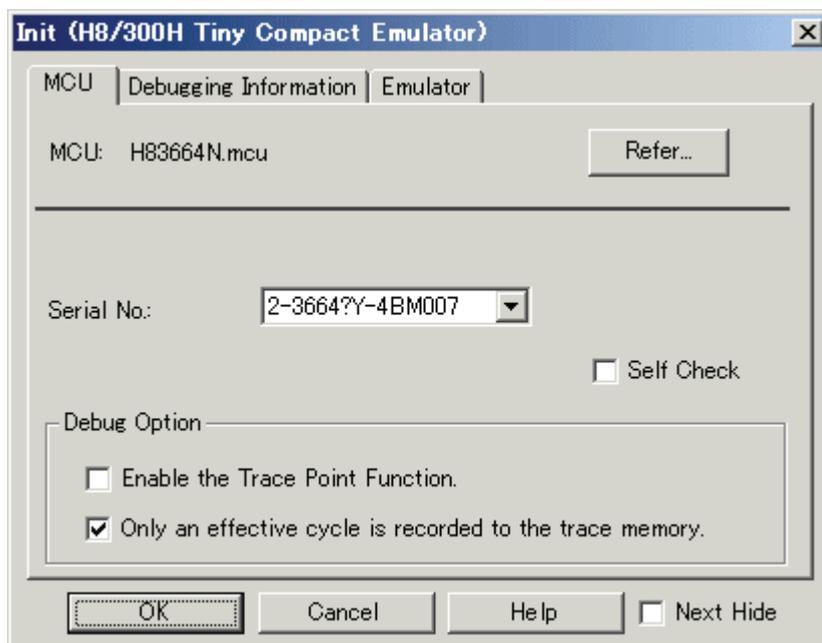
To keep the Init dialog closed next time the debugger is started, check "Next Hide" at the bottom of the Init dialog. You can open the Init dialog using either one of the following methods:

- After the debugger gets started, select Menu - [Setup] -> [Emulator] -> [System...].
- Start Debugger while holding down the Ctrl key.

---

## 5.1.1 MCU Tab

The specified content becomes effective when the next being start.



### 5.1.1.1 Specifying the MCU file



Click the "Refer" button.

The File Selection dialog is opened. Specify the corresponding MCU file.

- An MCU file contains the information specific to the target MCU.
- The specified MCU file is displayed in the MCU area of the MCU tab.

### 5.1.1.2 Setting of the Communication Interface

The displayed data varies depending on the specified communication interface.

The available communication interface varies depending on the products.

The following shows the setting for each communication interface.

#### Setting of the USB Interface

USB communication uses the personal computer's USB interface. It is compliant with USB 1.1.

Before USB communication can be performed, the computer must have a dedicated device driver installed in it. For details on how to install USB device drivers, see "3.3.1.1 Install of USB device driver."

The currently USB-connected emulators are listed in the Serial No. area. Select the serial No. of the emulator you want to connect.

Serial No.:

#### 5.1.1.3 Executing Self-Check

Specify this option to execute self-check\* on the emulator when the debugger starts up.

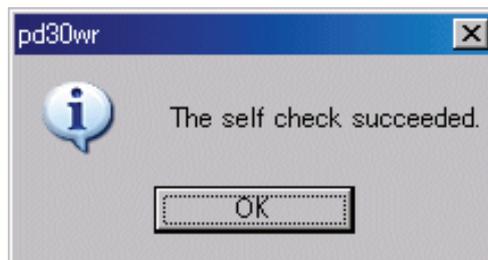
Self Check

Be sure to select the above check box only when you want to perform self-check at startup. Specify this option in the following cases:

- When the firmware cannot be downloaded
- When although the firmware is successfully downloaded, the debugger does not start
- When the MCU goes wild or something is wrong with the trace results and you want to check whether the emulator is operating normally.

Select the check box to close the Init dialog box. After connecting to the emulator and confirming the firmware, the debugger will immediately start self-check on the emulator. (Self-check takes about 30 seconds to 1 minute.)

If an error is found in this self-check, the debugger displays the content of the error and is finished. When the self-check terminated normally, the dialog box shown below is displayed. When you click OK, the debugger starts up directly in that state.



This specification is effective only when the debugger starts up.

\* Self-check refers to the function to check the emulator's internal circuit boards for memory condition, etc. Refer to the user's manual of your emulator for details about the self-check function.

#### 5.1.1.4 Choosing to use or not to use the trace point setting function

Specify whether or not you want to use the trace point setting function. (By default, the trace point function is unused.)

Enable the Trace Point Function.

Select the above check box when you use the event of Compact emulator as a trace point. This specification can only be set or changed when you start Debugger.

#### Supplementary explanation

When the trace point setting function is enabled, the following limitations apply:

- Hardware break function cannot use.

---

### 5.1.1.5 Recording only the valid cycles in trace memory

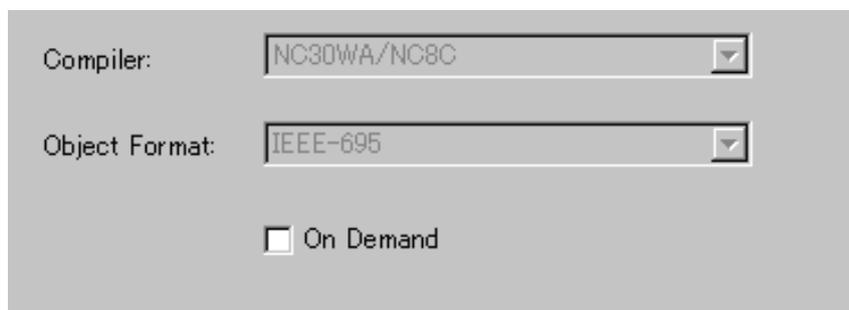
The compact emulator permits you to choose to record only the valid cycles or record all cycles in trace memory.

Only an effective cycle is recorded to the trace memory.

To record only the valid cycles, select this check box.

### 5.1.2 Debugging Information Tab

The specified content becomes effective when the next being download.



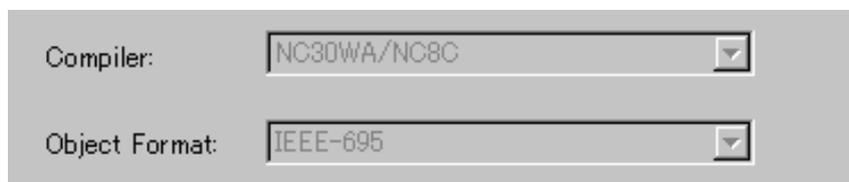
Compiler: NC30WA/NC80

Object Format: IEEE-695

On Demand

#### 5.1.2.1 display the compiler used and its object format

Display the compiler used and its object file format.



Compiler: NC30WA/NC80

Object Format: IEEE-695

Please specify the compiler used and its object file format in the dialog opened by menu [Debug] -> [Debug Settings...].

#### 5.1.2.2 Specify the Storing of Debugging Information

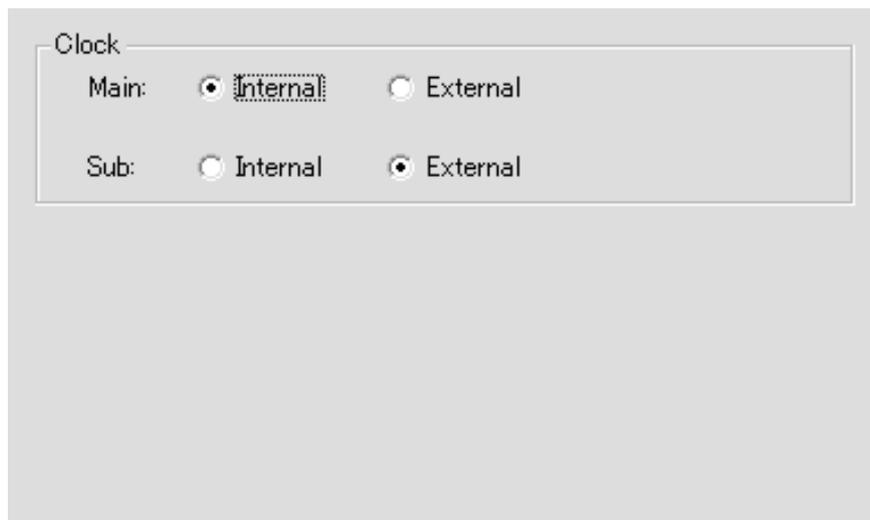
There are two methods for storing debugging information: on-memory and on-demand. Select one of these two methods. (The on-memory method is selected by default.)

To select the on-demand method, click the On Demand check box.

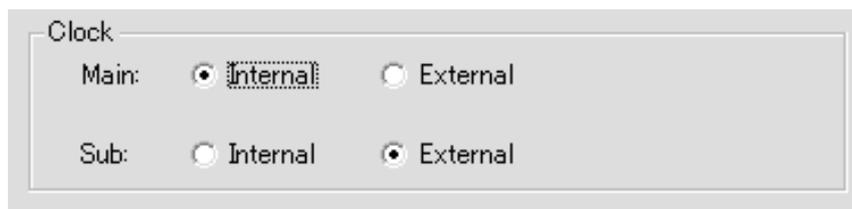
- On-memory method  
Debugging information is stored in the internal memory of your computer.  
This method is suitable when the load module (target program) size is small.
- On-demand method  
Debugging information is stored in a reusable temporary file on the hard disk of your computer.  
Because the stored debugging information is reused, the next time you download the same load module it can be downloaded at high speed.  
This method is suitable when the load module (target program) size is large.

**Notes**

- If the load module size is large, the on-memory method may be inefficient because it requires a very large amount of time for downloading. In such a case, select the on-demand method.
- In the on-demand method, a folder in which to store a reusable temporary file is created in the folder that contains the downloaded load module. This folder is named after the load module name by the word "~INDEX\_" to it. If the load module name is "sample.abs", for example, the folder name is "~INDEX\_sample". This folder is not deleted even after quitting the debugger.

**5.1.3 Emulator Tab****5.1.3.1 Specify the Target Clock**

Change the setting by synchronizing with the clock used by the target microcomputer. (Internal is set by default.)



Select Internal to set the internal clock, and External to set the external clock. The specified content becomes effective when the next being start.

---

## 5.2 MCU Setting Dialog (H8/300H Tiny Debugger)

In the MCU Setting dialog box, setting information on the user target. The MCU Setting dialog box opens after closing the Init dialog box.

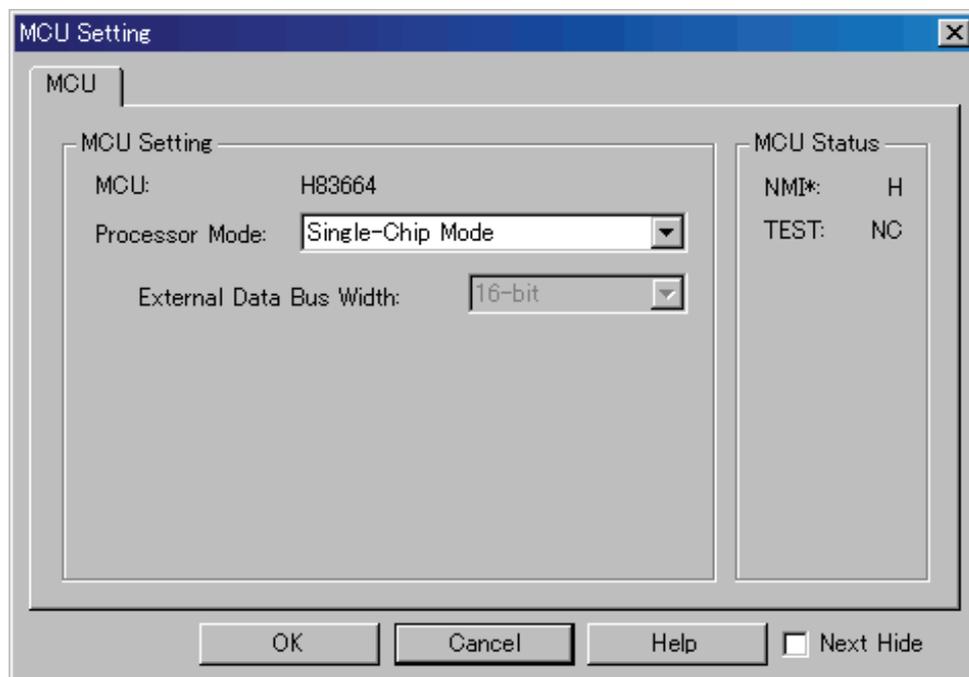
To keep the MCU Setting dialog box closed next time the debugger is started, check "Next Hide" at the bottom of the MCU Setting dialog box.

You can open the MCU Setting dialog using either one of the following methods:

- After the debugger gets started, select Menu - [Setup] -> [Emulator] -> [Target...].

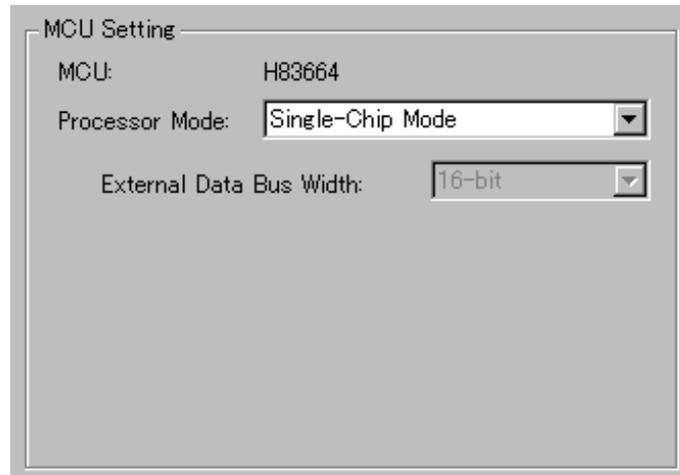
### 5.2.1 MCU Tab

The specified content becomes effective when the next being start.



### 5.2.1.1 Select the Processor Mode

Specify the processor mode for the target system.



Either the following can be specified.

- Single-chip Mode  
Single-chip Mode

### 5.2.1.2 Inspecting the MCU status

Clicking this tab displays the status of each MCU pin. It allows to check whether the MCU pin status matches the processor mode to be set.



"NC" means that the value is indeterminate.

---

[MEMO]

# Tutorial

(blank)

## 6. Tutorial

### 6.1 Introduction

This section describes the main functions of this debugger by using a tutorial program.

The tutorial program is based on the C program that sorts ten random data items in ascending or descending order.

The tutorial program performs the following actions:

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

#### Note

After recompilation, the addresses may differ from those given in this section.

---

## 6.2 Usage

Please follow these instructions:

### 6.2.1 Step1 : Starting the Debugger

#### 6.2.1.1 Preparation before Use

To run the High-performance Embedded Workshop and connect the emulator, refer to “4 Preparation before Use”.

#### 6.2.1.2 Setup the Debugger

If it connects with an emulator, the dialog box for setting up a debugger will be displayed. Please set up the debugger in this dialog box.

To setup the debugger in this dialog box, refer to “5 Setup the Debugger”.

After the setup of a debugger, it will function as a debugger.

## 6.2.2 Step2 : Checking the Operation of RAM

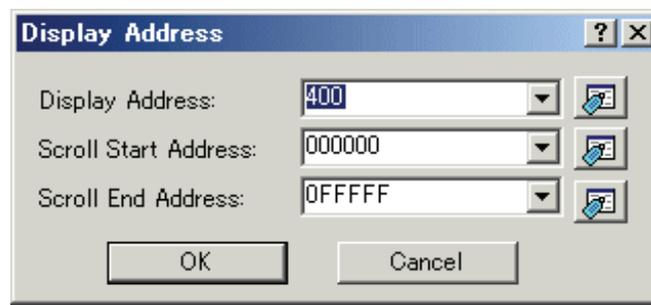
Check that RAM is operating correctly. Display and edit the contents of the memory in the [Memory] window to check that the memory is operating correctly.

### Note

The memory can be installed on the board in some microcomputers. In this case, however, the above way of checking the operation of memory may be inadequate. It is recommended that a program for checking the memory be created.

### 6.2.2.1 Checking the Operation of RAM

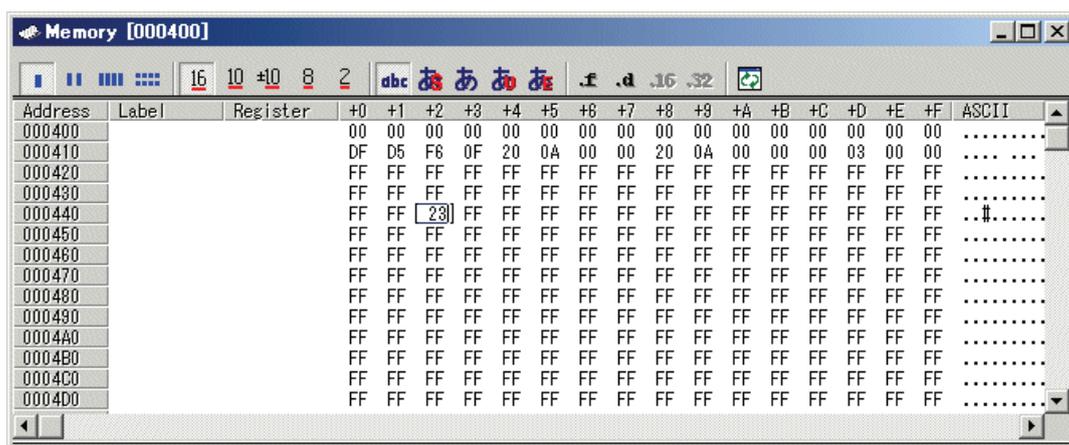
Select [Memory] from the [CPU] submenu of the [View] menu and enter the RAM address (Here, enter H'400) in the [Display Address] edit boxes. The [Scroll Start Address] and [Scroll End Address] editing box is left to a default setting. (By default, the scroll range is set to 0h to the maximum address of MCU.)



### Note

The settings of the RAM area differ depending on the product. For details, refer to the hardware manual.

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



Placing the mouse cursor on a point in the display of data in the [Memory] window and double-clicking allows the values at that point to be changed.

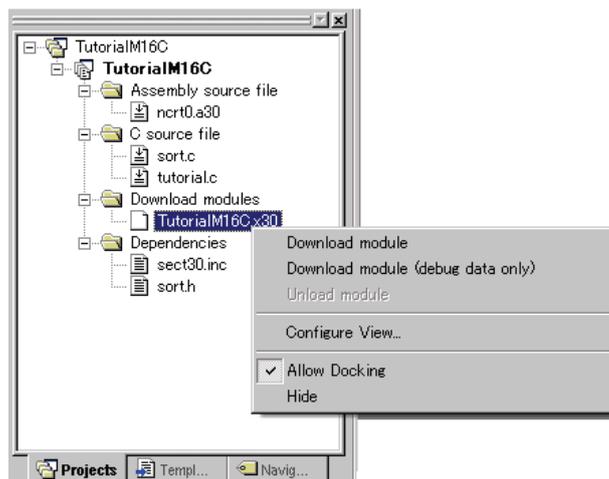
---

## 6.2.3 Step3 : Downloading the Tutorial Program

### 6.2.3.1 Downloading the Tutorial Program

Download the object program to be debugged.

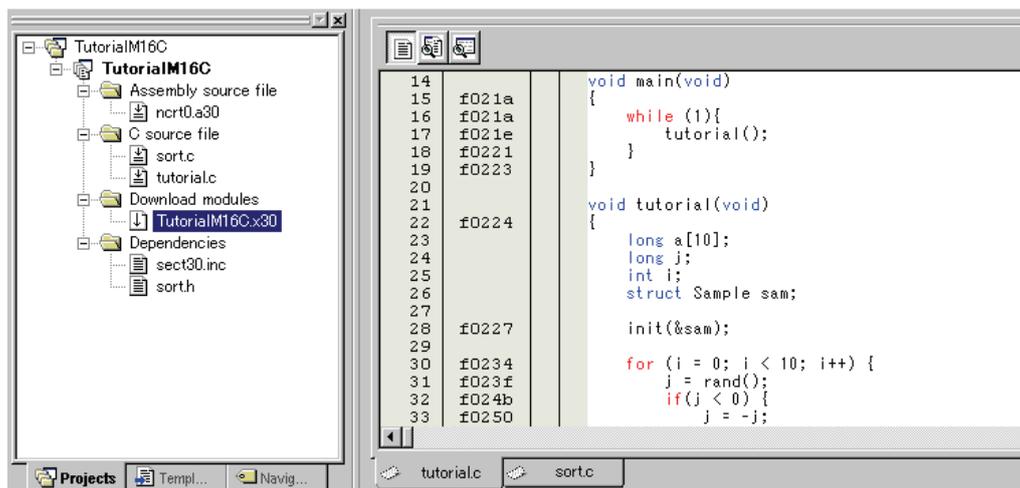
- The Debugger for H8/300H Tiny  
Select [Download module] from [Tutorial.abs] under [Download modules].



### 6.2.3.2 Displaying the Source Program

This debugger allows the user to debug a user program at the source level.

Double-click [tutorial.c] under [C source file]. A [Editor(Source)] window opens and the contents of a "Tutorial.c" file are displayed.



Select the [Format Views...] option from the [Setup] menu to set a font and size that are legible, if necessary.

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

## 6.2.4 Step4 : Setting a Breakpoint

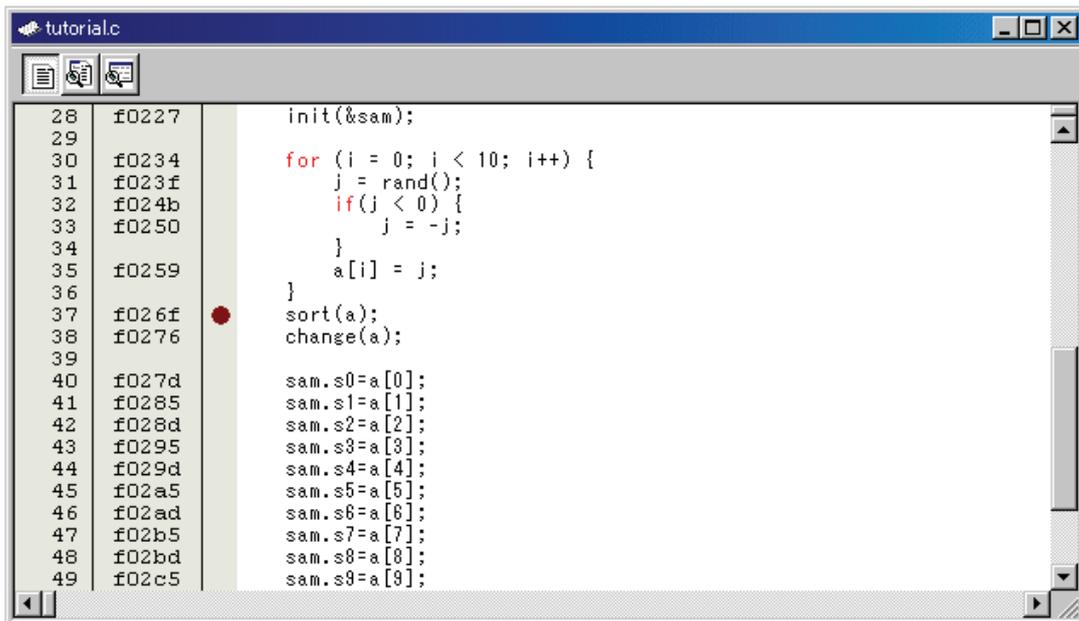
A software breakpoint is a basic debugging function.

The [Editor(Source)] window provides a very simple way of setting a software breakpoint at any point in a program.

### 6.2.4.1 Setting a Software Breakpoint

For example, to set a software breakpoint at the sort function call:

Double-click the [S/W breakpoints] column on the line containing the sort function call.



The screenshot shows a code editor window titled "tutorial.c". The editor displays a C program with the following code:

```
28 f0227      init(&sam);
29
30 f0234      for (i = 0; i < 10; i++) {
31 f023f          j = rand();
32 f024b          if(j < 0) {
33 f0250              j = -j;
34
35 f0259          }
36              a[i] = j;
37 f026f      }
38 f0276      sort(a);
39      change(a);
40
41 f027d      sam.s0=a[0];
42 f0285      sam.s1=a[1];
43 f028d      sam.s2=a[2];
44 f0295      sam.s3=a[3];
45 f029d      sam.s4=a[4];
46 f02a5      sam.s5=a[5];
47 f02ad      sam.s6=a[6];
48 f02b5      sam.s7=a[7];
49 f02bd      sam.s8=a[8];
          sam.s9=a[9];
```

A red dot is visible in the left margin of the editor, positioned to the left of line 37, indicating that a software breakpoint has been set on the `sort(a);` line.

The red symbol will appear on the line containing the sort function call. This shows that a softwarebreak breakpoint has been set.

---

## 6.2.5 Step5 : Executing the Program

Execute the program as described in the following:

### 6.2.5.1 Resetting of CPU

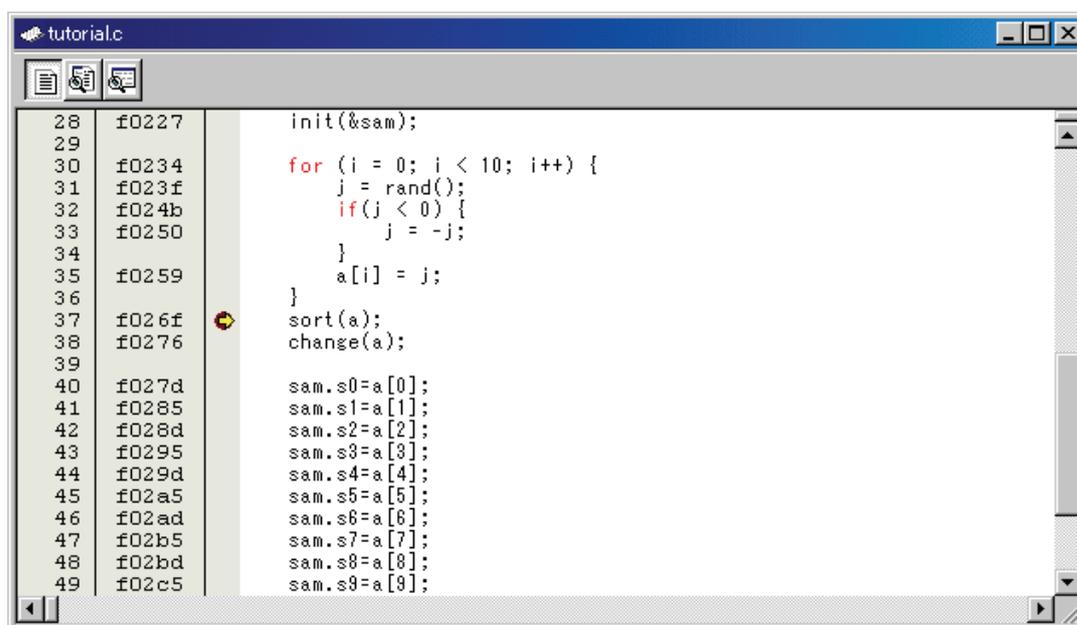
By default, CPU is not reset after downloading a program.

To reset the CPU, select [Reset CPU] from the [Debug] menu, or click the [Reset CPU] button  on the toolbar.

### 6.2.5.2 Executing the Program

To execute the program, select [Go] from the [Debug] menu, or click the [Go] button  on the toolbar.

The program will be executed up to the breakpoint that has been set, and an arrow will be displayed in the [S/W Breakpoints] column to show the position that the program has halted.



The screenshot shows a debugger window titled 'tutorial.c'. The window contains a list of memory addresses and corresponding C code. A yellow arrow points to line 37, indicating a breakpoint. The code is as follows:

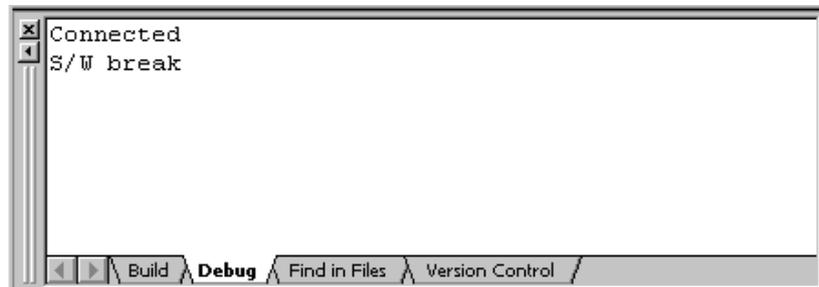
```
28 f0227      init(&sam);
29
30 f0234      for (i = 0; i < 10; i++) {
31 f023f          j = rand();
32 f024b          if(j < 0) {
33 f0250              j = -j;
34
35 f0259              a[i] = j;
36
37 f026f          }
38 f0276      sort(a);
39
40 f027d      sam.s0=a[0];
41 f0285      sam.s1=a[1];
42 f028d      sam.s2=a[2];
43 f0295      sam.s3=a[3];
44 f029d      sam.s4=a[4];
45 f02a5      sam.s5=a[5];
46 f02ad      sam.s6=a[6];
47 f02b5      sam.s7=a[7];
48 f02bd      sam.s8=a[8];
49 f02c5      sam.s9=a[9];
```

#### Note

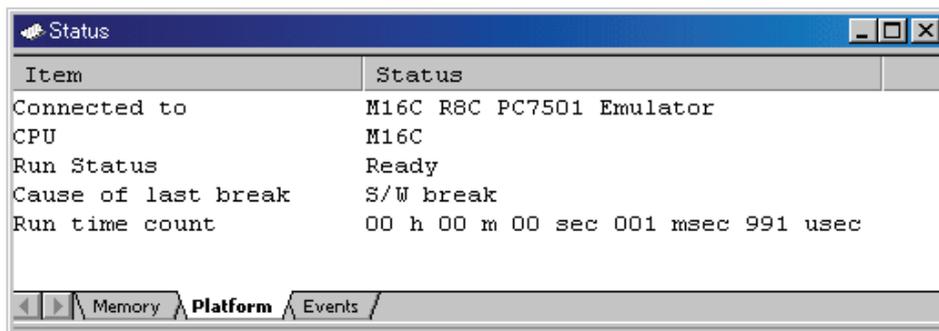
- When the source file is displayed after a break, a path of the source file may be inquired. In this case, please specify the location of a source file.

### 6.2.5.3 Reviewing Cause of the Break

The break factor is displayed in the [Output] window.



The user can also see the cause of the break that occurred last time in the [Status] window. Select [Status] from the [CPU] submenu of the [View] menu. After the [Status] window is displayed, open the [Platform] sheet, and check the Status of Cause of last break.



Please refer to "Display the Cause of the Program Stoppage" about the notation of a break factor.

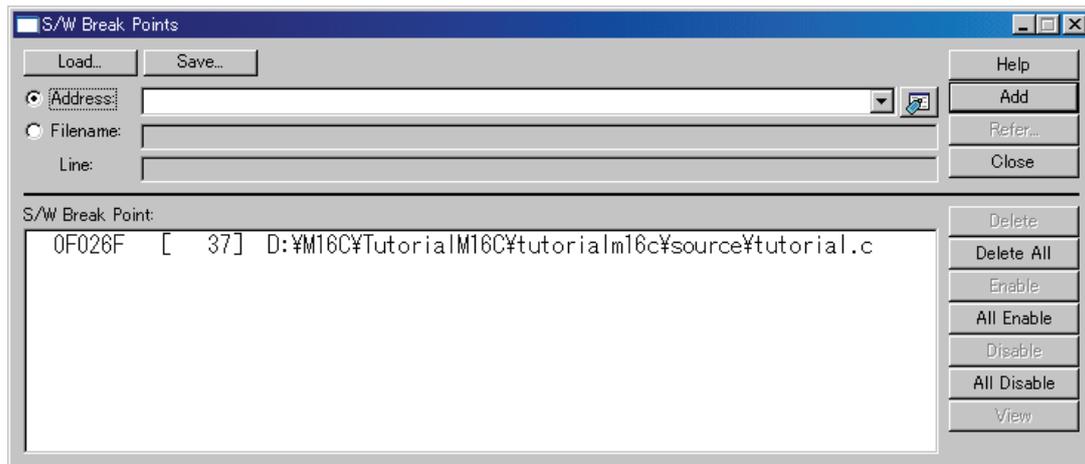
---

## 6.2.6 Step6 : Reviewing Breakpoints

The user can see all the breakpoints set in the program in the [S/W Break Points] window.

### 6.2.6.1 Reviewing Breakpoints

Select [S/W Break Points] from the [Break] submenu of the [View] menu. The [S/W Break Points] window is displayed.



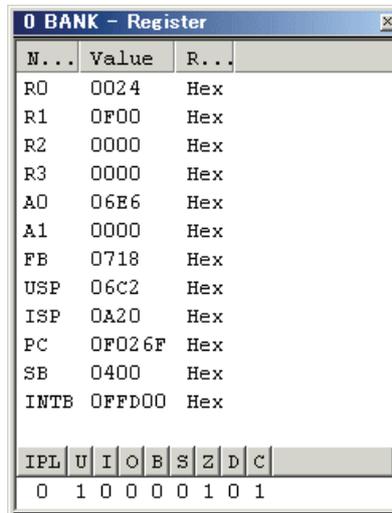
This window allows the user to set or change breakpoints, define new breakpoints, and delete, enable, or disable breakpoints.

## 6.2.7 Step7 : Viewing Register

The user can see all registers/flags value in the [Register] window.

### 6.2.7.1 Viewing Register

Select [Registers] from the [CPU] submenu of the [View] menu. The [Register] window is displayed.



N...	Value	R...
R0	0024	Hex
R1	0F00	Hex
R2	0000	Hex
R3	0000	Hex
A0	06E6	Hex
A1	0000	Hex
FB	0718	Hex
USP	06C2	Hex
ISP	0A20	Hex
PC	0F026F	Hex
SB	0400	Hex
INTB	0FFD00	Hex

IPL	U	I	O	B	S	Z	D	C
0	1	0	0	0	0	1	0	1

### 6.2.7.2 Setting the Register Value

You can change a register/flag value from this window.

Double-click the register line to be changed. The dialog is opened. Enter the value to be changed.



PC - Set Value

Value : 0F026F

Radix : Hex

Set As: Whole Register

OK Cancel

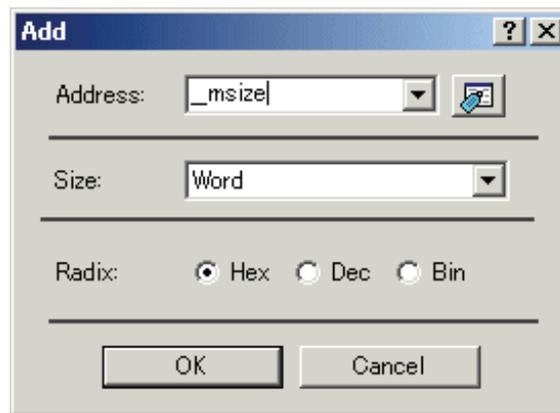
---

## 6.2.8 Step8 : Viewing Memory

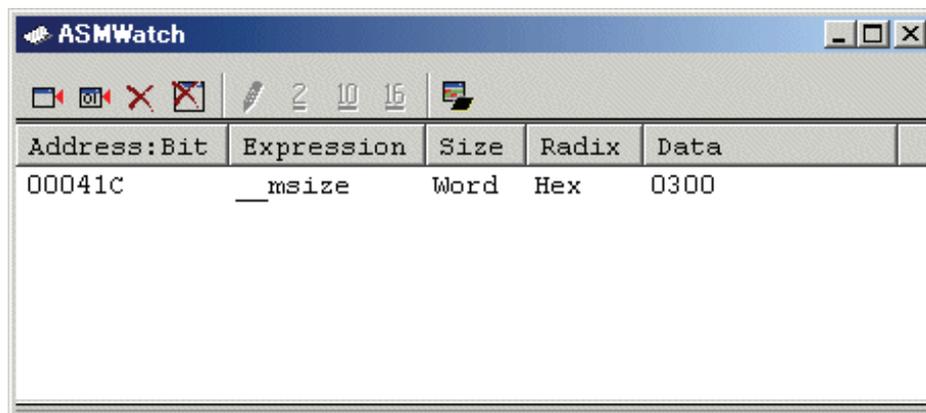
When the label name is specified, the user can view the memory contents that the label has been registered in the [ASM Watch] window.

### 6.2.8.1 Viewing Memory

For example, to view the memory contents corresponding to `__msize` in word size:  
Select [ASM Watch] from the [Symbol] submenu of the [View] menu, open the [ASM Watch] window.  
And click the [ASM Watch] window with the right-hand mouse button and select [Add...] from the popup menu, enter `__msize` in the [Address] edit box, and set Word in the [Size] combo box.



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



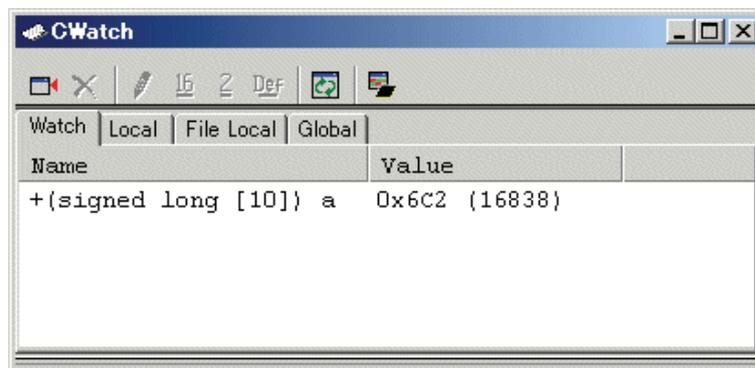
## 6.2.9 Step9 : Watching Variables

As the user steps through a program, it is possible to watch that the values of variables used in the user program are changed.

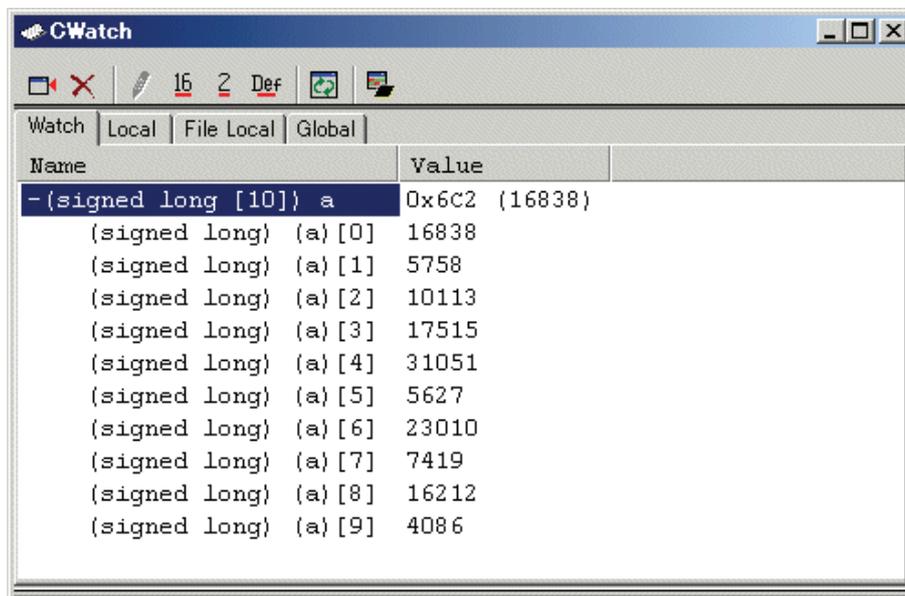
### 6.2.9.1 Watching Variables

For example, set a watch on the long-type array a declared at the beginning of the program, by using the following procedure:

Click the left of displayed array a in the [Editor(Source)] window to position the cursor, and select [Add C Watch...] with the right-hand mouse button. The [Watch] tab of [C watch] window in which the variable is displayed opens.



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



---

### 6.2.9.2 Registering Variable

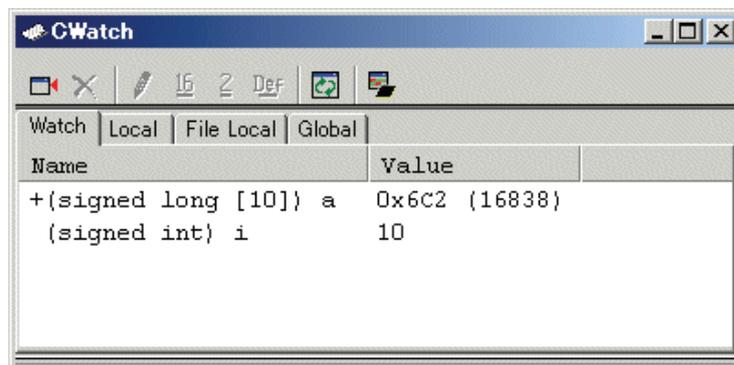
The user can also add a variable to the [C Watch] window by specifying its name.

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

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



Click the [OK] button. The [C Watch] window will now also show the int-type variable i.



## 6.2.10 Step10 : Stepping Through a Program

This debugger provides a range of step menu commands that allow efficient program debugging.

1. **Step In**  
Executes each statement, including statements within functions.
2. **Step Out**  
Steps out of a function, and stops at the statement following the statement in the program that called the function.
3. **Step Over**  
Executes a function call in a single step.
4. **Step...**  
Steps the specified times repeatedly at a specified rate.

### 6.2.10.1 Executing [Step In] Command

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

To step through the sort function, select [Step In] from the [Debug] menu, or click the [Step In] button



on the toolbar.

The PC cursor moves to the first statement of the sort function in the [Editor(Source)] window.

```

16 f0087      p_sam->s5 = 0;
17 f009f      p_sam->s6 = 0;
18 f00b7      p_sam->s7 = 0;
19 f00cf      p_sam->s8 = 0;
20 f00e7      p_sam->s9 = 0;
21 f00ff      }
22
23          sort(long #a)
24  f0102      ↪ {
25              long t;
26              int i, j, k, gap;
27
28              gap = 5;
29              while( gap > 0 ){
30                  for( k=0; k<gap; k++){
31                      for( i=k+gap; i<10; i=i+gap ){
32                          for( j=i-gap; j>=k; j=j-gap){
33                              if( a[j]>a[j+gap]){
34                                  t = a[j];
35                                  a[j] = a[j+gap];

```

### 6.2.10.2 Executing [Step Out] Command

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

To step out of the sort function, select [Step Out] from the [Debug] menu, or click the [Step Out]



button on the toolbar.

The PC cursor slips out of a sort function, and moves to the position before a change function.

```

31 f023f      j = rand();
32 f024b      if(j < 0) {
33 f0250          j = -j;
34             }
35 f0259      a[i] = j;
36             }
37 f026f      sort(a);
38 f0276      change(a);
39
40 f027d      sam.s0=a[0];
41 f0285      sam.s1=a[1];
42 f028d      sam.s2=a[2];
43 f0295      sam.s3=a[3];
44 f029d      sam.s4=a[4];

```

**Note**

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

**6.2.10.3 Executing [Step Over] Command**

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

To step through all statements in the change function at a single step, select [Step Over] from the

[Debug] menu, or click the [Step Over] button  on the toolbar.

The PC cursor moves to the next position of a change function.

```

30 f0234      for (i = 0; i < 10; i++) {
31 f023f          j = rand();
32 f024b          if(j < 0) {
33 f0250              j = -j;
34             }
35 f0259          a[i] = j;
36             }
37 f026f      sort(a);
38 f0276      change(a);
39
40 f027d      sam.s0=a[0];
41 f0285      sam.s1=a[1];
42 f028d      sam.s2=a[2];
43 f0295      sam.s3=a[3];
44 f029d      sam.s4=a[4];
45 f02a5      sam.s5=a[5];
46 f02ad      sam.s6=a[6];
47 f02b5      sam.s7=a[7];
48 f02bd      sam.s8=a[8];
49 f02c5      sam.s9=a[9];
50             }
51

```

## 6.2.11 Step11 : Forced Breaking of Program Executions

This debugger can force a break in the execution of a program.

### 6.2.11.1 Forced Breaking of Program Executions

Cancel all breaks.

To execute the remaining sections of the main function, select [Go] from the [Debug] menu or the [Go]

button  on the toolbar.

The program goes into an endless loop. To force a break in execution, select [Halt Program] from the

[Debug] menu or the [Halt] button  on the toolbar.

---

## 6.2.12 Step12 : Displaying Local Variables

The user can display local variables in a function using the [C Watch] window.

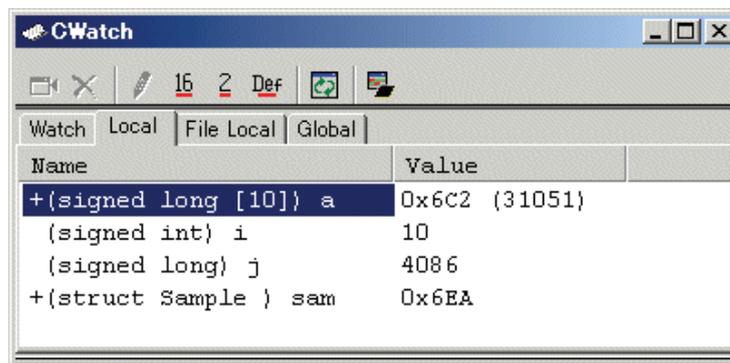
### 6.2.12.1 Displaying Local Variables

For example, we will examine the local variables in the tutorial function, which declares four local variables: a, j, i, and sam.

Select [C Watch] from the [Symbol] submenu of the [View] menu. The [C Watch] window is displayed. By default, [C watch] window has four tabs as following:

- [Watch] tab  
Only the variable which the user registered is displayed.
- [Local] tab  
All the local variables that can be referred to by the scope in which the the PC exists are displayed. If a scope is changed by program execution, the contents of the [Local] tab will also change.
- [File Local] tab  
All the file local variables of the file scope in which the PC exists are displayed. If a file scope is changed by program execution, the contents of the [File Local] tab will also change.
- [Global] tab  
All the global variables currently used by the downloaded program are displayed.

Please choose the [Local] tab, when you display a local variable.



Click mark '+' at the left side of array a in the [Locals] window to display the elements.

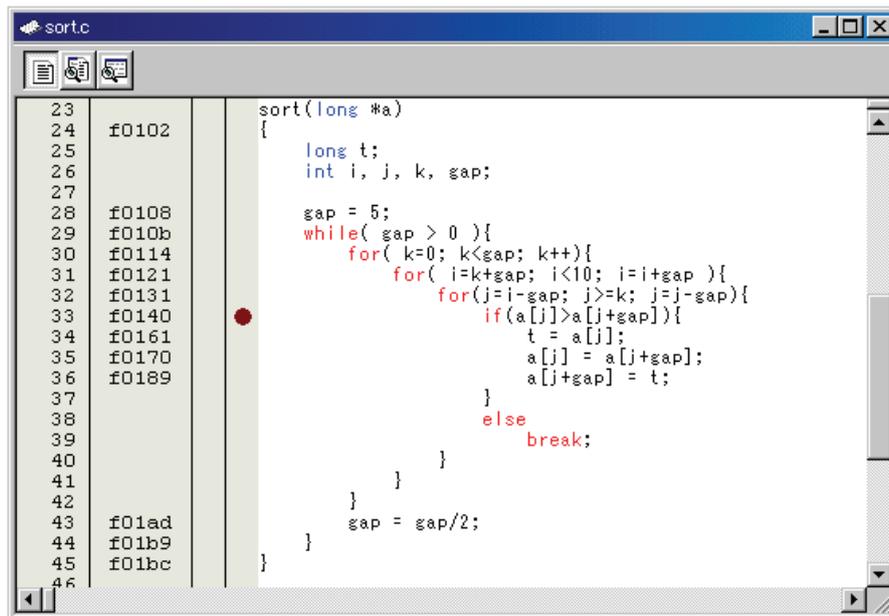
When the user refers to the elements of array a before and after the execution of the sort function, it is clarified that random data is sorted in descending order.

### 6.2.13 Step13 : Stack Trace Function

The debugger uses the information on the stack to display the names of functions in the sequence of calls that led to the function to which the program counter is currently pointing.

#### 6.2.13.1 Reference the function call status

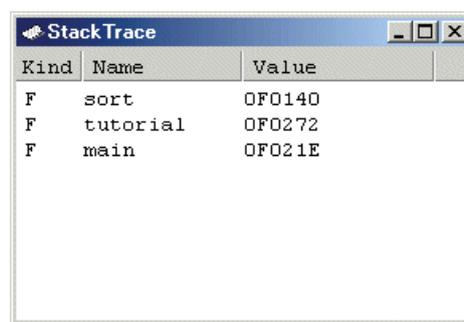
Double-click the [S/W Breakpoints] column in the sort function and set a software breakpoint.



To execute the user program from the reset vector address, select [Reset Go] from the [Debug] menu,

or click the [Reset Go] button  on the toolbar.

After the break in program execution, select [Stack Trace] from the [Code] submenu of the [View] menu to open the [Stack Trace] window.



The upper figure shows that the position of the program counter is currently at the selected line of the sort() function, and that the sort() function is called from the tutorial() function.

---

### 6.2.14 What Next?

This tutorial has described the usage of this debugger.

Sophisticated debugging can be carried out by using the emulation functions that the emulator offers. This provides for effective investigation of hardware and software problems by accurately isolating and identifying the conditions under which such problems arise.

# Reference

(blank)

## 7. Windows/Dialogs

The window of this debugger is shown below.

When the window name is clicked, the reference is displayed.

Window Name	View Menu
RAM Monitor Window	[View]->[CPU]->[RamMonitor]
ASM Watch Window	[View]->[Symbol]->[ASMWatch]
C Watch Window	[View]->[Symbol]->[CWatch]
S/W Break Point Setting Window	[View]->[Break]->[S/W Break Points]
H/W Break Point Setting Window	[View]->[Break]->[H/W Break Points]
Trace Point Setting Window	[View]->[Trace]->[Trace Points]
Trace Window	[View]->[Trace]->[Trace]
GUI I/O Window	[View]->[Graphic]->[GUI I/O]

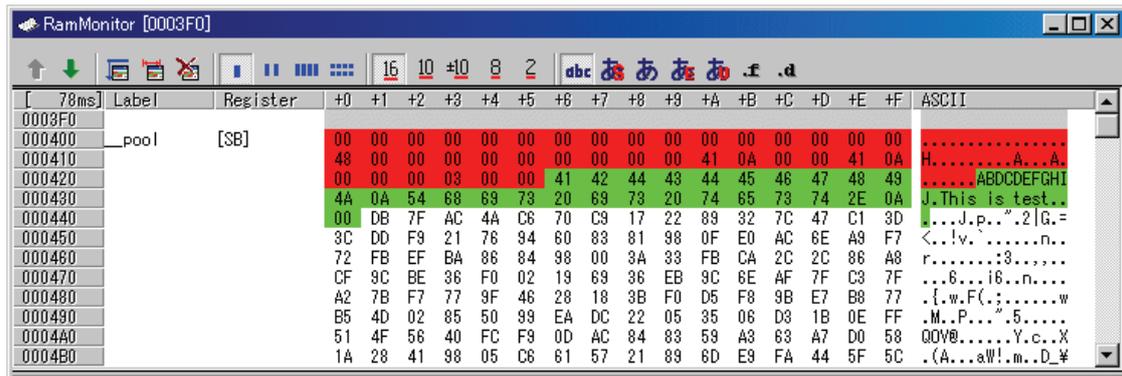
For the reference of the following windows, refer to the help attached to a High-performance Embedded Workshop main part.

- Differences Window
- Map Window
- Command Line Window
- Workspace Window
- Output Window
- Disassembly Window
- Memory Window
- IO Window
- Status Window
- Register Window
- Image Window
- Waveform Window
- Stack Trace Window

## 7.1 RAM Monitor Window

The RAM monitor window is a window in which changes of memory contents are displayed while running the target program.

The relevant memory contents are displayed in dump form in the RAM monitor area by using the realtime RAM monitor function. The displayed contents are updated at given intervals (by default, every 100 ms) while running the target program.



- This system has 1 Kbyte of RAM monitor area which can be located in any contiguous address location or in 4 separate blocks comprised of 256 bytes each.
- The RAM monitor area can be changed to any desired address range.  
Refer to "Setting the RAM monitor area" for details on how to change the RAM monitor area.  
The default RAM monitor area is mapped into a 1-Kbyte area beginning with the start address of the internal RAM.
- The display content updating interval can be set for each window individually.  
The actual updating interval at which the display contents are actually updated while running the target program is shown in the title field of the Address display area.
- The background colors of the data display and code display areas are predetermined by access attribute, as shown below.

Access attribute	Background color
Read accessed address	Green
Write accessed address	Red
Non-accessed address	White

The background colors can be changed.

### ATTENTION

- The RAM monitor window shows the data that have been accessed through the bus. Therefore, changes are not reflected in the displayed data unless they have been accessed via the target program as in the case where memory is rewritten directly from an external I/O.
- If the data in the RAM monitor area are displayed in lengths other than the byte, it is possible that the data will have different memory access attributes in byte units. If bytes in one data have a different access attribute as in this case, those data are enclosed in parentheses when displayed in the window. In that case, the background color shows the access attribute of the first byte of the data.

```

001B  00C8  00D2  0000  007C
0000  0000  0000  0000  0000
0000  (007C) FF8C  0000  0000
0000  0000  0000  0050  0000

```

- The displayed access attributes are initialized by downloading the target program.
- The interval time at which intervals the display is updated may be longer than the specified interval depending on the operating condition (shown below).
  - Host machine performance/load condition
  - Communication interface
  - Window size (memory display range) or the number of windows displayed

### 7.1.1 Extended Menus

This window has the following popup menus that can be brought up by right-clicking in the window.

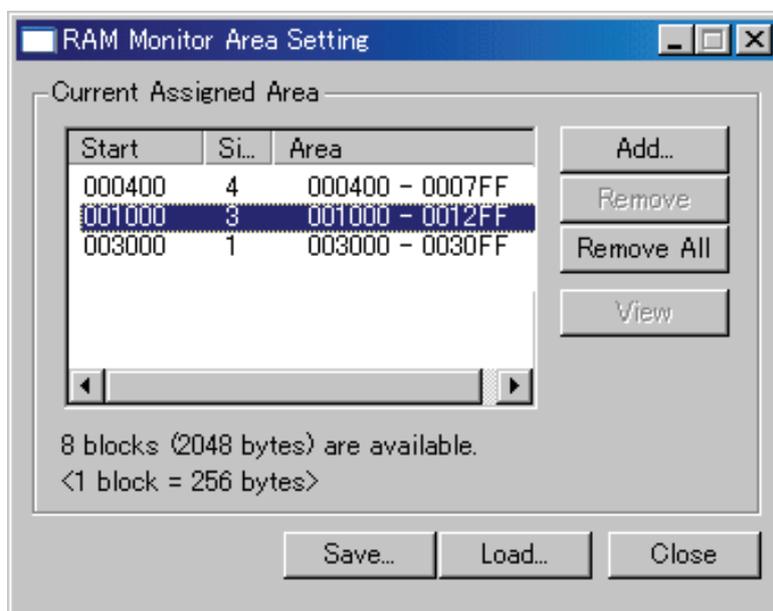
Menu		Function
RAM Monitor Area...		Set RAM monitor base address.
Sampling Period...		Set RAM monitor sampling period.
Clear		Clear access attribute.
Up		Moves display position to the immediately preceding RAM monitor area (smaller address)
Down		Moves display position to the immediately following RAM monitor area (larger address)
Address...		Display from specified address.
Scroll Area...		Specify scroll range.
Data Length	1byte	Display in 1Byte unit.
	2bytes	Display in 2Byte unit.
	4bytes	Display in 4Byte unit.
	8bytes	Display in 8Byte unit.
Radix	Hex	Display in Hexadecimal.
	Dec	Display in Decimal.
	Single Dec	Display in Signed Decimal.
	Oct	Display in Octdecimal.
	Bin	Display in Binary.
Code	ASCII	Display as ASCII character.
	SJIS	Display as SJIS character.
	JIS	Display as JIS character.
	UNICODE	Display as UNICODE character.
	EUC	Display as EUC character.
	Float	Display as Floating-point.
	Double	Display as Double Floating-point.
Layout	Label	Switch display or non-display of Label area.
	Register	Switch display or non-display of Register area.
	Code	Switch display or non-display of Code area.
Column...		Set the number of columns displayed on one line.
Split		Split window.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

---

## 7.1.2 Setting the RAM monitor area

Choose the popup menu [RAM Monitor Area...] in the RAM monitor window.

The RAM monitor area setup window shown below will appear. The currently set RAM monitor areas are listed in this window.



Use this window to add, delete or change RAM monitor areas.

- Specify a RAM monitor area by its start address and size (the latter by a number of blocks.)
- The start address can be specified in 0x100 byte units.  
If you specify a non-aligned address value, it is rounded off to the nearest address value in 0x100 byte units before being set.
- Specify the size of the RAM monitor area by a number of blocks.  
For the Compact Emulator, one block is 256 bytes in size. Up to 4 blocks can be specified.
- RAM monitor areas can be added until the total number of blocks used reaches 4.  
(The number of blocks (and the size) that are currently available to use are displayed below the list.)

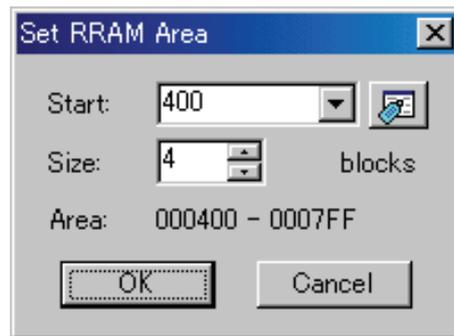
### 7.1.2.1 Changing the RAM Monitor Area

The start address and the size of the RAM monitor area can be changed.

- Changing from a dialog box

Select the RAM monitor area you want to change from a list of RAM monitor areas and double-click on it.

The Set RRAM Area dialog box shown below will appear. Specify the start address and the size (by a number of blocks) of the RAM monitor area in the Start and the Size fields of this dialog box.



- Changing directly in the window

Select the RAM monitor area you want to change from a list of RAM monitor areas and click again in its Start display column or Size display column.

Specify a new start address or a new size with which you want to be changed in the ensuing edit box. Press the Enter key to confirm what you've entered, or the Esc key to cancel.

Start	Size	Area
000400	4	000400 - 0007FF
001000	3	001000 - 0013FF

Changing the address

Start	Size	Area
000400	4	000400 - 0007FF
001000	3	001000 - 0013FF

Changing the size

### 7.1.2.2 Adding RAM Monitor Areas

Click the [Add...] button.

The Set RRAM Area dialog box will appear. Specify the start address and the size (by a number of blocks) of a new RAM monitor area in the Start and the Size fields of this dialog box.

### 7.1.2.3 Deleting RAM Monitor Areas

Select the RAM monitor area you want to delete from a list of RAM monitor areas and click the [Remove] button.

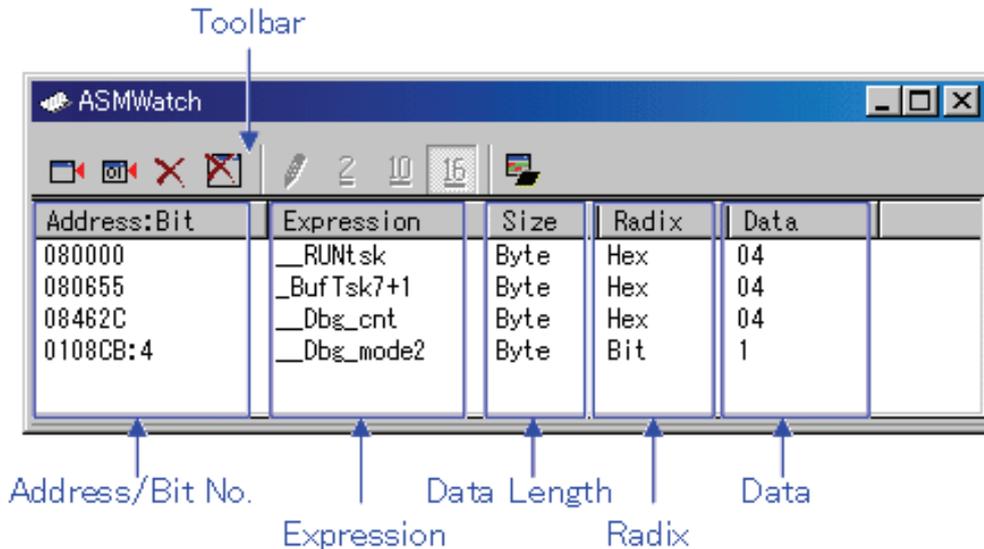
To delete all RAM monitor areas, click the [Remove All] button.

---

## 7.2 ASM Watch Window

The ASM watch window is a window in which you can register specific addresses as watchpoints and inspect memory contents at those addresses.

If a registered address resides within the RAM monitor area, the memory content at that address is updated at given intervals (by default, every 100 ms) during program execution.



- The addresses to be registered are called the "watchpoints." One of the following can be registered:
  - Address (can be specified using a symbol)
  - Address + Bit number
  - Bit symbol
- The registered watchpoints are saved in the debugger when the ASM watch window is closed and are automatically registered when the window is reopened.
- If symbols or bit symbols are specified for the watchpoints, the watchpoint addresses are recalculated when downloading the target program.
- The invalid watchpoints are marked by "-<not active>-" when displayed on the screen.
- The order in which the watchpoints are listed can be changed by a drag-and-drop operation.
- The watchpoint expressions, sizes, radices and datas can be changed by in-place editing.

### ATTENTION

- The RAM monitor obtains the data accessed through the bus. Any change other than the access from the target program will not be reflected.
- If the display data length of the RAM monitor area is not 1 byte, the data's access attribute to the memory may varies in units of 1 byte. In such a case that the access attribute is not unified within a set of data, the data's access attribute cannot be displayed correctly. In this case, the background colors the access attribute color of the first byte of the data.

### 7.2.1 Extended Menus

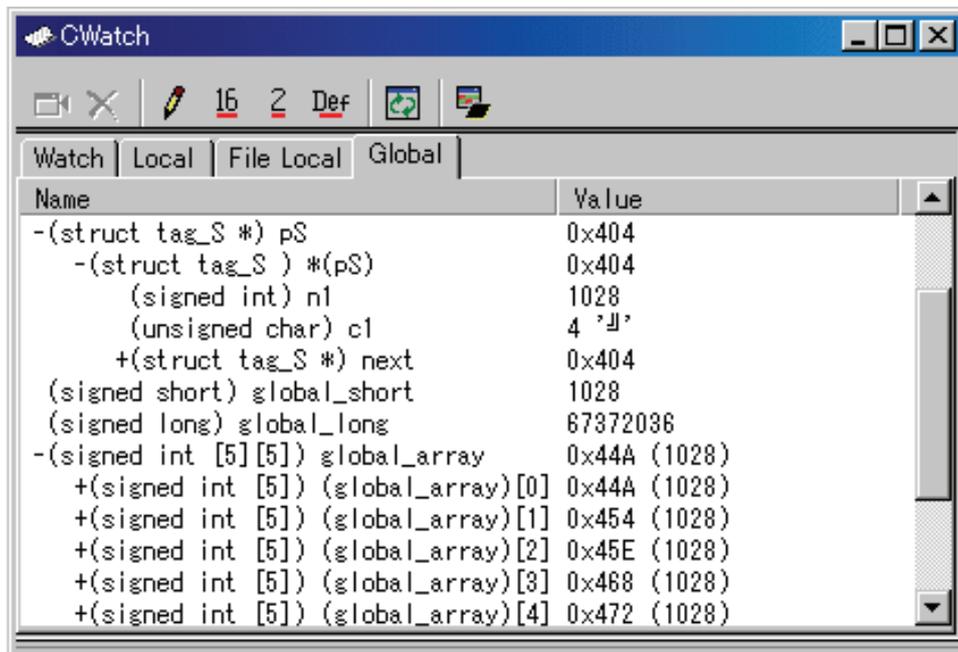
This window has the following popup menus that can be brought up by right-clicking in the window.

Menu		Function
Add...		Add watchpoint.
Add Bit...		Add bit-level watchpoint.
Remove		Remove the selected watchpoint.
Remove All		Remove all watchpoints.
Set...		Set new data to selected watchpoint.
Radix	Bin	Display in Binary.
	Dec	Display in Decimal.
	Hex	Display in Hexadecimal.
Refresh		Refresh memory data.
Layout	Address Area	Switch display or non-display of Address area.
	Size Area	Switch display or non-display of Size area.
RAM Monitor	Enable RAM Monitor	Switch enable or disable RAM monitor function.
	Sampling Period...	Set RAM monitor sampling period.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

---

## 7.3 C Watch Window

The C Watch Window displays C/C++ expressions and their values (results of calculations). The C/C++ expressions displayed in the C Watch Window are known as C watchpoints. The displays of the results of calculating the C watchpoints are updated each time a command is executed. When RAM monitor function is effective and the C watch points are within the RAM monitor area, the displayed values are updated during execution of the target program.



- Variables can be inspected by scope (local, file local or global).
- The display is automatically updated at the same time the PC value changes.
- Variable values can be changed.
- The display radix can be changed for each variable individually.
- Any variable can be registered to the Watch tab, so that it will be displayed at all times:
  - The registered content is saved for each project separately.
  - If two or more of the C watch window are opened at the same time, the registered
- The C watchpoints can be registered to separate destinations by adding Watch tabs.
- Variables can be registered from another window or editor by a drag-and-drop operation.
- The C watchpoints can be sorted by name or by address.
- Values can be inspected in real time during program execution by using the RAM monitor function.

**ATTENTION**

- You cannot change the values of the C watch points listed below:
  - Bit field variables
  - Register variables
  - C watch point which does not indicate an address(invalid C watch point)
- If a C/C++ language expression cannot be calculated correctly (for example, when a C/C++ symbol has not been defined), it is registered as invalid C watch point. It is displayed as "--<not active>--". If that C/C++ language expression can be calculated correctly at the second time, it becomes an effective C watch point.
- The display settings of the Local, File Local and Global tabs are not saved. The contents of the Watch tab and those of newly added tabs are saved.
- The RAM monitor obtains the data accessed through the bus. Any change other than the access from the target program will not be reflected.
- The variables, which are changed in real-time, are global variables and file local variables only.
- If the display data length of the RAM monitor area is not 1 byte, the data's access attribute to the memory may varies in units of 1 byte. In such a case that the access attribute is not unified within a set of data, the data's access attribute cannot be displayed correctly. In this case, the background colors the access attribute color of the first byte of the data.

**7.3.1 Extended Menus**

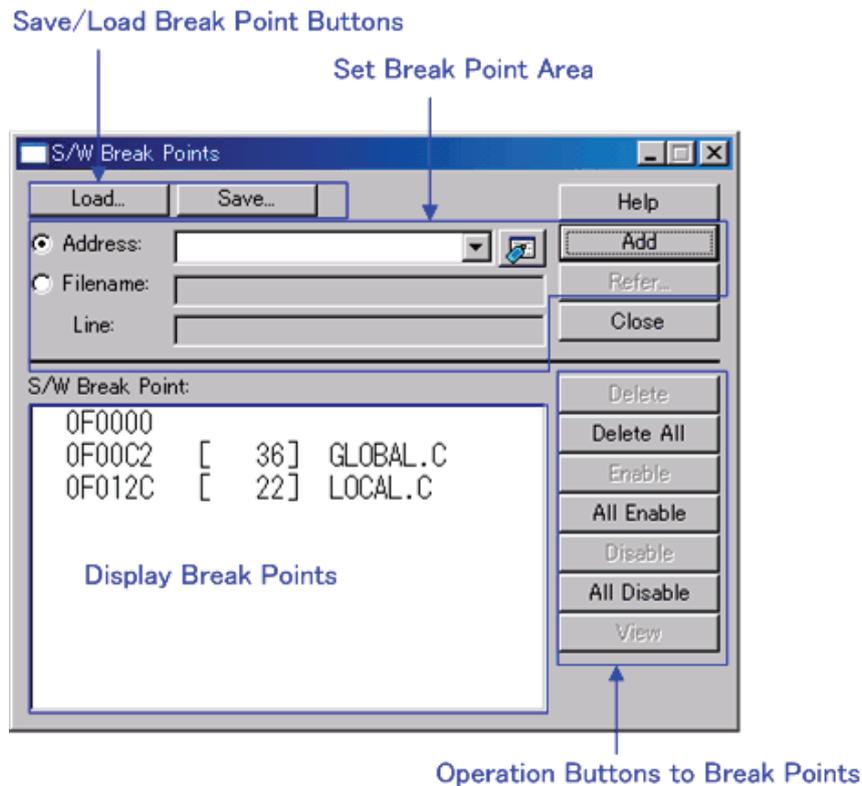
This window has the following popup menus that can be brought up by right-clicking in the window.

<b>Menu</b>		<b>Function</b>
Add...		Add C watchpoint.
Remove		Remove the selected C watchpoint.
Initialize		Reevaluates the selected C watchpoint.
Set New Value...		Set new data to selected C watchpoint.
Radix	Hex	Display in Hexadecimal.
	Bin	Display in Binary.
	Default	Display in Default Radix.
	Toggle(All Variables)	Change radix (toggle).
Refresh		Refresh memory data.
Hide type name		Hide type names from variables.
Show char* as string		Selects whether to display char* type as a string.
Sort	Sort by Name	Sort variables by its name.
	Sort by Address	Sort variables by its address.
RAM Monitor	Enable RAM Monitor	Switch enable or disable RAM monitor function.
	Sampling Period...	Set RAM monitor sampling period.
Add New Tab...		Add new tab.
Remove Tab		Remove the selected tab.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

---

## 7.4 S/W Break Point Setting Window

The S/W Break Point Setting window allows you to set software break points. Software breaks stop the execution of instructions immediately before the specified break point.



- If you have set multiple software breakpoints, program execution stops when any one software break address is encountered (OR conditions).
- You can continue to set software breakpoints until you click the "Close" button to close the S/W Break Point Setting Window.
- You can clear, enable or disable software breakpoints selected by clicking in the software breakpoint display area. You can also enable and disable software breakpoints by double-clicking on them.
- Click on the "Save" button to save the software break points in the file. To reload software break point settings from the saved file, click the "Load" button. If you load software break points from a file, they are added to any existing break points.

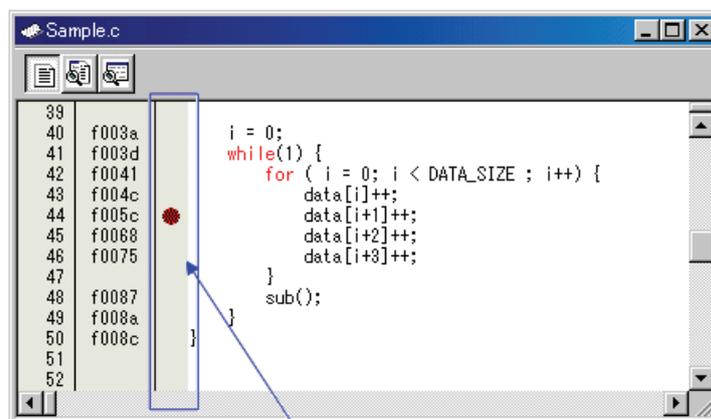
### 7.4.1 Command Button

The buttons on this window has the following meanings.

Button	Function
Load...	Load setting information from a file in which it was saved.
Save...	Save the contents set in the window to a file.
Help	Display the help of this window.
Add	Add the break point.
Refer...	Open file selection dialog box.
Close	Close the window.
Delete	Remove the selected break point.
Delete All	Remove all break points.
Enable	Enable the selected break points.
All Enable	Enable all break points.
Disable	Disable the selected break point.
All Disable	Disable all break points.
View	Shows the selected breakpoint positions in the Editor(Source) window.

### 7.4.2 Setting and Deleting a Break Points from Editor(Source) Window

You can set break points in the Editor(Source) Window. To do so, double-click the break point setting area ("S/W breakpoints" column) for the line in which you want to set the break. (A red marker is displayed on the line to which the break point was set.)

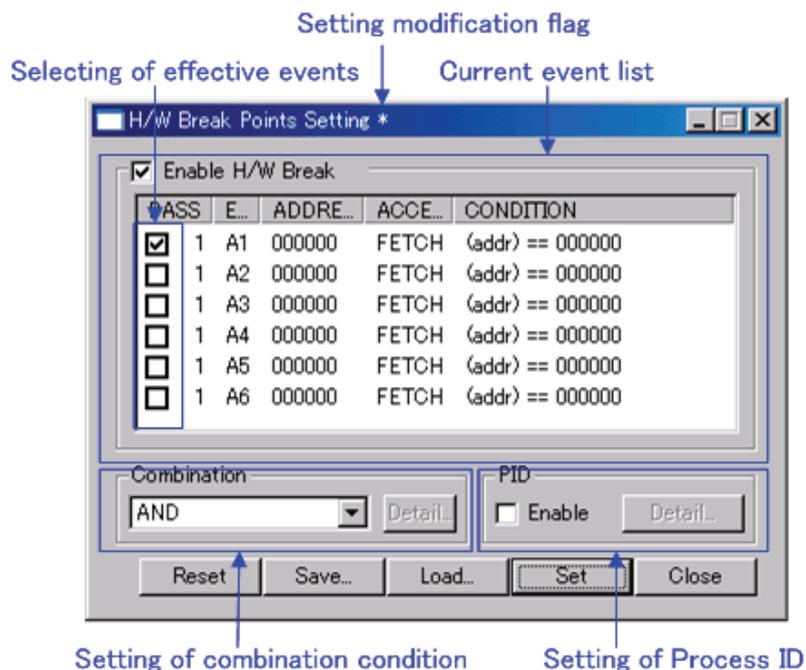


You can delete the break point by double-clicking again in the break point setting area ("S/W breakpoints" column).

In the Editor(Source) window, a display of "S/W breakpoints" column is set to "Enable" by default. To erase this column, deselect the [S/W breakpoints] check box in the dialog box opened by choosing the main menu - [Edit] -> [Define Column Format]. The "S/W breakpoints" column is erased from all Editor (Source) windows. And select popup menu - [Columns] -> [S/W breakpoints] in the Editor (Source) window, A column can be set up for each Editor (Source) windows.

## 7.5 H/W Break Point Setting Window

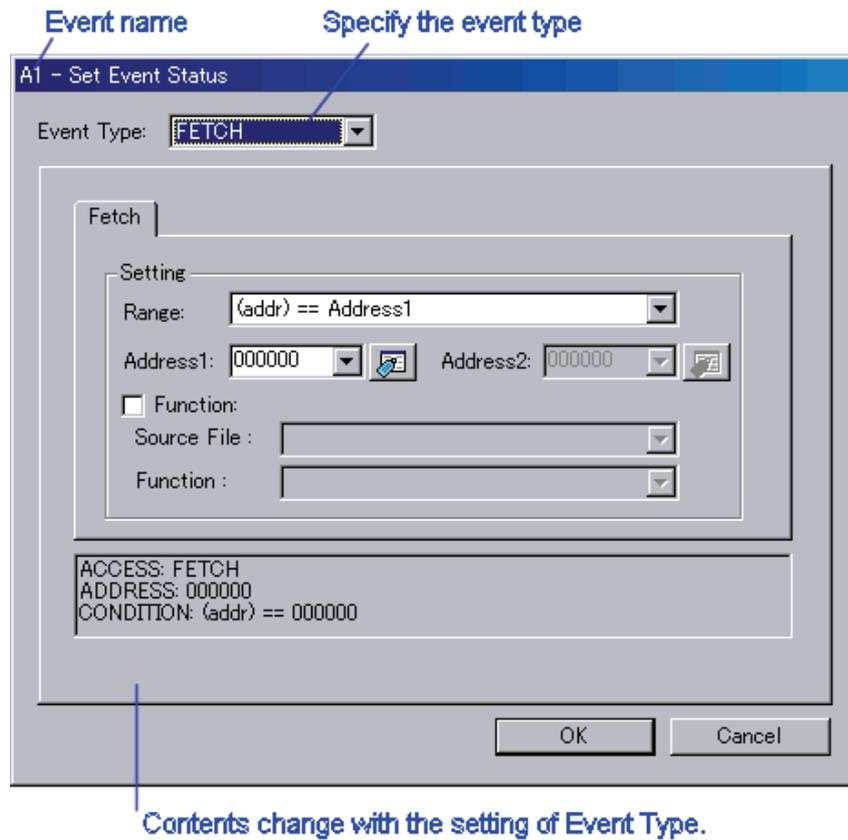
The H/W Breakpoint Setting window is used to set hardware breakpoints for the Emulators.



- The events listed below can be specified as break events. If the contents of events are altered, they are marked by an asterisk (\*) on the title bar. The asterisks (\*) are not displayed after setting up the emulator.
  - Fetch, Memory Access, Bit Access
  - Events at up to two points can be used.
- The H/W Break Point Setting window and the Trace Point Setting windows use the same resource of the emulator. Use the MCU tab in the Init dialog box, in order to specify for which function the resources are used. On this tab, deselect the Enable the Trace Point Function check box.
- These events can be combined in one of the following ways:
    - Break when all of the valid events are established (AND condition)
    - Break when all of the valid events are established at the same time (simultaneous AND condition)
    - Break when one of the valid events is established (OR condition)
  - At the time the debugger starts up, the hardware breaks have no effect.

### 7.5.1 Specify the Break Event

To set events, double-click to select the event you want to set from the event setting area of the H/W Break Point Setting Window. This opens the dialog box shown below.



Following events can be set by specifying Event Type in this dialog box.

---

**When FETCH is selected**

Breaks for the instruction fetch.

The screenshot shows a dialog box titled 'Fetch'. It has a 'Setting' section with the following fields: 'Range' set to '(addr) == Address1', 'Address1' set to '\_main', and 'Address2' set to '000000'. There are checkboxes for 'Function', 'Source File', and 'Function', all of which are currently unchecked. Below the settings, a text box displays the following information: 'ACCESS: FETCH', 'ADDRESS: \_main', and 'CONDITION: (addr) == 0F002C'.

**When DATA ACCESS is selected**

Breaks for the memory access.

The screenshot shows a dialog box titled 'Data'. It has a 'Setting' section with the following fields: 'Range' set to 'Data1 <= (data) <= Data2', 'Data 1' set to '0000', 'Data 2' set to '0000', 'Access' set to 'R/W', and 'Mask' checked with a value of 'FFFF'. Below the settings, a text box displays the following information: 'ACCESS: R/W', 'ADDRESS: \_data', and 'CONDITION: (addr) == 00042C, 0000 <= (data) <= 0000'.

**When BIT SYMBOL is selected**

Breaks for the bit access.

Bit

Address: 400 Bit No.: 2

Bit Symbol:

Condition

Access: WRITE

Value: 1

ACCESS: WRITE  
ADDRESS: \_pool  
CONDITION: (addr) == 000400, (data&0004) == 0004

**7.5.2 Specify the Combinatorial Condition**

To specify a combinatorial condition, specify the desired condition from the combinatorial condition specification area.

**When AND or OR is selected**

In the event specification area, the event used and a pass count for that event can be specified. To alter the pass count, while the event to alter is being selected, click the pass count value of that event.

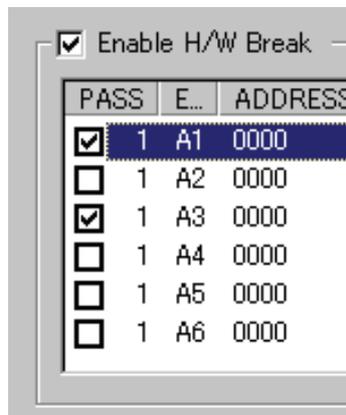
Enable H/W Break

PASS	E...	ADDRESS
<input checked="" type="checkbox"/>	1 A1	0000
<input type="checkbox"/>	1 A2	0000
<input checked="" type="checkbox"/>	1 A3	0000
<input type="checkbox"/>	1 A4	0000
<input type="checkbox"/>	1 A5	0000
<input type="checkbox"/>	1 A6	0000

---

### When AND (Same Time) is selected

In the event specification area, the event used can be specified. No pass counts can be specified.



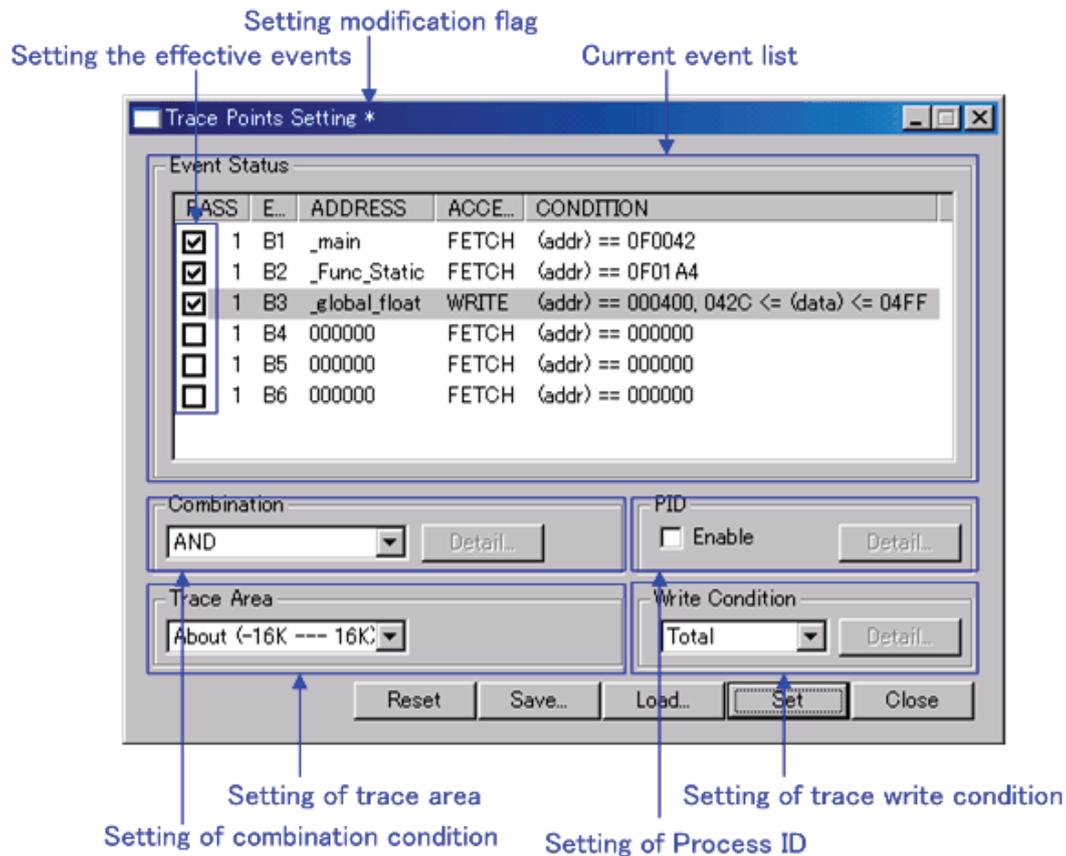
### 7.5.3 Command Button

The buttons on this window has the following meanings.

Button	Function
Reset	Discards the contents being displayed in the window and loads contents from the emulator in which they were set.
Save...	Saves the contents set in the window to a file.
Load...	Loads event information from a file in which it was saved.
Set	Sends the contents set in the window to the emulator.
Close	Closes the window.

## 7.6 Trace Point Setting Window

The Trace Point Setting window is used to set trace points.

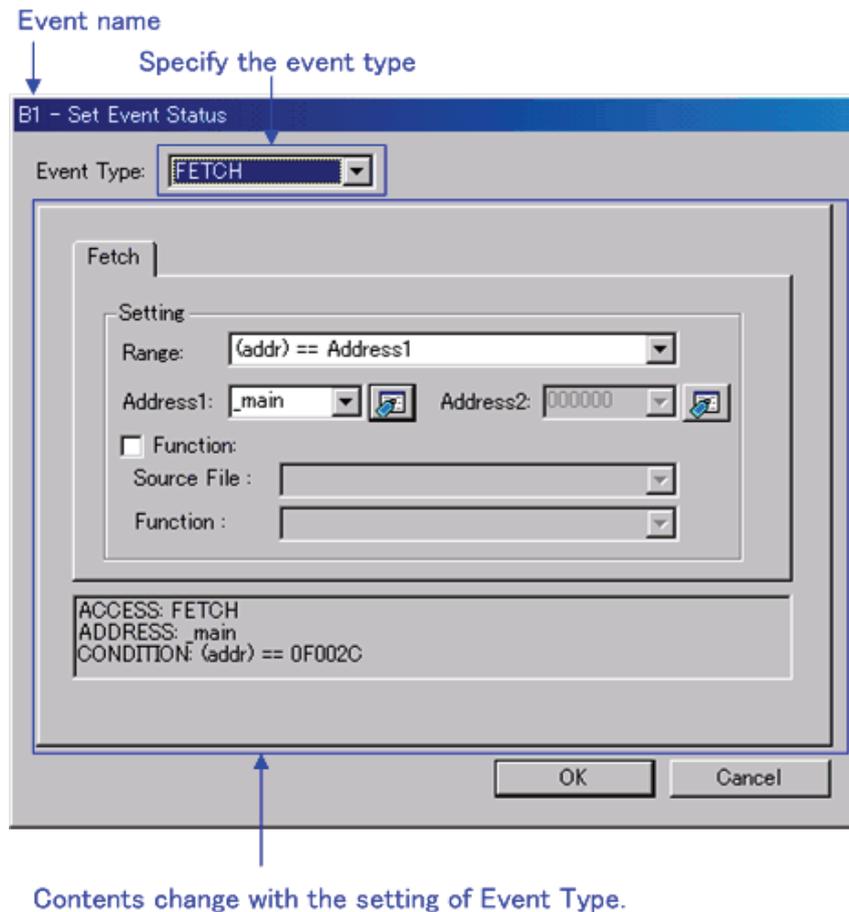


- The events listed below can be specified as trace events. If the contents of events are altered, they are marked by an asterisk (\*) on the title bar. The asterisks (\*) are not displayed after setting up the emulator.
  - Fetch, Memory Access, Bit Access
  - Events at up to two points can be used.
- The H/W Break Point Setting window and the Trace Point Setting windows use the same resource of the emulator. Use the MCU tab in the Init dialog box, in order to specify for which function the resources are used. On this tab, select the Enable the Trace Point Function check box.
- These events can be combined in one of the following ways:
    - Trace when all of the valid events are established (AND condition)
    - Trace when all of the valid events are established at the same time (simultaneous AND condition)
    - Trace when one of the valid events is established (OR condition)

---

## 7.6.1 Specify the Trace Event

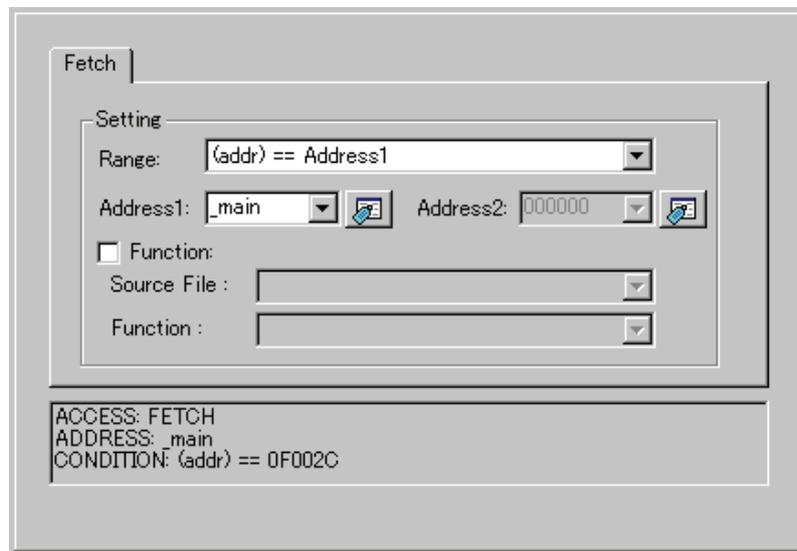
To set events, double-click to select the event you want to set from the event setting area of the Trace Point Setting Window. This opens the dialog box shown below.



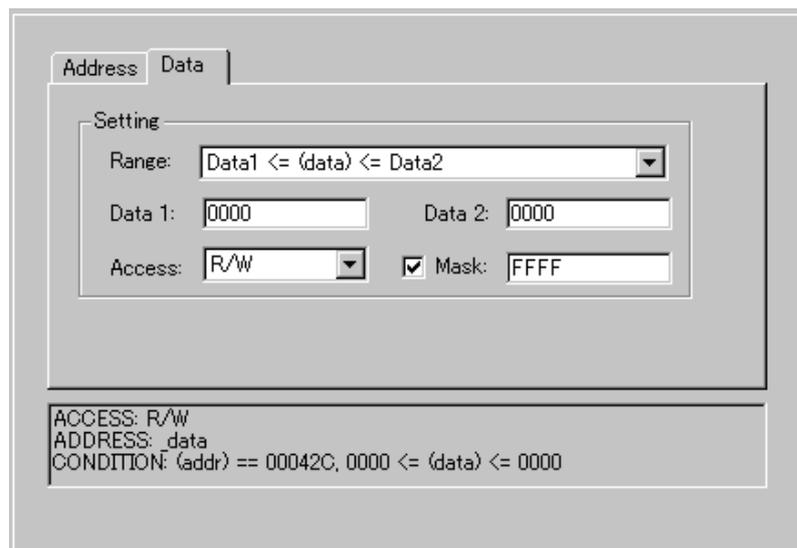
Following events can be set by specifying Event Type in this dialog box.

**When FETCH is selected**

Traces for the instruction fetch.

**When DATA ACCESS is selected**

Traces for the memory access.



---

### When BIT SYMBOL is selected

Traces for the bit access.

The screenshot shows a configuration dialog box with three main sections:

- Bit:** Contains two radio buttons. The first, labeled "Address:", is selected and has a dropdown menu showing "400" and a small icon to its right. The second, labeled "Bit Symbol:", is unselected and has an empty dropdown menu. To the right of these is a "Bit No." field with a dropdown menu showing "2".
- Condition:** Contains two dropdown menus. The first is labeled "Access:" and shows "WRITE". The second is labeled "Value:" and shows "1".
- Summary:** A text area at the bottom containing the following text:  
ACCESS: WRITE  
ADDRESS: \_pool  
CONDITION: (addr) == 000400, (data&0004) == 0004

### 7.6.2 Specify the Combinatorial Condition

To specify a combinatorial condition, specify the desired condition from the combinatorial condition specification area.

When AND or OR is selected

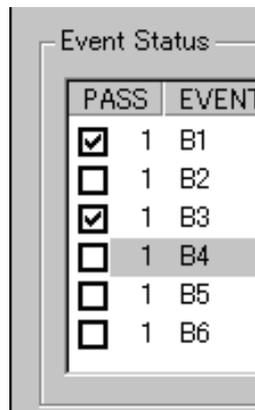
In the event specification area, the event used and a pass count for that event can be specified. To alter the pass count, while the event to alter is being selected, click the pass count value of that event.

The screenshot shows a dialog box titled "Event Status" containing a table with two columns: "PASS" and "EVENT".

PASS	EVENT
<input checked="" type="checkbox"/>	1 B1
<input type="checkbox"/>	1 B2
<input checked="" type="checkbox"/>	1 B3
<input type="checkbox"/>	1 B4
<input type="checkbox"/>	1 B5
<input type="checkbox"/>	1 B6

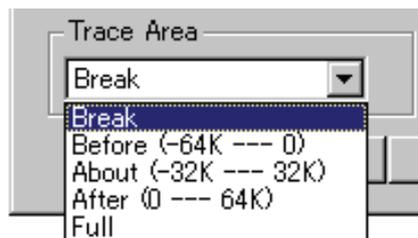
When AND (Same Time) is selected

In the event specification area, the event used can be specified. No pass counts can be specified.



### 7.6.3 Specify the Trace Range

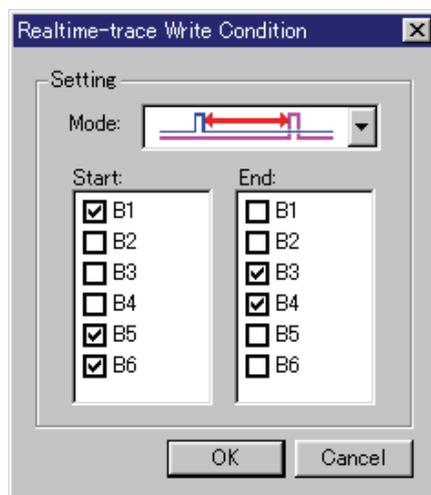
For the compact emulator debugger, 64K cycles equivalent of data can be recorded.



Break	Stores the 64K cycles (-64K to 0 cycles) to the point at which the target program stops.
Before	Stores the 64K cycles (-64K to 1 cycles) to the point at which the trace point is passed.
About	Stores the 64K cycles (-32K to 32K cycles) either side of the trace point.
After	Stores the 64K cycles (0 to 64K cycles) of trace data after the trace point.
Full	Stores the 64K cycles (-64K to 0 cycles) of trace data after the trace starts.

## 7.6.4 Specify the Trace Write Condition

Conditions for cycles to be written to trace memory can be specified.



Total	Writes all cycles.
Pick up	Writes only the cycles where specified condition holds true.
Exclude	Writes only the cycles where specified condition does not hold true.

Also, following three write modes are supported.

	Only cycles where specified event is established
	Cycles from where specified event is established to where specified event is not established
	Cycles from where start event is established to where end event is established

## 7.6.5 Command Button

The buttons on this window has the following meanings.

Button	Function
Reset	Discards the contents being displayed in the window and loads contents from the emulator in which they were set.
Save...	Saves the contents set in the window to a file.
Load...	Loads event information from a file in which it was saved.
Set	Sends the contents set in the window to the emulator.
Close	Closes the window.

## 7.7 Trace Window

The Trace Window is used to display the results of real-time trace measurement. The measurement result can be displayed in the following display modes.

### Bus mode

This mode allows you to inspect cycle-by-cycle bus information. The display content depends on the MCU and emulator system used. In addition to bus information, this mode allows disassemble, source line or data access information to be displayed in combination.

### Disassemble mode

This mode allows you to inspect the executed instructions. In addition to disassemble information, this mode allows source line or data access information to be displayed in combination.

### Data access mode

This mode allows you to inspect the data read/write cycles. In addition to data access information, this mode allows source line information to be displayed in combination.

### Source mode

This mode allows you to inspect the program execution path in the source program.

The measurement result is displayed when a trace measurement has finished. When a trace measurement restarts, the window display is cleared.

The range of a trace measurement can be altered in the Trace Point Setting Window. For details about this window, refer to "Referencing the Trace Point Setting Window." With default settings, the trace information immediately before the program has stopped is recorded.

### 7.7.1 Configuration of Bus Mode

When bus mode is selected, trace information is displayed in bus mode. Bus mode is configured as shown below.

The display content in bus mode differs depending on the MCU or emulator system used.

Cycle	Label	Address	Data	BUS	BIU	R/W	RWT	CPU	QB	B-T	Q-T	76543210	H' m' s: ms. us
-07958		00042C	00DE	16b	DW	R	0	RW	0	1	1	11111111	00'00'00:055.114
-07957		0F01DC	7AF3	16b	IW	R	0	--	2	1	1	11111111	00'00'00:055.114
-07956	_Func_Exec	0F01DE	F27C	16b	IW	R	0	--	4	1	1	11111111	00'00'00:055.115
-07955		00042C	00DF	16b	DW	W	0	CB	3	1	1	11111111	00'00'00:055.115
-07954		0007E8	0083	16b	DW	R	0	--	3	1	1	11111111	00'00'00:055.115
-07953		0007EA	FF0F	16b	DB	R	0	--	3	1	1	11111111	00'00'00:055.115
-07952		0007EA	FF0F	16b	--	--	1	--	3	1	1	11111111	00'00'00:055.115
-07951		0F0083	FDFF	16b	IB	R	0	QC	1	1	1	11111111	00'00'00:055.115
-07950		0F0084	012C	16b	IW	R	0	--	3	1	1	11111111	00'00'00:055.115
-07949		0F0086	F50F	16b	IW	R	0	CB	4	1	1	11111111	00'00'00:055.115

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



## 7.7.2 Configuration of Disassemble Mode

When disassemble mode is selected while bus mode is unselected, trace information is displayed in disassemble mode. Disassemble mode is configured as shown below.

Cycle	Address	Obj-code	Label	Mnemonic	h" m' s: ms. us
-18960	0F0199	730BF0		MOV.W RO, -10H[FB]	00"00'00:054.427
-18957	0F019C	C91BFD		ADD.W #1H, -3H[FB]	00"00'00:054.427
-18953	0F019F	FEC1		JMP.B F0161H	00"00'00:054.427
-18948	0F0161	778BFD0A00		CMP.W #000AH, -3H[FB]	00"00'00:054.428
-18943	0F0166	7DCA39		JGE F01A1H	00"00'00:054.428
-18938	0F01A1	7DF2		EXITD	00"00'00:054.428
-18929	0F0087	F50600		JSR.W _random_ F008EH	00"00'00:054.429
-18920	0F008E	7CF204	_random_access	ENTER #04H	00"00'00:054.429
-18916	0F0091	FDD40B0F		JSR.A _rand F0BD4H	00"00'00:054.430
-18906	0F0BD4	75C06D4E	_rand	MOV.W #4E6DH, RO	00"00'00:054.430
-18904	0F0BD8	75C2C641		MOV.W #41C6H, R2	00"00'00:054.430
-18902	0F0BDC	754F4004		PUSH.W 0440H	00"00'00:054.430
-18898	0F0BE0	754F3E04		PUSH.W 043EH	00"00'00:054.431
-18893	0F0BE4	FE01		JMP.B FOBE6H	00"00'00:054.431
-18889	0F0BE6	FD1COCOF		JSR.A __i4mulU FOC1CH	00"00'00:054.431
-18879	0F0C1C	EC50	__i4mulU	PUSHM R1, R3	00"00'00:054.432
-18875	0F0C1E	75B107		MOV.W 7H[SP], R1	00"00'00:054.432
-18870	0F0C21	7121		MUL.W R2, R1	00"00'00:054.432
-18865	0F0C23	7312		MOV.W R1, R2	00"00'00:054.433
-18863	0F0C25	75B109		MOV.W 9H[SP], R1	00"00'00:054.433
-18859	0F0C28	7101		MUL.W RO, R1	00"00'00:054.433
-18854	0F0C2A	A112		ADD.W R1, R2	00"00'00:054.433

### (1) Address display area:

Shows addresses corresponding to instructions. Double-click here to bring up a dialog box to search for addresses.

### (2) Object code display area:

Shows the object codes of instructions.

### (3) Label display area:

Shows labels corresponding to instruction addresses. Double-click here to bring up a dialog box to search for addresses.

### (4) Mnemonic display area:

Shows the mnemonics of instructions.

Other display areas are the same as in bus mode.

In addition to disassemble information, the window can display source line or data access information in combination. In this case, the display will be similar to the one shown below.

Cycle	Address	Obj-code	Label	Mnemonic	DataAccess	h" m' s: ms. us
-19026	0F0161	778BFD0A00		CMP.W #000AH, -3H[FB]	(0007E3 09 R ) (0007E4 00 R )	00"00'00:054.423
-19021	0F0166	7DCA39		JGE F01A1H		00"00'00:054.423
-19019	0F0169	C661FF		MOV.B #61H, -1H[FB]		00"00'00:054.423
-19017	0F016C	D90BF9		MOV.W #0H, -7H[FB]	(0007E5 61 W )	00"00'00:054.423
-19014	0F016F	D90BFB		MOV.W #0H, -5H[FB]	(0007DF 00 W )	00"00'00:054.423

### 7.7.3 Configuration of Data Access Mode

When data access mode is selected while bus mode and disassemble mode are unselected, trace information is displayed in data access mode. Data access mode is configured as shown below.

Cycle	Label	DataAccess	h' m' s: ms. us
-05012	_global_struct	(00047C 42 W )	00"00'00:055.299
-05007		(00047D 05 W )	00"00'00:055.299
-05006		(00047E 00 W )	00"00'00:055.299
-05001		(00047F 06 W )	00"00'00:055.299
-05000		(000480 00 W )	00"00'00:055.299
-04995		(000481 00 W )	00"00'00:055.300
-04994		(000482 00 W )	00"00'00:055.300
-04989		(000483 10 W )	00"00'00:055.300
-04984		(000484 0020 W )	00"00'00:055.300
-04977		(000486 0030 W )	00"00'00:055.301
-04972		(0007E4 0000 W )	00"00'00:055.301
-04971		(0007E4 0000 R )	00"00'00:055.301
-04964		(0007E2 0000 W )	00"00'00:055.302
-04963		(0007E2 0000 R )	00"00'00:055.302
-04957		(0007E4 0000 R )	00"00'00:055.302
-04949		(0007E2 0000 R )	00"00'00:055.302
-04939	_global_array	(00044A 0000 W )	00"00'00:055.303
-04938		(0007E2 0000 R )	00"00'00:055.303
-04935		(0007E2 0001 W )	00"00'00:055.303
-04929		(0007E2 0001 R )	00"00'00:055.304
-04924		(0007E4 0000 R )	00"00'00:055.304
-04916		(0007E2 0001 R )	00"00'00:055.305

(1)

**(1) Data access display area:**

Shows data access information. If the information displayed here is "000400 1234 W," for example, it means that data "1234H" was written to the address 000400H in 2-byte width.

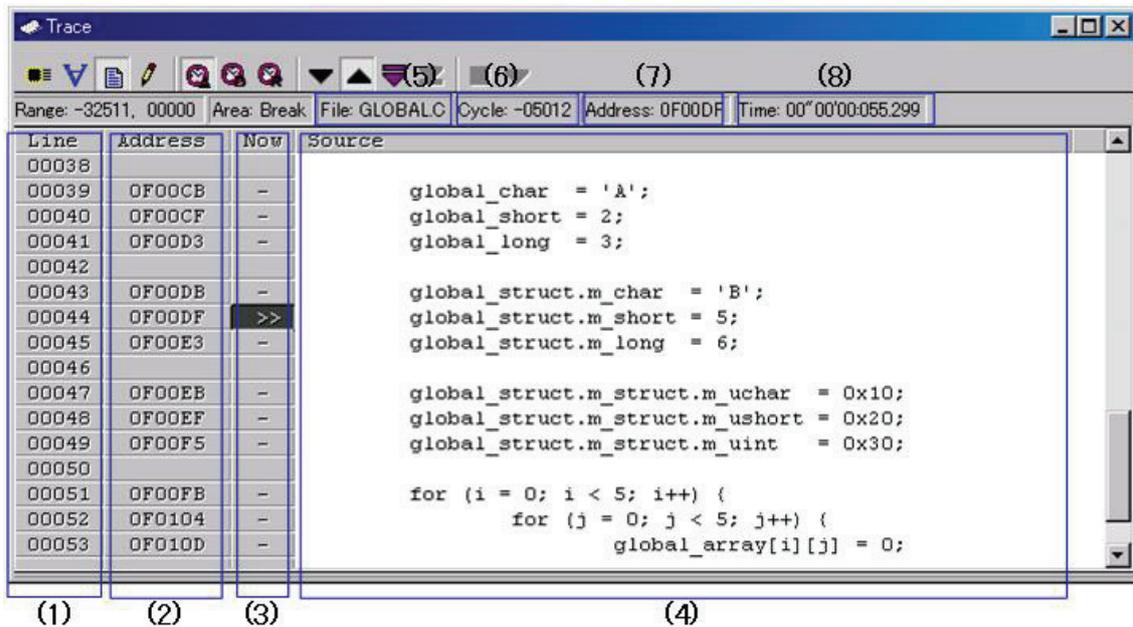
Other display areas are the same as in bus mode.

In addition to data access information, the window can display source line information in combination. In this case, the display will be similar to the one shown below.

Cycle	Label	DataAccess	h' m' s: ms. us
-05012	GLOBAL.C, 44: _global_struct	(00047C 42 W )	00"00'00:055.299
-05007	GLOBAL.C, 45: _global_struct	(00047D 05 W )	00"00'00:055.299
-05006		(00047E 00 W )	00"00'00:055.299
-05001		(00047F 06 W )	00"00'00:055.299
-05000		(000480 00 W )	00"00'00:055.299
-04995	GLOBAL.C, 47: _global_struct	(000481 00 W )	00"00'00:055.300
-04994		(000482 00 W )	00"00'00:055.300

## 7.7.4 Configuration of Source Mode

When only source mode is selected, trace information is displayed in source mode. Source mode is configured as shown below.



**(1) Line number display area:**

Shows the line number information of the displayed file. Double-click here to bring up a dialog box to change the displayed file.

**(2) Address display area:**

Shows addresses corresponding to source lines. Double-click here to bring up a dialog box to search for addresses.

**(3) Referenced cycle display area:**

Shows the currently referenced cycle that is marked by ">>." Furthermore, the addresses corresponding to source lines, if any, are marked by "-."

**(4) Source display area:**

Shows the content of the source file.

**(5) File name:**

Shows the file name of the currently displayed source file.

**(6) Referenced cycle:**

Shows the currently referenced cycle.

**(7) Referenced address:**

Shows the address corresponding to the currently referenced cycle.

**(8) Referenced time:**

Shows the time information corresponding to the currently referenced cycle.

Other display areas are the same as in bus mode.

---

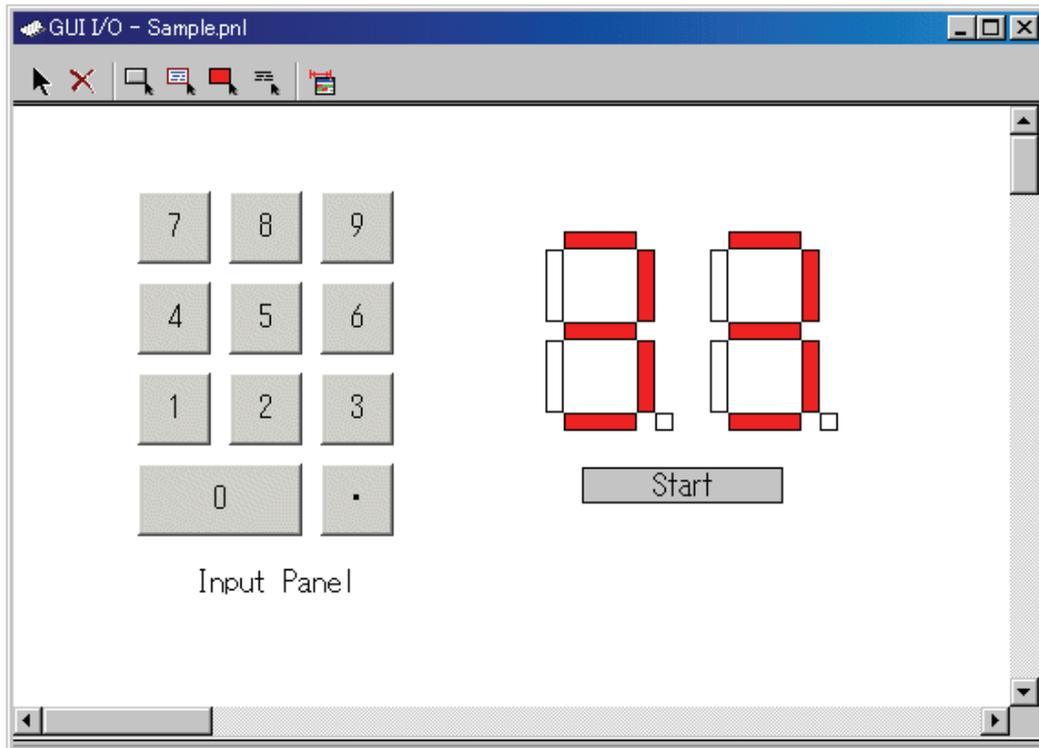
## 7.7.5 Extended Menus

This window has the following popup menus that can be brought up by right-clicking in the window.

Menu	Function	
BUS	Display the information of BUS mode.	
DIS	Display the information of Disassemble mode.	
SRC	Display the information of Source mode.	
DATA	Display the information of Data access mode.	
View	Cycle...	Changes the displayed position by specifying a cycle.
	Address...	Changes the displayed position by searching an address.
	Source...	Display a selected source file.
Time	Absolute Time	Shows elapsed time from the time the program started running up to now in terms of absolute time.
	Differences	Shows a differential time from the immediately preceding displayed cycle.
	Relative Time	Shows a relative time from the currently selected cycle.
Trace	Forward	Changes the direction of search to forward direction.
	Backward	Changes the direction of search to reverse direction.
	Step	Searches in Step mode in the specified direction of search.
	Come	Searches in Come mode in the specified direction of search.
	Stop	Stops trace measurement in the middle and displays the measured content at the present point of time.
	Restart	Restarts trace measurement.
Layout...	Change layout of the corrent view.	
Copy	Copy selected lines.	
Save...	Save trace data to file.	
Load...	Load trace data from file.	
Toolbar display	Display toolbar.	
Customize toolbar...	Open toolbar customize dialog box.	
Allow Docking	Allow window docking.	
Hide	Hide window.	

## 7.8 GUI I/O Window

The GUI I/O window allows you for port input by creating a user target system key input panel (button) in the window and clicking the created button. And this window also allows you to implement the user target system output panel in the window.



- You can arrange the following parts on the window.
  - Label (character string)
    - Displays/erases a character string specified by the user when any value is written to the specified address (bit).
  - LED
    - Changes the display color of any area when any value is written to the specified address (bit). (Substitution for LED ON)
  - Button
    - A virtual port input can be executed at the time the button is pressed.
  - Text
    - Display the text string.
- You can also save the created panel in a file and reload it.
- You can set up to 200 address points to the created part. If different addresses are set to the individual parts, you can arrange up to 200 parts.

---

## 7.8.1 Extended Menus

This window has the following popup menus that can be brought up by right-clicking in the window.

<b>Menu</b>	<b>Function</b>
Select Item	Select an I/O item.
Delete	Delete the selected I/O item.
Copy	Copy the selected I/O item.
Paste	Paste the copied I/O item.
Create Button	Create a new button item.
Create Label	Create a new label item.
Create LED	Create a new LED item.
Create Text	Create a new text item.
Display grid	Display the grid line.
Save...	Save I/O panel file.
Load...	Load I/O panel file.
Sampling Period...	Set RAM monitor sampling period.
Toolbar display	Display toolbar.
Customize toolbar...	Open toolbar customize dialog box.
Allow Docking	Allow window docking.
Hide	Hide window.

## 8. Target Dependent Commands

The table below lists the commands that can be used in the H8/300H Tiny compact emulator.

- For details about the expressions that can be used in the command arguments, refer to "9.1 Expressions" in the pages to follow.
- For details about common command specifications, and on how to set the command line, refer to "Using the Command Line" in the "High-performance Embedded Workshop Help Library."

No.	Command Name	Short Name	Contents
1	ASSEMBLE	AS	Assembles one line at a time beginning with a specified address.
2	BREAKPOINT	BP	Sets a software breakpoint.
3	BREAKPOINT_CLEAR	BC	Clears software breakpoints.
4	BREAKPOINT_DISPLAY	BI	Shows software breakpoints.
5	BREAKPOINT_ENABLE	BE	Enable/disables software breakpoints.
6	CLOCK_SOURCE	CKS	Select/inspects the clock supplied to the MCU.
7	DISASSEMBLE	DA	Shows the disassembled result of a specified range.
8	EEPROM_COPY	EEPC	Copies an area of the EEPROM memory.
9	EEPROM_DISPLAY	EEPD	Shows the content of the EEPROM memory.
10	EEPROM_EDIT	EEPE	Changes the content of the EEPROM memory.
11	EEPROM_FILL	EEPF	Collectively changes the content of an EEPROM memory area.
12	EEPROM_SETMODE	EEPS	Specifies mode in which data will be written to the EEPROM memory.
13	EVALUATE	EV	Shows the value of a specified assembler expression.
14	HARDWAREBREAK_COMBINATION	HBC	Specifies a combinatorial condition of hardware breakpoints.
15	HARDWAREBREAK_DISPLAY	HBD	Shows hardware breakpoints.
16	HARDWAREBREAK_ENABLE	HBE	Inspect/sets break mode.
17	HARDWAREBREAK_SET	HBS	Specifies a hardware breakpoint.
18	REGISTER_DISPLAY	RD	Inspects the value of a specified register.
19	REGISTER_SET	RS	Sets the value of a specified register.
20	RESET	RE	Resets the target program.
21	TRACE_DISPLAY	TD	Shows a real-time trace result by displaying bus signals or disassembled code.
22	TRACEPOINT_AREA	TPA	Specifies a trace range.
23	TRACEPOINT_COMBINATION	TPC	Specifies a combinatorial condition of trace points.
24	TRACEPOINT_DISPLAY	TPD	Shows trace points.
25	TRACEPOINT_SET	TPS	Specifies a trace point.

---

## 9. Expressions

### 9.1 Expressions

An expression can be comprised of the following constituent elements:

- Constants
- Register Variables
- Symbols and labels
- String literals
- Operators

#### 9.1.1 Constants

A binary, octal, decimal, or hexadecimal number can be entered. The radix of a numeric value must be identified by adding the appropriate symbol at the beginning of the value.

	<b>Hexadecimal</b>	<b>Decimal</b>	<b>Octal</b>	<b>Binary</b>
<b>Prefix</b>	0x, 0X, H', h'	D', d'	O', o'	B', b'
<b>ex.</b>	0xAB24 H'AB24 h'AB24	D'1234 d'1234	O'1234 o'1234	B'1001010010 b'1001010010

If numeric values are entered using a predetermined radix, the symbol representing the radix can be omitted.

Use the RADIX command to set a predetermined radix.

#### 9.1.2 Symbols and Labels

The symbols/labels defined in the target program and those that are defined by the Assemble command can be used.

Alphanumeric characters, underscore (\_), and dollar mark (\$) can be used in the symbol/label names. However, numerals cannot be used at the beginning of the symbol/label name.

The symbol/label names are case-sensitive.

### 9.1.3 Register Variables

Use a register variable when a register value is to be used in an expression. To enter a register variable, use a register name that is prefixed with the symbol "#." The usable register names are listed below.

Target	Register Name
H8/300H Tiny Compact Emulator Debugger	PC, CCR, SP, FP ER0, ER1, ER2, ER3, ER4, ER5, ER6, ER7 E0, E1, E2, E3, E4, E5, E6, E7 R0, R1, R2, R3, R4, R5, R6, R7 R0H, R1H, R2H, R3H, R4H, R5H, R6H, R7H R0L, R1L, R2L, R3L, R4L, R5L, R6L, R7L

The register names are not case-sensitive. Upper-case and lower-case letters can both be used.

### 9.1.4 String Literals

A specified character or string is converted into ASCII code for use as a character constant.

Include the character or string in single quotes.

The string must be within 4 characters (32 bits) long. If this limit is exceeded, an error results.

Escape characters are not supported. (If you enter '\n' for example, it is recognized as 0x5C6E, and not as 0x0A.)

### 9.1.5 Operators

The following lists the operators that can be written in an expression.

The operators have priority, level 1 representing the highest priority, and level 13 the lowest. If the operators used in an expression have the same priority, calculation is performed from left to right of the expression.

Operator	Name	Priority
()	parenthesis	Level 1
+, -, ~, !	Unary plus operator, Unary minus operator, Bitwise OR operator, Logical NOT operator	Level 2
*, /	Multiplication operator, Division operator	Level 3
%	Modulo operator	Level 4
+, -	Addition operator, Subtraction operator	Level 5
>>, <<	Bitwise right-shift operator, Bitwise left-shift operator	Level 6
<, <=, >, >=	Relational Operators	Level 7
==, !=	Equality operators	Level 8
&	Bitwise AND operator	Level 9
^	Bitwise exclusive OR operator	Level 10
	Bitwise OR operator	Level 11
&&	Logical AND operator	Level 12
	Logical OR operator	レベル 13

---

## 9.2 Writing C/C++ Expressions

You can use C/C++ expressions consisting of the tokens shown below for registering C watchpoints and for specifying the values to be assigned to C watchpoints.

Token	Example
Immediate values	10, 0x0a, 012, 1.12, 1.0E+3
Scope	::name, classname::member
Mathematical operators	+, -, *, /
Pointers	*, **, ...
Reference	&
Sign inversion	-
Member reference using dot operator	Object.Member
Member reference using arrow	Pointer->Member, this->Member
Pointers to Members	Object.*var, Pointer->*var
Parentheses	(, )
Arrays	Array[2], DArray[2] [3], ...
Casting to basic types	(int), (char*), (unsigned long *), ...
Casting to typedef types	(DWORD), (ENUM), ...
Variable names and function names	var, i, j, func, ...
Character constants	'A', 'b', ...
Character string literals	"abcdef", "I am a boy.", ...

### Immediate Values

You can use hexadecimals, decimals, octals as immediate values. Values starting with 0x are processed as hexadecimals, those with 0 as octals, and those without either prefix as decimals. Floating-point numbers can also be used to assign values to variables.

#### Notes

- You cannot register only immediate values as C watchpoints.
- The immediate value is effective only when it is used in C/C++ language expressions that specify C/C++ watchpoints or when it is used to specify the value to be assigned to those expressions. When using floating-point numbers, operation cannot be performed on an expression like 1.0+2.0.

### Mathematical Operators

You can use the addition (+), subtraction (-), multiplication (\*), and division (/) mathematical operators. The following shows the order of priority in which they are evaluated.

(\*), (/), (+), (-)

#### Notes

- There is no support currently for mathematical operators for floating point numbers.

### Pointers

Pointers are indicated by the asterisk (\*). You can use pointer to pointers \*\*, and pointer to pointer to pointers \*\*\*, etc.

Examples: `**variable_name`, `***variable_name`, etc.

#### Notes

Immediate values cannot be processed as pointers. That is, you cannot specify `*0xE000`, for example.

**Reference**

References are indicated by the ampersand (&). You can only specify "&variable\_name".

**Sign Inversion**

Sign inversion is indicated by the minus sign (-). You can only specify "-immediate\_value" or "-variable\_name". No sign inversion is performed if you specify 2 (or any even number of) minus signs.

**Notes**

- There is no support currently for sign inversion of floating point numbers.

**Member Reference Using Dot Operator**

You can only use "variable\_name.member\_name" for checking the members of structures and unions using the dot operator.

**Example:**

```
class T {
public:
    int member1;
    char member2;
};
class T t_cls;
class T *pt_cls = &t_cls;
```

In this case, t\_cls.member1, (\*pt\_cls).member2 correctly checks the members.

**Member Reference Using Arrow**

You can only use "variable\_name->member\_name" for checking the members of structures and unions using the arrow.

**Example:**

```
class T {
public:
    int member1;
};
class T t_cls;
class T *pt_cls = &t_cls;
```

In this case, (&t\_cls)->member1, pt\_cls->member2 correctly checks the members.

**Pointers to Members**

Pointers to members using the ".\*" or "->\*" operator can be referred only in the forms of variable name .\* member name or variable name ->\* member name.

**Example:**

```
class T {
public:
    int member;
};
class T t_cls;
class T *pt_cls = &t_cls;

int T::*mp = &T::member;
```

In this case, t\_cls.\*mp and tp\_cls->\*mp can correctly reference the variable of pointer-to-member type.

---

### Note

- Note that the expression `*mp` cannot be considered as the variable of pointer-to-member type.

### Parentheses

Use the '(' and ')' to specify priority of calculation within an expression.

### Arrays

You can use the '[' and ']' to specify the elements of an array. You can code arrays as follows: "variable\_name [ (element\_No or variable) ] ", "variable\_name [ (element\_No or variable) ] [ (element\_No or variable) ] ", etc.

### Casting to Basic Types

You can cast to C basic types char, short, int, and long, and cast to the pointer types to these basic types. When casting to a pointer type, you can also use pointers to pointers and pointers to pointers to pointers, etc.

Note that if signed or unsigned is not specified, the default values are as follows:

Basic type	Default
char	unsigned
short	signed
int	signed
long	signed

### Notes

- Of the basic types of C++, casts to bool type, wchar\_t type, and floating-point type (float or double) cannot be used.
- Casts to register variables cannot be used.

### Casting to typedef Types

You can use casting to typedef types (types other than the C basic types) and the pointer types to them. When casting to a pointer type, you can also use pointers to pointers and pointers to pointers to pointers, etc.

### Notes

- You cannot cast to struct or union types or the pointers to those types.

### Variable Names and Function Names

Function names that begin with English alphabets as required under C conventions can be used. In the case of C++, no function names can be used.

### Character Constants

You can use characters enclosed in single quote marks (') as character constants. For example, 'A', 'b', etc. These character constants are converted to ASCII code and used as 1-byte immediate values.

### Notes

- You cannot register character constants only as C watchpoints.
- Character constants are valid only when used in a C/C++ expression that specifies a C watchpoint, and when specifying a value to be assigned (character constants are processed in the same manner as immediate values).

**Character String Literals**

You can use character strings enclosed in double quote marks (") as character string literals. Examples are "abcde", "I am a boy.", etc.

**Notes**

- Character string literals can only be placed on the right side of an assignment operator in an expression. They can only be used when the left side of the assignment operator is a char array or a char pointer type. In all other cases, a syntax error results.

---

## 9.3 Display Format of C/C++ Expressions

C/C++ expressions in the data display areas of the C Watch Windows are displayed as their type name, C/C++ expression (variable name), and result of calculation (value), as shown below.

The following describes the display formats of the respective types.

### Enumeration Types

- When the result (value) of calculation has been defined, its name is displayed.

```
(DATE) date = Sunday (all Radices)
```

- If the result (value) of calculation has not been defined, it is displayed as follows:

```
(DATE) date = 16 (when Radix is in initial state)
(DATE) date = 0x10 (when Radix is hex)
(DATE) date = 0000000000010000B (when Radix is binary)
```

### Basic Types

- When the result of calculation is a basic type other than a char type or floating point type, it is displayed as follows:

```
(unsigned int) i = 65280 (when Radix is in initial state)
(unsigned int) i = 0xFF00 (when Radix is hex)
(unsigned int) i = 1111111100000000B (when Radix is binary)
```

- When the result of calculation is a char type, it is displayed as follows:

```
(unsigned char) c = 'J' (when Radix is in initial state)
(unsigned char) c = 0x4A (when Radix is hex)
(unsigned char) c = 10100100B (when Radix is binary)
```

- When the result of calculation is a floating point, it is displayed as follows:

```
(double) d = 8.207880399131839E-304 (when Radix is in initial state)
(double) d = 0x10203045060708 (when Radix is hex)
(double) d = 000000010.....1000B (when Radix is binary)
(..... indicates abbreviation)
```

### Pointer Types

- When the result of calculation is a pointer type to other than a char\* type, it is displayed in hexadecimal as follows:

```
(unsigned int *) p = 0x1234 (all Radices)
```

- When the result of calculation is a char\* type, you can select the display format of the string or a character in the C Watch window's menu [Display String].

### **string types**

```
(unsigned char *) str = 0x1234 "Japan" (all Radices)
```

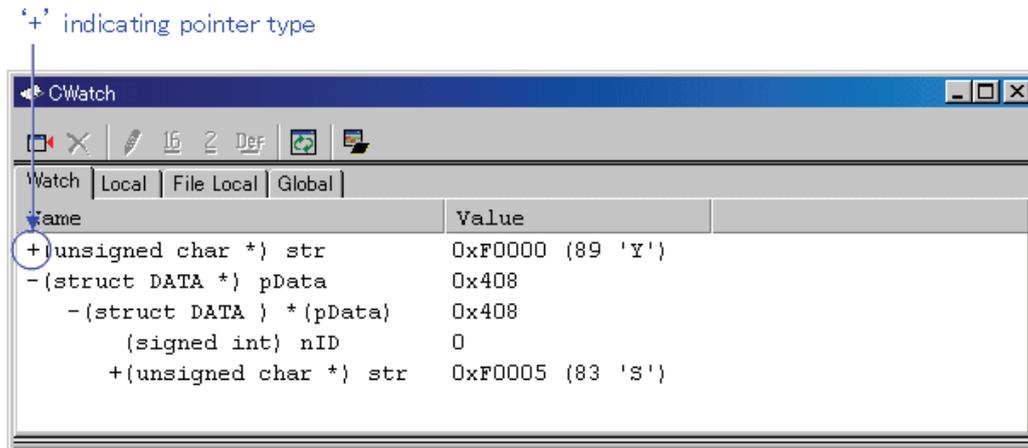
### **character types**

```
(unsigned char *) str = 0x1234 (74 'J') (all Radices)
```

1 When the result of calculation is a char\* type, it is displayed as follows:

```
(unsigned char *) str = 0x1234 "Jap (all Radices)
```

If the string contains a non-printing code prior to the code to show the end of the string (0), it is displayed up to the non-printing character and the closing quote mark is not displayed.



You can double-click on lines indicated by a '+' to see the members of that structure or union. The '+' changes to a '-' while the members are displayed. To return to the original display, double click the line, now indicated by the '-'.

### Array Types

- When the result of calculation is an array type other than a char [ ] type, the starting address is displayed in hex as follows:

```
(signed int [10]) z = 0x1234 (all Radices)
```

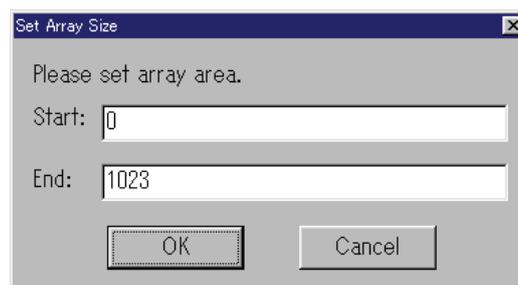
- When the result of calculation is a char [ ] type, it is displayed as follows:

```
(unsigned char [10]) str = 0x1234 "Japan" (all Radices)
```

If the string contains a non-printing code prior to the code to show the end of the string (0), it is displayed up to the non-printing character and the closing quote mark is not displayed.

```
(unsigned char [10]) str = 0x1234 "Jap(all Radices)
```

Also if the string contains more than 80 characters, the closing quote mark is not displayed. When the C/C++ expression is an array type as same as pointer type, a '+' is display to the left of the type name. You can see the elements of the array by using this indicating. (for the details, refer Pointer types) When the number of the array elements is more than 1000, the following dialog box open. Specify the number of the elements in the dialog box.



---

The elements from the index specified in "Start" to the index specified in "End" are displayed. If you specify the value more than the max index of the array, the value is regarded as max index of the array. When you click the "Cancel" button, the elements are not displayed.

### Function Types

- When the result of calculation is a function type, the starting address is displayed in hex as follows:

```
(void()) main = 0xF000 (all Radices)
```

### Reference Types

When the result of calculation is a reference type, the reference address is displayed in hex as follows:

```
(signed int &) ref = 0xD038 (all Radices)
```

### Bit Field Types

- When the result of calculation is a bit field type, it is displayed as follows:

```
(unsigned int :13) s.f = 8191 (when Radix is in initial state)
(unsigned int :13) s.f = 0x1FFF (when Radix is hex)
(unsigned int :13) s.f = 1111111111111B (when Radix is binary)
```

### When No C Symbol is Found

- If the calculated expression contained a C symbol that could not be found, it is displayed as follows:

```
() x = <not active> (all Radices)
```

### Syntax Errors

- When the calculated expression contains a syntax error, it is displayed as follows:

```
() str*(p = <syntax error> (all Radices)
(where str*(p is the syntax error)
```

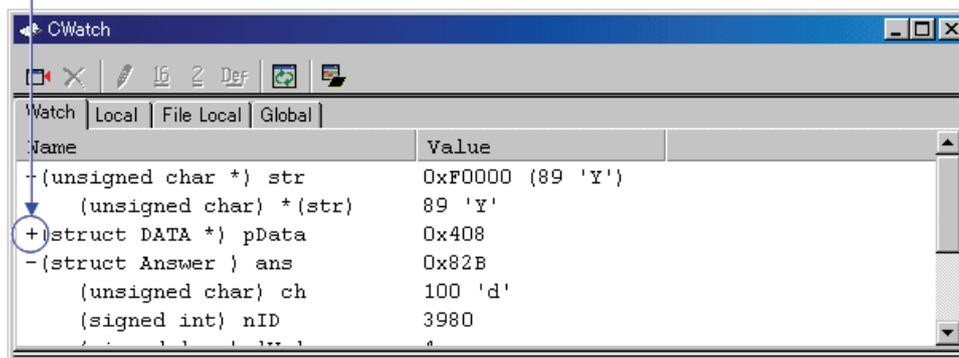
### Structure and Union Types

- When the result of calculation is a structure or union type, the address is displayed in hex as follows:

```
(Data) v = 0x1234 (all Radices)
```

If, as in structures and unions, the C/C++ expression consists of members, a '+' is displayed to the left of the type name (tag name).

'+' indicating structure or union



You can double-click on lines indicated by a '+' to see the members of that structure or union. The '+' changes to a '-' while the members are displayed. To return to the original display, double click the line, now indicated by the '-'. This function allows you to check the members of structures and unions.

**Attention**

If a variable is declared with the same name as the type definition name declared by typedef, you cannot reference that variable.

**Register Variables**

When the result of calculation is a register variable, "register" is displayed to the left of the type name as follows:

```
(register signed int) j = 100
```

---

## 10. Display the Cause of the Program Stoppage

If the program is stopped by the debug function, the cause of the stoppage is displayed in the Output window or Status window ([Platform] sheet).

The contents of a display and the meaning of "the cause of the stoppage" are as follows.

Display	The cause of the stoppage
Halt	The stop by the [Halt Program] button/menu
S/W break	Software break
H/W event, Combination	Hardware break, logical combination AND or AND(same time)condition was met
H/W event, Combination, Ax	Hardware break, logical combination OR condition was met (Ax: The event number of which condition was met.)
H/W event, State transition, from xx	Hardware break, State Transition condition was met (from xx: previous state (start, state1, state2))
H/W event, State transition, Timeout	Hardware break, State Transition, Time Out condition was met

### Note

To be able to show the cause of break or not depends on the connected target. Some targets may always show "Halt" or show "---".

# 11. Attention

## 11.1 Common Attention

### 11.1.1 File operation on Windows

The following points should be noted:

#### File Name and Directory Name

- Do not use directory names or filenames that include blanks.
- Operation is not guaranteed if your directory names and filenames include kanji.
- Use only one period in a filename.

#### Specify the File and Directory

- You cannot use "." to specify two levels upper directories.
- You cannot use a network pathname. You must allocate a drive.

### 11.1.2 Area where software breakpoint can be set

The area which can be set for software breakpoint varies depending on the type of MCU. Please refer to "Area where software breakpoint can be set" for details.

### 11.1.3 Get or set C variables

- If a variable is declared with the same name as the type definition name declared by typedef, you cannot reference that variable.
- Values cannot be changed for register variables and bit fields.
- Values cannot be changed for 64 bit width variables (long long, double, and so on).
- Values cannot be changed for C/C++ expressions that do not indicate the memory address and size.
- For the sake of optimization, the C compiler may place different variables at the same address. In this case, values of the C variable may not be displayed correctly.
- Literal character strings can only be substituted for char array and char pointer type variables.
- No arithmetic operations can be performed on floating point types.
- No sign inversion can be performed on floating point types.
- Casting cannot be performed on floating point types.
- Casting cannot be performed on register variables.
- Casting cannot be performed on structure types, union types, or pointer types to structure or union types.
- Character constants and literal character strings cannot contain escape sequences.

---

#### **11.1.4 Function name in C++**

- When you input the address using the function name in setting display address, setting break points, and so on, you can not specify the member function, operator function, and overloaded function, of a class.
- You can not use function names for C/C++ expression
- In address value specifying columns of dialog boxes, no addresses can be specified using function names.

#### **11.1.5 Debugging multi modules**

If you register two or more absolute module file in one session, you can download only one file in same time.

If you register one absolute module file and one or more machine language file in one session, you can download all file in same time.

#### **11.1.6 Synchronized debugging**

Synchronized debugging function is not available.

#### **11.1.7 Compact Emulator reset switch**

If system reset of the compact emulator does not function normally, terminate the debugger, turn ON the compact emulator again, and restart the debugger.

Then re-download the program.

## 11.2 Attention of the H8/300H Tiny Debugger

Attention of the H8/300H Tiny Debugger are shown below.

- Map of stack area used by the compact emulator
- Hardware break function
- Hardware Event

### 11.2.1 Map of stack area used by the compact emulator

The compact emulator uses the interrupt stack area as its work area (20 bytes).

When debugging, allocate a sufficient interrupt stack area consisting of the regularly used size plus 20 bytes.

### 11.2.2 Hardware break function

While running program, the following operations are not performed:

- execute `HARDWAREBREAK_ENABLE` command
- open H/W break point setting window

### 11.2.3 Hardware Event

If you specify word-length (2-byte length) data from an odd address as an event in the following data accesses, the event is not detected. Also, even when any other bit of the address that contains a specified bit is accessed during bit access, the event may become effective.

- Hardware Break Event
- Real-time Trace Event

---

[MEMO]

---

H8/300H Tiny Compact Emulator Debugger  
User's Manual

Publication Date: May. 01, 2005      Rev.1.00

Published by: Sales Strategic Planning Div.  
Renesas Technology Corp.

Edited by: Microcomputer Tool Development Department  
Renesas Solutions Corp.

# H8/300H Tiny Compact Emulator Debugger User's Manual



Renesas Electronics Corporation

1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan

REJ10J0933-0100