

To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

"Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

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

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

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

SuperH™ Family E10A Emulator

User's Manual

Renesas Microcomputer
Development Environment
System

SuperH™ Family E10A
HS0005KCM01HE

NOTICE:

There are corrections on page 219 in this document.

Keep safety first in your circuit designs!

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

Notes regarding these materials

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

IMPORTANT INFORMATION

READ FIRST

- **READ this user's manual before using this emulator product.**
- **KEEP the user's manual handy for future reference.**

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

Emulator Product:

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

- Emulator
- User system interface cable

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

Purpose of the Emulator Product:

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

Limited Applications:

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

Improvement Policy:

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

Target User of the Emulator Product:

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

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

LIMITED WARRANTY

Renesas warrants its emulator products to be manufactured in accordance with published specifications and free from defects in material and/or workmanship. Renesas, at its option, will replace any emulator products returned intact to the factory, transportation charges prepaid, which Renesas, upon inspection, shall determine to be defective in material and/or workmanship. The foregoing shall constitute the sole remedy for any breach of Renesas' warranty. See the Renesas warranty booklet for details on the warranty period. This warranty extends only to you, the original Purchaser. It is not transferable to anyone who subsequently purchases the emulator product from you. Renesas is not liable for any claim made by a third party or made by you for a third party.

DISCLAIMER

RENESAS MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, ORAL OR WRITTEN, EXCEPT AS PROVIDED HEREIN, INCLUDING WITHOUT LIMITATION THEREOF, WARRANTIES AS TO MARKETABILITY, MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE OR USE, OR AGAINST INFRINGEMENT OF ANY PATENT. IN NO EVENT SHALL RENESAS BE LIABLE FOR ANY DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE, OR LOSSES OR EXPENSES RESULTING FROM ANY DEFECTIVE EMULATOR PRODUCT, THE USE OF ANY EMULATOR PRODUCT, OR ITS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCEPT AS EXPRESSLY STATED OTHERWISE IN THIS WARRANTY, THIS EMULATOR PRODUCT IS SOLD "AS IS", AND YOU MUST ASSUME ALL RISK FOR THE USE AND RESULTS OBTAINED FROM THE EMULATOR PRODUCT.

State Law:

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

The Warranty is Void in the Following Cases:

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

All Rights Reserved:

This user's manual and emulator product are copyrighted and all rights are reserved by Renesas. No part of this user's manual, all or part, may be reproduced or duplicated in any form, in hard-copy or machine-readable form, by any means available without Renesas' prior written consent.

Other Important Things to Keep in Mind:

1. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Renesas' semiconductor products. Renesas assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.
2. No license is granted by implication or otherwise under any patents or other rights of any third party or Renesas.

Figures:

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

Device names:

Sections 1 to 6 of the Debugger Part in this user's manual use SHxxxx as an example of the device names.

Limited Anticipation of Danger:

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

SAFETY PAGE

READ FIRST

- **READ** this user's manual before using this emulator product.
- **KEEP** the user's manual handy for future reference.

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

DEFINITION OF SIGNAL WORDS



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



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



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



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



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

NOTE emphasizes essential information.

WARNING

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

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

Warnings on Emulator Usage

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

WARNING

Always switch OFF the host computer and user system before connecting or disconnecting any CABLES or PARTS. Failure to do so will result in a FIRE HAZARD and will damage the user system and the emulator product or will result in PERSONAL INJURY. The USER PROGRAM will be LOST.

CAUTION

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

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

Introduction

The High-performance Embedded Workshop (HEW) is a powerful development environment for embedded applications targeted at Renesas microcontrollers. The main features are:

- A configurable build engine that allows you to set-up compiler, assembler and linker options via an easy to use interface.
- An integrated text editor with user customizable syntax coloring to improve code readability.
- A configurable environment to run your own tools.
- An integrated debugger which allows you to build and debug in the same application.
- Version control support.

The High-performance Embedded Workshop has been designed with two key aims; firstly to provide you, the user, with a set of powerful development tools and, secondly, to unify and present them in a way that is easy to use.

About This Manual

This manual is comprised of three parts: HEW part, Debugger Part, and appendix.

HEW Part: Information on the basic “look and feel” of the HEW and customizing the HEW environment, and detail of the HEW’s build function.

Debugger Part: Emulator Functions, Preparation before Use, Preparations for Debugging, Debugging, and Tutorial.

Appendix: Troubleshooting, Regular Expressions, Placeholders, I/O File Format, Symbol File Format, Menus, Command-Line Functions, Notes on HEW, and Diagnostic Test Procedure.

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

Microsoft® and Windows® are registered trademarks of Microsoft Corporation.

Visual SourceSafe is a trademark of Microsoft Corporation.

IBM is a registered trademark of International Business Machines Corporation.

All brand or product names used in this manual are trademarks or registered trademarks of their respective companies or organizations.

Document Conventions

This manual uses the following typographic conventions:

Table 1 Typographic Conventions

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

Contents

HEW Part	1
1. Overview.....	3
1.1 Workspaces, Projects and Files.....	3
1.2 The Main Window	4
1.2.1 The Title Bar.....	5
1.2.2 The Menu Bar	5
1.2.3 The Toolbars	5
1.2.4 The Workspace Window.....	8
1.2.5 The Editor Window.....	12
1.2.6 The Output Window	13
1.2.7 The Status Bar.....	14
1.3 The Help System.....	15
1.4 Launching the HEW.....	16
1.5 Exiting the HEW.....	16
1.6 Component System Overview.....	17
2. Build Basics	19
2.1 The Build Process	19
2.2 Project Files	21
2.2.1 Adding Files to a Project.....	22
2.2.2 Removing Files from a Project	24
2.2.3 Excluding a Project File from Build	27
2.2.4 Including a Project File in Build	27
2.3 File Extensions and File Groups	27
2.4 Specifying How to Build a File.....	36
2.5 Build Configurations.....	37
2.5.1 Selecting a Configuration.....	38
2.5.2 Adding and Deleting Configurations	39
2.6 Building a Project	40
2.6.1 Building a Project	40
2.6.2 Building Individual Files	40
2.6.3 Stopping a Build	41
2.6.4 Building Multiple Projects.....	41
2.6.5 The Output Window	42
2.6.6 Controlling the Content of the Output Window.....	42
2.7 File Dependencies.....	43
2.8 Configuring the Workspace Window.....	44
2.8.1 Show Dependencies under Each File.....	44

2.8.2	Show Standard Library Includes	45
2.8.3	Show File Paths	46
2.9	Setting the Current Project	46
2.10	Inserting a Project into a Workspace.....	46
2.11	Specifying Dependencies between Projects	48
2.12	Removing a Project from a Workspace.....	50
2.13	Loading/Unloading a Project to/from a Workspace.....	50
2.14	Relative Projects Paths in the Workspace	51
3.	Advanced Build Features.....	53
3.1	The Build Process Revisited	53
3.1.1	What is a Build?.....	53
3.2	Creating a Custom Build Phase.....	56
3.3	Ordering Build Phases	61
3.3.1	Build Phase Order.....	63
3.3.2	Build File Phase Order.....	67
3.4	Setting Custom Build Phase Options	68
3.4.1	Options Tab.....	69
3.4.2	Output Files Tab	69
3.4.3	Dependent Files Tab	72
3.5	File Mappings	74
3.6	Controlling the Build	76
3.7	Logging Build Output	77
3.8	Changing Toolchain Version	78
3.9	Using an External Debugger.....	79
3.10	Generating a Makefile.....	81
4.	Using the Editor.....	83
4.1	The Editor Window.....	83
4.2	Working with Multiple Files.....	85
4.2.1	The Editor Toolbars	85
4.2.2	Editor Toolbar Buttons	85
4.2.3	Search Toolbar Buttons.....	88
4.2.4	Bookmarks Toolbar Buttons	88
4.2.5	Templates Toolbar Buttons.....	88
4.3	Standard File Operations.....	90
4.3.1	Creating a New File	90
4.3.2	Saving a File	90
4.3.3	Saving all Files.....	91
4.3.4	Opening a File.....	91
4.3.5	Closing Files	92
4.4	Editing a File.....	93
4.5	Searching and Navigating through Files	94

High-performance Embedded Workshop User's Manual

4.5.1	Finding Text.....	94
4.5.2	Finding Text in Multiple Files	96
4.5.3	Replacing Text.....	97
4.5.4	Jumping to a Specified Line.....	98
4.6	Bookmarks.....	98
4.7	Printing a File.....	100
4.8	Configuring Text Layout	101
4.8.1	Page Set-up.....	101
4.8.2	Changing Tabs	103
4.8.3	Auto Indentation	104
4.9	Splitting a Window	105
4.10	Configuring Text.....	106
4.10.1	Changing the Editor Font.....	106
4.11	Syntax Coloring	107
4.12	Templates.....	111
4.12.1	Defining a Template	111
4.12.2	Deleting a Template	112
4.12.3	Inserting a Template	112
4.12.4	Brace Matching.....	113
4.13	Editor Column management	114
5.	Tools Administration	117
5.1	Tool Locations	119
5.2	HEW Registration Files (*.HRF).....	119
5.3	Registering Components	121
5.3.1	Searching Drives for Components	121
5.3.2	Registering a Single Component.....	123
5.4	Unregistering Components.....	123
5.5	Viewing and Editing Component Properties.....	124
5.6	Uninstalling Components.....	127
5.7	Technical Support Issues	130
5.8	Custom Project Types	132
6.	Customizing the Environment	133
6.1	Customizing the Toolbar.....	133
6.2	Customizing the Tools Menu.....	137
6.3	Configuring the Help System.....	139
6.4	Specifying Workspace Options.....	141
6.4.1	Open last workspace at start-up	142
6.4.2	Restore the files on opening workspace.....	142
6.4.3	Display workspace information dialog on opening workspace.....	143
6.4.4	Save workspace before executing any tools.....	144
6.4.5	Prompt before saving workspace	144

6.4.6	Default directory for new workspaces	144
6.4.7	Prompt before saving session.....	144
6.5	Using an External Editor.....	145
6.6	Customizing File Save	147
6.6.1	Save files before executing any tools.....	148
6.6.2	Prompt before saving files	148
6.7	Using an External Debugger.....	149
6.8	Using Custom Placeholders	151
6.9	Using Confirmation Dialog Boxes.....	152
7.	Version Control	155
7.1	Selecting a Version Control System.....	156
8.	Using the Custom Version Control System.....	159
8.1.1	System menu options and toolbar buttons	161
8.1.2	User menu options	164
8.2	Defining Version Control Commands.....	166
8.2.1	Executable return code.....	167
8.3	Specifying Arguments.....	168
8.3.1	Specifying File Locations	169
8.3.2	Specifying Environment	173
8.3.3	Specifying Comments.....	175
8.3.4	Specifying a User Name and Password	176
8.4	Controlling Execution.....	178
8.4.1	Prompt before executing command	178
8.4.2	Run in DOS Window	178
8.4.3	Use forward slash '/' as version control directory delimiter	178
8.5	Importing and exporting a Set-up	179
9.	Using Visual SourceSafe	181
9.1	Attaching Visual SourceSafe to a Workspace.....	181
9.1.1	Selecting Visual SourceSafe	181
9.1.2	Adding files to Visual SourceSafe	182
9.2	Visual SourceSafe commands.....	184
9.2.1	Removing a File from Version Control	184
9.2.2	Getting a Read Only Copy of a File from Version Control	184
9.2.3	Checking Out a Writable Copy of a File from Version Control	185
9.2.4	Checking In a Writable Copy of a File into Version Control.....	185
9.2.5	Undoing a Check Out Operation.....	185
9.2.6	Viewing the Status of a File.....	186
9.2.7	Viewing the History of a File.....	186
9.3	Visual SourceSafe Integration Options	187

Debugger Part	189
Section 1 Overview.....	191
1.1 Warnings.....	194
1.2 Environmental Conditions	195
1.3 Components	197
Section 2 E10A Emulator Functions.....	199
2.1 Overview.....	199
2.2 Trace Functions	201
2.2.1 Internal Trace Function	201
2.2.2 AUD Trace Function.....	201
2.3 Break Function.....	204
2.4 Performance Measurement Function	205
2.4.1 Function for Measuring the Number of Cycles from Point to Point	205
2.4.2 Profiling Function	205
2.5 Memory Access Functions.....	206
2.6 Stack Trace Function	208
2.7 Online Help.....	208
Section 3 Preparation before Use.....	209
3.1 Emulator Preparation	209
3.2 Installing Emulator's Software	210
3.2.1 Installing under Windows [®] 98 or Windows [®] Me Operating System.....	210
3.2.2 Installing under Windows NT [®] 4.0 Operating System	211
3.2.3 Installing under Windows [®] 2000 or Windows [®] XP Operating System.....	212
3.3 Connecting the Card Emulator to the Host Computer	213
3.4 Connecting the Card Emulator to the User System	214
3.5 System Check	217
3.6 Uninstalling the Emulator's Software.....	227
Section 4 Preparations for Debugging.....	233
4.1 Workspaces, Projects, and Files.....	233
4.2 Method for Activating HEW.....	234
4.2.1 Creating the New Workspace (Toolchain Not Used).....	235
4.2.2 Creating the New Workspace (Toolchain Used).....	239
4.2.3 Selecting an Existing Workspace.....	244
4.3 Setting at Emulator Activation.....	246
4.3.1 Setting at Emulator Activation	246
4.3.2 Downloading a Program	248
4.3.3 Setting the Writing Flash Memory Mode.....	249
4.4 Debugger Sessions	256
4.4.1 Selecting a Session.....	256

- 4.4.2 Adding and Deleting Sessions 257
- 4.4.3 Saving Session Information 260
- 4.5 Connecting the Emulator 261
- 4.6 Ending the Emulator 262

- Section 5 Debugging 263
- 5.1 Setting the Environment for Emulation 263
 - 5.1.1 Opening the [Configuration] Dialog Box 263
 - 5.1.2 [General] Page 263
 - 5.1.3 Downloading to the Flash Memory..... 266
- 5.2 Downloading a Program 268
 - 5.2.1 Downloading a Program 268
 - 5.2.2 Viewing the Source Code 269
 - 5.2.2 Viewing the Assembly-Language Code..... 272
 - 5.2.3 Modifying the Assembly-Language Code 273
 - 5.2.4 Viewing a Specific Address 274
 - 5.2.5 Viewing the Current Program Counter Address 274
- 5.3 Debugging with the Command Line Interface 275
 - 5.3.1 Opening the [Command Line] Window 275
 - 5.3.2 Specifying a Command File..... 276
 - 5.3.3 Executing a Command File..... 276
 - 5.3.4 Stopping Command Execution 276
 - 5.3.5 Specifying a Log File..... 276
 - 5.3.6 Starting or Stopping Logging..... 277
 - 5.3.7 Entering a Full Path to the File 277
 - 5.3.8 Pasting a Placeholder 277
- 5.4 Looking at Registers 278
 - 5.4.1 Opening the [Register] Window 278
 - 5.4.2 Expanding a Bit Register 279
 - 5.4.3 Modifying Register Contents..... 279
 - 5.4.4 Using Register Contents 280
- 5.5 Operating Memory 281
 - 5.5.1 Viewing a Memory Area..... 281
 - 5.5.2 Displaying Data in Different Formats..... 282
 - 5.5.3 Splitting Up the Window Display 283
 - 5.5.4 Viewing a Different Memory Area..... 283
 - 5.5.5 Modifying the Memory Contents..... 284
 - 5.5.6 Selecting a Memory Range..... 284
 - 5.5.7 Finding a Value in Memory 285
 - 5.5.8 Filling a Memory Area with a Value 286
 - 5.5.9 Copying a Memory Area 287
 - 5.5.10 Saving and Verifying a Memory Area..... 288
 - 5.5.11 Disabling Update of the Window Contents 289

5.5.12	Updating the Window Contents	289
5.5.13	Comparing the Memory Contents	289
5.5.14	Loading a File to a Memory Area	291
5.6	Viewing the I/O Memory	292
5.6.1	Opening the [IO] Window	292
5.6.2	Expanding the I/O Register Display	293
5.6.3	Modifying the I/O Register Contents	293
5.7	Viewing the Current Status	294
5.8	Looking at Labels	295
5.8.1	Looking at Label Lists	295
5.8.2	Adding a Label	296
5.8.3	Editing a Label	296
5.8.4	Deleting a Label	297
5.8.5	Deleting All Labels	297
5.8.6	Loading Labels from a File	298
5.8.7	Saving Labels into a File	298
5.8.8	Searching for a Label	299
5.8.9	Searching for the Next Label	299
5.8.10	Viewing the Source Corresponding to a Label	299
5.9	Executing Your Program	300
5.9.1	Running from Reset	300
5.9.2	Continuing Run	300
5.9.3	Running to the Cursor	301
5.9.4	Running from a Specified Address	302
5.9.5	Single Step	303
5.9.6	Multiple Steps	304
5.10	Stopping Your Program	305
5.10.1	Stopping the Program by the [STOP] Toolbar Button	305
5.10.2	Standard Breakpoints (PC Breakpoints)	305
5.11	Elf/Dwarf2 Support	307
5.11.1	C/C++ Operators	307
5.11.2	C/C++ Expressions	307
5.11.3	Supporting Duplicate Labels	308
5.11.4	Debugging an Overlay Program	309
5.12	Looking at Variables	312
5.12.1	Tooltip Watch	312
5.12.2	Instant Watch	312
5.12.3	[Watch] Window	313
5.12.4	[Locals] Window	320
5.13	Using the Event Points	321
5.13.1	PC Breakpoints	321
5.13.2	Break Conditions	321
5.13.3	Opening the [Event] Window	321

5.13.4	Setting PC Breakpoints	322
5.13.5	Add	322
5.13.6	Edit.....	323
5.13.7	Enable	323
5.13.8	Disable	323
5.13.9	Delete.....	323
5.13.10	Delete All.....	323
5.13.11	Go to Source	323
5.13.12	[Breakpoint] Dialog Box.....	324
5.13.13	Setting Break Conditions	325
5.13.14	Edit.....	326
5.13.15	Enable	326
5.13.16	Disable	326
5.13.17	Delete.....	326
5.13.18	Delete All.....	326
5.13.19	Go to Source	326
5.13.20	Sequential Conditions.....	326
5.13.21	Editing Break Conditions.....	327
5.13.22	Modifying Break Conditions	327
5.13.23	Enabling Break Conditions	327
5.13.24	Disabling Break Conditions.....	327
5.13.25	Deleting Break Conditions.....	327
5.13.26	Deleting All Break Conditions.....	327
5.13.27	Viewing the Source Line for Break Conditions.....	327
5.14	Viewing the Trace Information	328
5.14.1	Opening the [Trace] Window	328
5.14.2	Acquiring Trace Information	328
5.14.3	Specifying Trace Acquisition Conditions	331
5.14.4	Searching for a Trace Record.....	337
5.14.5	Clearing the Trace Information.....	344
5.14.6	Saving the Trace Information in a File	344
5.14.7	Viewing the [Editor] Window.....	344
5.14.8	Trimming the Source	344
5.14.9	Temporarily Stopping Trace Acquisition.....	345
5.14.10	Extracting Records from the Acquired Information.....	345
5.14.11	Analyzing Statistical Information	353
5.14.12	Extracting Function Calls from the Acquired Trace Information	355
5.15	Viewing the Function Call History	356
5.15.1	Opening the [Stack Trace] Window.....	356
5.15.2	Viewing the Source Program.....	356
5.15.3	Specifying the View.....	357
5.16	Displaying Memory Contents as an Image	358
5.16.1	Opening the [Image] Window.....	358

5.16.2	Automatically Updating the Window Contents	362
5.16.3	Updating the Window Contents	362
5.16.4	Displaying the Pixel Information	363
5.17	Displaying Memory Contents as Waveforms	364
5.17.1	Opening the [Waveform View] Window	364
5.17.2	Automatically Updating the Window Contents	365
5.17.3	Updating the Window Contents	365
5.17.4	Zoom-In Display	365
5.17.5	Zoom-Out Display	366
5.17.6	Resetting the Zoom Display	366
5.17.7	Setting the Zoom Magnification	366
5.17.8	Setting the Horizontal Scale	366
5.17.9	Non-Display of Cursor	366
5.17.10	Displaying the Sampling Information	366
5.18	Analyzing Performance	368
5.18.1	Opening the [Performance Analysis] Window	368
5.18.2	Setting Conditions for Measurement	369
5.18.3	Starting Performance Data Acquisition	369
5.18.4	Deleting a Measurement Condition	369
5.18.5	Deleting All Measurement Conditions	369
5.19	Viewing the Profile Information	370
5.19.1	Stack Information Files	370
5.19.2	Profile Information Files	371
5.19.3	Loading Stack Information Files	372
5.19.4	Enabling the Profile	373
5.19.5	Specifying Measuring Mode	373
5.19.6	Executing the Program and Checking the Results	374
5.19.7	List Sheet	374
5.19.8	[Tree] Sheet	375
5.19.9	[Profile-Chart] Window	377
5.19.10	Types and Purposes of Displayed Data	378
5.19.11	Creating Profile Information Files	378
5.19.12	Notes	379
Section 6 Tutorial		381
6.1	Introduction	381
6.2	Running the HEW	382
6.3	Setting up the Emulator	382
6.4	Setting the [Configuration] Dialog Box	383
6.5	Checking the Operation of the Target Memory for Downloading	384
6.6	Downloading the Tutorial Program	386
6.6.1	Downloading the Tutorial Program	386
6.6.2	Displaying the Source Program	387

6.7	Setting a PC Breakpoint.....	388
6.8	Setting Registers	389
6.9	Executing the Program.....	391
6.10	Reviewing Breakpoints.....	394
6.11	Referring to Symbols	395
6.12	Viewing Memory.....	396
6.13	Watching Variables.....	398
6.14	Displaying Local Variables.....	401
6.15	Stepping Through a Program.....	402
	6.15.1 Executing [Step In] Command	402
	6.15.2 Executing [Step Out] Command	404
	6.15.3 Executing [Step Over] Command	406
6.16	Forced Breaking of Program Executions	408
6.17	Break Function.....	409
	6.17.1 PC Break Function	409
6.18	Hardware Break Function	414
	6.18.1 Setting the Sequential Break Condition	420
6.19	Trace Functions	425
	6.19.1 Displaying the Trace Window.....	426
	6.19.2 Internal Trace Function	427
	6.19.3 AUD Trace Function.....	429
	6.19.4 MMU Support	432
6.20	Stack Trace Function	435
6.21	Performance Measurement Function	437
	6.21.1 Performance Measurement Function.....	437
	6.21.2 Profiling Function	438
6.22	Download Function to the Flash Memory Area.....	443
6.23	What Next?	449
Appendix A Troubleshooting		451
Appendix B Regular Expressions.....		453
Appendix C Placeholders		455
C.1	What is a Placeholder?.....	455
C.2	Inserting a Placeholder.....	455
C.3	Available Placeholders.....	457
C.4	Placeholder Tips.....	459
Appendix D I/O File Format		461
D.1	File Format (Bit Field Not Supported).....	461
D.2	File Format (Bit Field Supported).....	463

Appendix E	Symbol File Format	467
Appendix F	Menus.....	469
Appendix G	Command-Line Functions.....	473
Appendix H	Notes on HEW	475

HEW Part

1. Overview

This chapter describes the fundamental concepts of the High-performance Embedded Workshop. It is intended to give users who are new to Windows® extra help, filling in the details that are required by later chapters.

1.1 Workspaces, Projects and Files

Just as a word processor allows you to create and modify documents, the High-performance Embedded Workshop 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. Figure 1.1 illustrates this graphically.

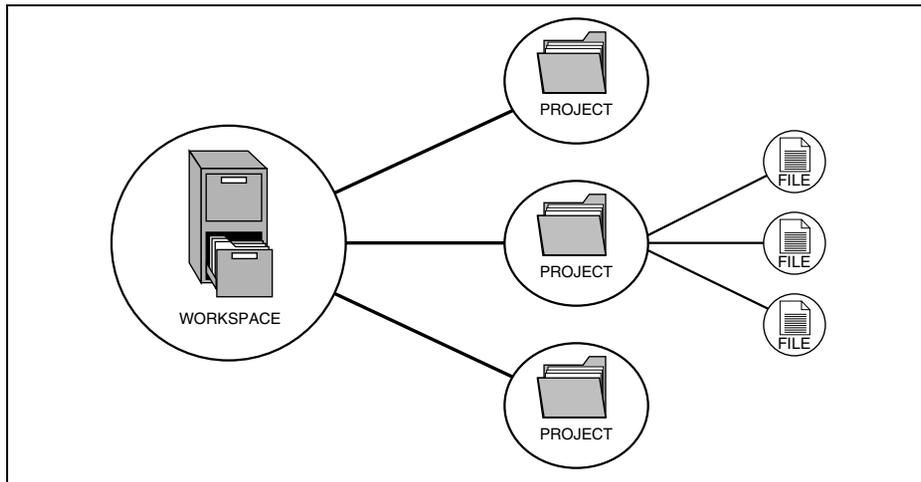


Figure 1.1: Workspaces, Projects and Files

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.

1.2 The Main Window

The HEW main window appears as shown in figure 1.2.

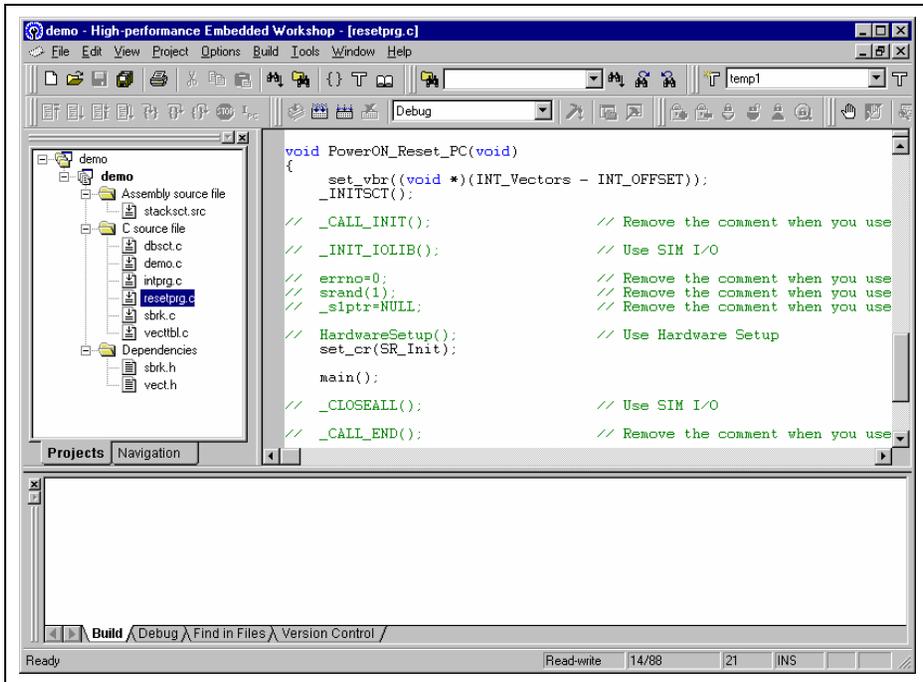


Figure 1.2: HEW Main Window

There are three main windows; the workspace window, the editor window and the output window. The workspace window shows the projects and files which are currently in the workspace, the editor window provides file viewing and editing facilities and the output window shows the results of a various processes (e.g. build, version control commands and so on).

1.2.1 The Title Bar

The title bar displays the name of the currently open workspace, project and file. It also contains the standard minimize, maximize and close buttons. Click the minimize button to minimize the HEW on the windows start bar. Click the maximize button to force HEW to fill the screen. Click the close button to close the HEW (this has the same effect as selecting **[File->Exit]** or pressing **ALT+F4**).

1.2.2 The Menu Bar

The menu bar contains nine menus: File, Edit, View, Project, Options, Build, Tools, Window and Help. All of the menu options are grouped logically under these headings. For instance, if you wanted to open a file then the file menu is where you will find the right menu option, if you wanted to set-up a tool then the tools menu is the correct selection. The following sections will cover the functions of the various menu options, as they become relevant. However, at this stage, it is worth taking a few moments to familiarize yourself with the options that each menu provides.

1.2.3 The Toolbars

The toolbars provide a shortcut to the options, which you will use the most often. There are eight default toolbars: Bookmarks, Debug, Debug Run, Editor, Search, Standard, Templates, and Version Control (as shown in figure 1.3 to 1.10). Toolbars can be created, modified and removed via the **[Tools->Customize...]** menu option (see chapter 6, “*Customizing the Environment*”, for further information).

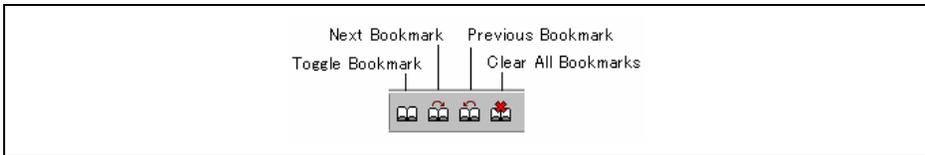


Figure 1.3: Bookmarks Toolbar

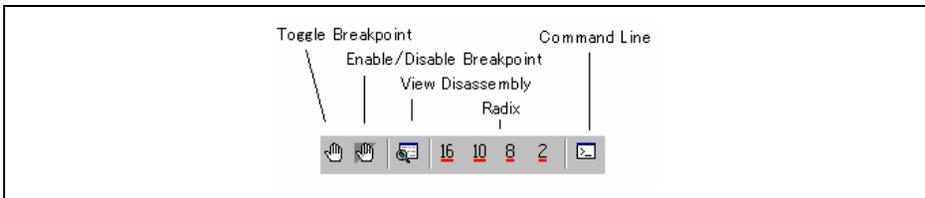


Figure 1.4: Debug Toolbar

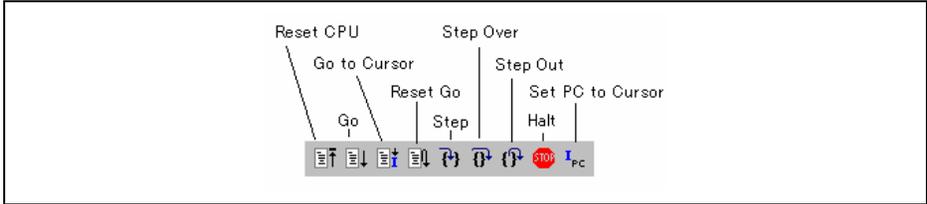


Figure 1.5: Debug Run Toolbar

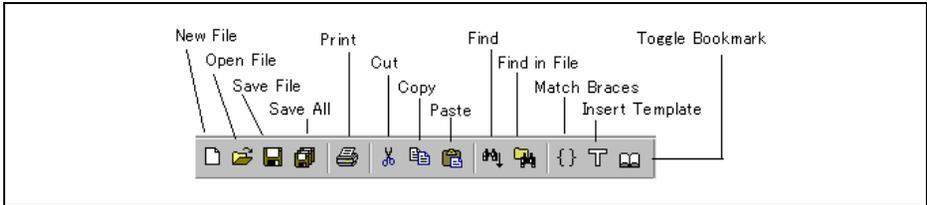


Figure 1.6: Editor Toolbar

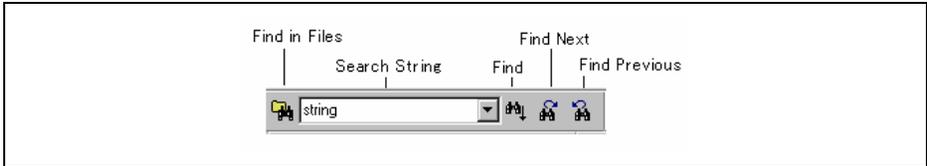


Figure 1.7: Search Toolbar

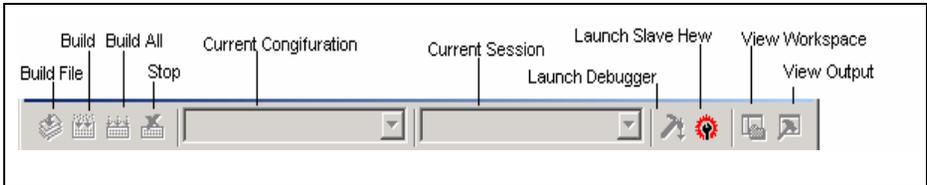


Figure 1.8: Standard Toolbar

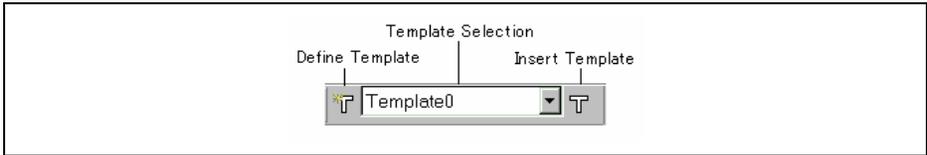


Figure 1.9: Templates Toolbar

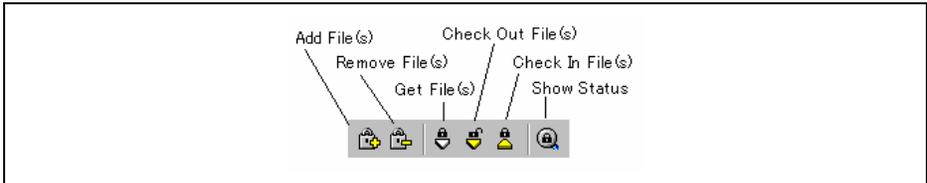


Figure 1.10: Version Control Toolbar

When the Standard toolbar or a toolbar is docked, it has a control bar as shown in figure 1.11 (i). If you want to move the docked Standard toolbar, click and drag its control bar to the new location. Figure 1.11 (i) shows the Standard toolbar when it is docked and figure 1.11 (ii) shows the Standard toolbar when it is floating.

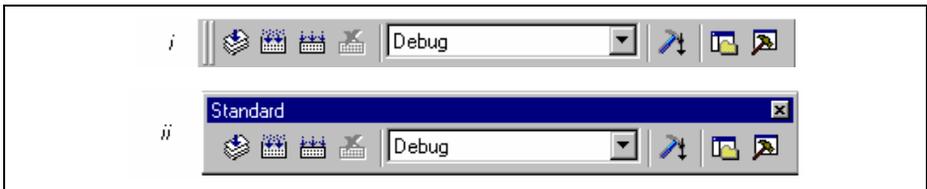


Figure 1.11: Standard Toolbar, Docked and Floating

- To dock the menu bar or a toolbar:
 1. Double-click on the title bar of a floating menu bar or toolbar.
 or:
 2. Drag the title bar of a floating menu bar or toolbar and draw it toward an edge of a docked window, menu bar, toolbar or the HEW main frame, on whose edge you would like to dock the window, until the shape of the floating bar changes.

- ☞ To float the menu bar or a toolbar:
 1. Double-click on the control bar of a docked menu bar or toolbar.or:
 2. Drag the control bar of a docked menu bar or toolbar and draw it away from the edge of the HEW main frame and from an edge of the other docked windows, menu bar or toolbar.

1.2.4 The Workspace Window

The “Workspace” window when the HEW is launched only has a single pane. This is the “Projects” tab. If a workspace is opened then the workspace window displays two default tabs. The “Projects” tab shows the current workspace, projects and files (figure 1.12). You can quickly open any project file or dependent file by double clicking on its corresponding icon.

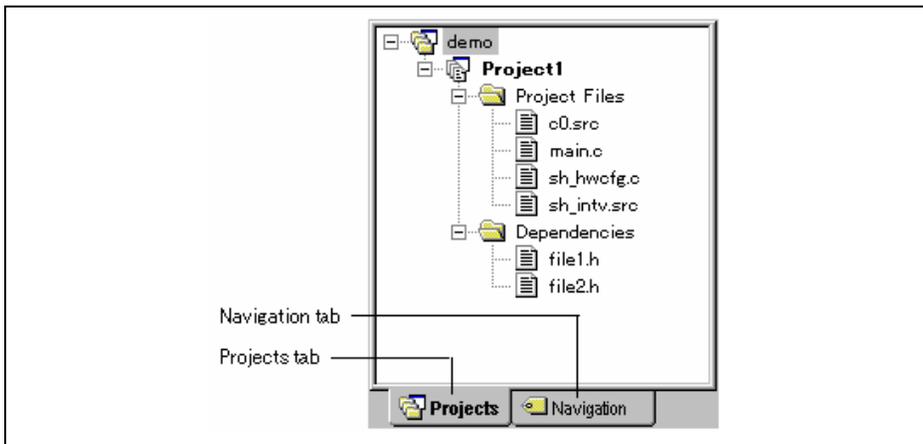


Figure 1.12: Workspace Window Projects Tab

The “Navigation” tab provides jumps to various textual constructs within your project’s files. What is actually displayed within the navigation tab depends upon what components are currently installed. Figure 1.13 shows ANSI C functions. See chapter 2, “Build Basics”, for more information on the “Workspace” window.

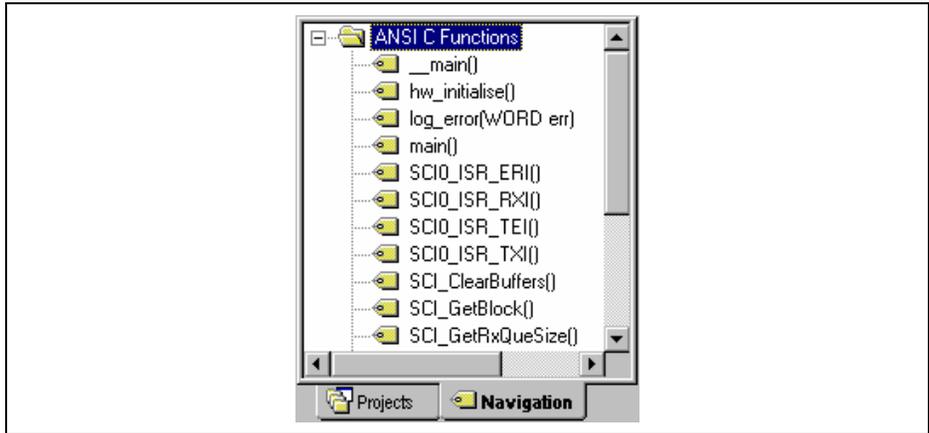


Figure 1.13: Workspace Window Navigation Tab

- To allow the “Workspace” window or the “Output” window docking:

Click the right mouse button anywhere inside the “Workspace” window or the “Output” window. Then a pop-up menu will be displayed. If [**Allow Docking**] is checked, docking is allowed; otherwise, docking is not allowed. Select [**Allow Docking**] to check or uncheck it.

When [**Allow Docking**] is checked, you can dock a window, a toolbar or a menu bar to the edge of the HEW main window or to the edge of another docked window. Also if [**Allow Docking**] is checked, you can float them “above” the other HEW windows or outside the HEW main window. Figure 1.14 (i) shows a docked “Workspace” window, and figure 1.14 (ii) shows a floating “Workspace” window.

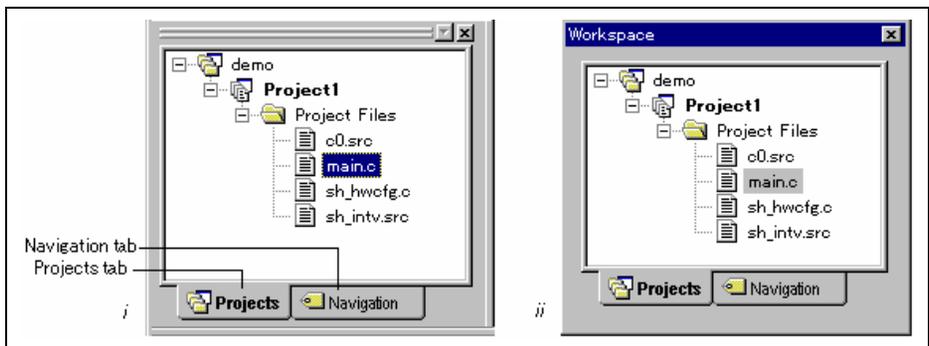


Figure 1.14: Workspace Window, Docked and Floating

When the “Workspace” window or the “Output” window is docked, it has a control bar as shown in figure 1.15. If you want to move a docked window, click and drag its control bar to the new location.



Figure 1.15: Control Bar of Docking Window

- To dock the “Workspace” window or the “Output” window:
[**Allow Docking**] must be checked on the pop-up menu of the window to dock the “Workspace” window or the “Output” window. (The pop-up menu will be displayed when you click the right mouse button anywhere inside the window.) Then you have two ways to dock the window.
 1. Double-click on the control bar of a floating window.or:
 2. Drag the title bar of a floating window and draw it toward an edge of a docked window, menu bar or toolbar, or the HEW main frame, on whose edge you would like to dock the window, until the shape of the floating window changes.

- To float the “Workspace” window or the “Output” window:
[**Allow Docking**] must be checked on the pop-up menu of the window to float the “Workspace” window or the “Output” window. (The pop-up menu will be displayed when you click the right mouse button anywhere inside the window.) Then you have two ways to float the window.
 1. Double-click on the control bar of a docking window.or:
 2. Drag the control bar of a docked window and draw it away from the edge of the HEW main frame and from an edge of the other docked windows, menu bar or toolbar.
- To hide the “Workspace” window or the “Output” window:
Click on the close button, which is located in the top right corner of the window. Or push the right mouse button anywhere inside a floating window and select [**Hide**] on the pop-up menu.
- To display the “Workspace” window or the “Output” window:
Select [**View->Workspace**] or [**View->Output**], respectively.

1.2.5 The Editor Window

The editor window is where you will work with the files of your project. The HEW allows you to have many files open at one time, to switch between them, to arrange them and to edit them in whichever order you want to. By default, the editor window is displayed in a notebook style, where each text file has a separate tab (as shown in figure 1.16).

The editor contains a gutter on the left-hand side of the window. The gutter in HEW can be configured to contain many columns. Each column can refer to a different component's capability. In figure 1.16 the editor is displayed with the debugger address column and the standard column. The standard column allows the user to configure the position of bookmarks and software breakpoints quickly and easily.

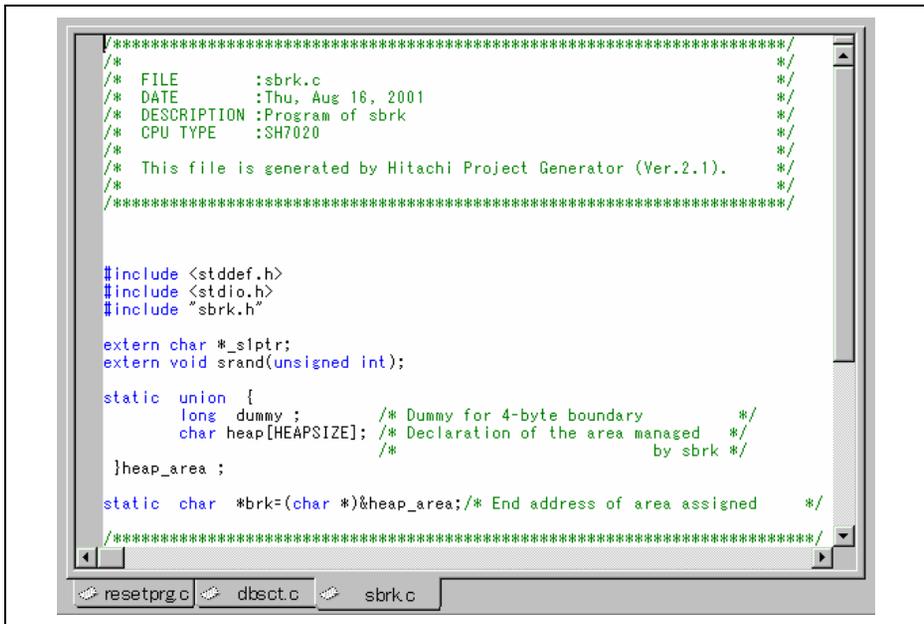


Figure 1.16: Editor Window

The editor window can be customized via the “Format Views” dialog box, which can be invoked via the [Tools->Format Views...] menu option. This dialog allows you to configure fonts, colors, tabs and so on for the editor window. It also allows the user to change the look of other views, which have been installed by HEW. If you would prefer to use your favorite editor rather than the HEW internal editor then specify your alternative in the “Options” dialog box, which can be

invoked via the [**Tools->Option...**] menu option. For further details on how to use and configure the editor, refer to chapter 4, “*Using the Editor*”.

1.2.6 The Output Window

The “Output” window by default has four tabs on display. The “Build” tab shows the output from any build process (e.g. compiler, assembler and so on). If an error is encountered in a source file then the error will be displayed in the build tab along with the source file name and line number. To quickly locate a problem, double click on the error to jump to the source file and line.

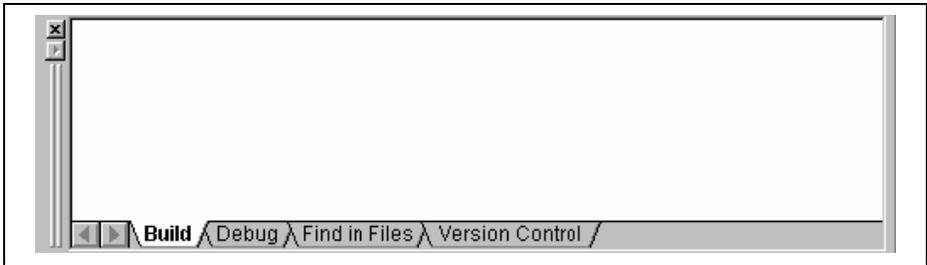


Figure 1.17: Output Window

The “Debug” tab shows the output from any debugger process. Any debug component that needs to display information will send its output to this window.

The “Find in Files” tab displays the results of the last “Find in Files” action. To activate find in files, select the [**Edit->Find in Files...**] menu option, the toolbar button. For further details on how to use find in files, refer to chapter 4, “*Using the Editor*”.

The “Version Control” tab displays the results of version control actions. The tab is only displayed if a version control system is in use. For further details on version control, refer to chapter 7, “*Version Control*”.

1.2.7 The Status Bar

The status bar displays information as to the current state of the HEW. Figure 1.18 shows the seven sections of the status bar.

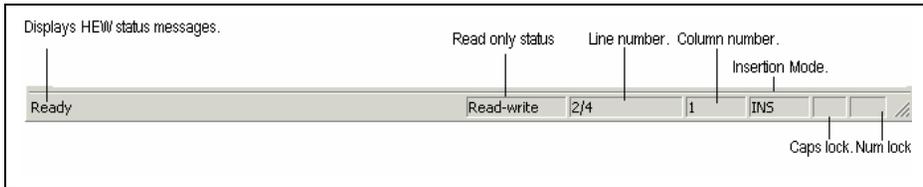


Figure 1.18: Status Bar

1.3 The Help System

The help menu is the rightmost menu on the HEW menu bar. It contains the menu option “Contents” which, when selected, takes you to the main HEW help window.

To obtain help on specific dialogs click on the context sensitive help button, which is located in the top right-hand corner of each dialog box (as shown in figure 1.19).

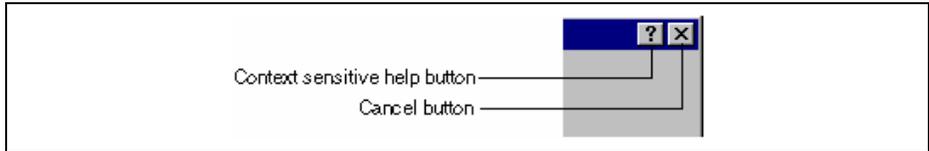


Figure 1.19: Help Button

When this is clicked, the mouse pointer will change to a pointer with a question mark above it. Whilst the mouse pointer is in this state, click on the part of the dialog box that you require assistance on.

Alternatively, select the control that you require help for and then press the **F1** key.

1.4 Launching the HEW

To run the HEW, open the “Start” menu of Windows®, select “Programs”, select “High-performance Embedded Workshop” and then select the shortcut of the “High-performance Embedded Workshop 2”. By default, the “Welcome!” dialog box shown in figure 1.20 will be displayed.

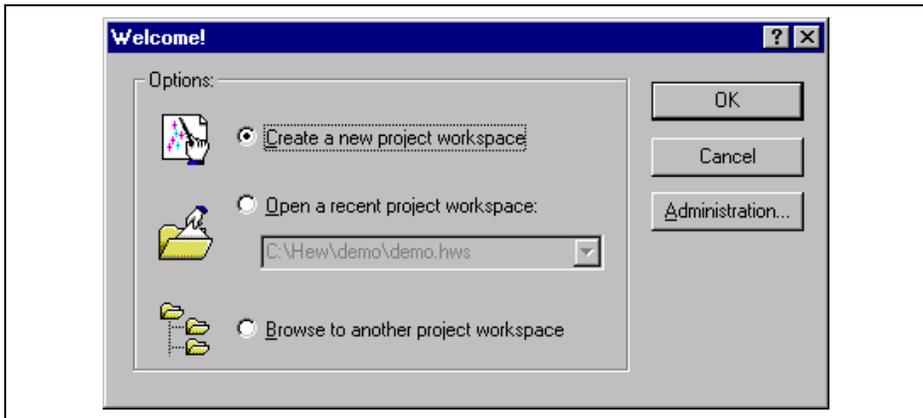


Figure 1.20: Welcome! Dialog

To create a new workspace, select “Create a new project workspace”, and click “OK”. To open one of recent project workspaces, select “Open a recent project workspace”, select a workspace from the drop-down list, and click “OK”. The recent project workspace list displays the same information as that seen in the workspace most recently used file list. This list appears on the file menu. To open a workspace by specifying a workspace file (.HWS file), select “Browse to another project workspace”, and click “OK”. To register a tool to or unregister a tool from the HEW, click the “Administration...” button (see chapter 5, “*Tool Administration*” for details). Click the “Cancel” button to use the HEW without opening a workspace.

1.5 Exiting the HEW

The HEW can be exited by selecting **[File->Exit]**, pressing **ALT+F4** or by selecting the close option from the system menu. (To open the system menu, click the icon at the upper-left corner of the HEW title bar.) If a workspace is open then the same workspace closedown procedure is followed as described in the previous section.

1.6 Component System Overview

The HEW allows the user to extend the HEW functionality by adding additional components to the system. This is achieved by registering the component in the Tools Administration dialog box. These components can add windows, menus and toolbars to the HEW system. Examples of the components are the debugger and builder components of HEW. The debugger component adds all of the menus and toolbars associated with the debugger and the builder component does the same for the build functionality. The components you have registered in the system will modify the look and feel of HEW. In some cases you may not have some of the menus which you can see in this manual. For instance if the debugger component is not installed you will not have the “Debug” menu in the HEW main window.

2. Build Basics

This chapter explains the general functions of the HEW whilst the more advanced features can be found in chapter 3, “*Advanced Build Features*”.

2.1 The Build Process

The typical build process is outlined in figure 2.1. This may not be the exact build process, which your installation of HEW will use as it depends upon the tools that were provided with your installation of HEW (e.g. you may not have a compiler for instance). In any case, the principles are the same - each step or phase of the build takes a set of project files and then builds them, if all succeeds then the next step or phase is executed.

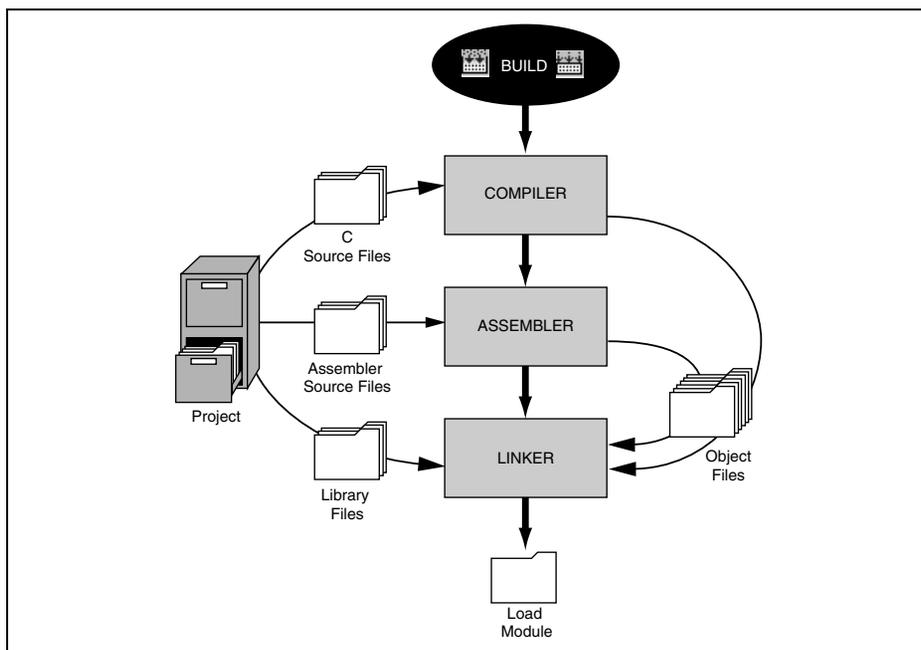


Figure 2.1: Typical Build Process

In the example shown in figure 2.1 the compiler is the first phase, the assembler is the second phase and the linker is the third and final phase. During the compiler phase, the C source files from the project are compiled in turn, during the assembler phase, the assembler source files are

assembled in turn. During the linker phase all library files and output files from the compiler and assembler phases are linked together to produce the load module. This module can then be downloaded and used by the debugger functionality in HEW.

The build process can be customized in several ways. For instance, you can add your own phase, disable a phase, delete phases and so forth. These advanced build issues are left to chapter 3, “*Advanced Build Features*”. In this chapter, only the general principles and basic features will be detailed.

2.2 Project Files

In order for the HEW to be able to build your application, you must first tell it, which files should be in the project, and how each file should be built (figure 2.2).

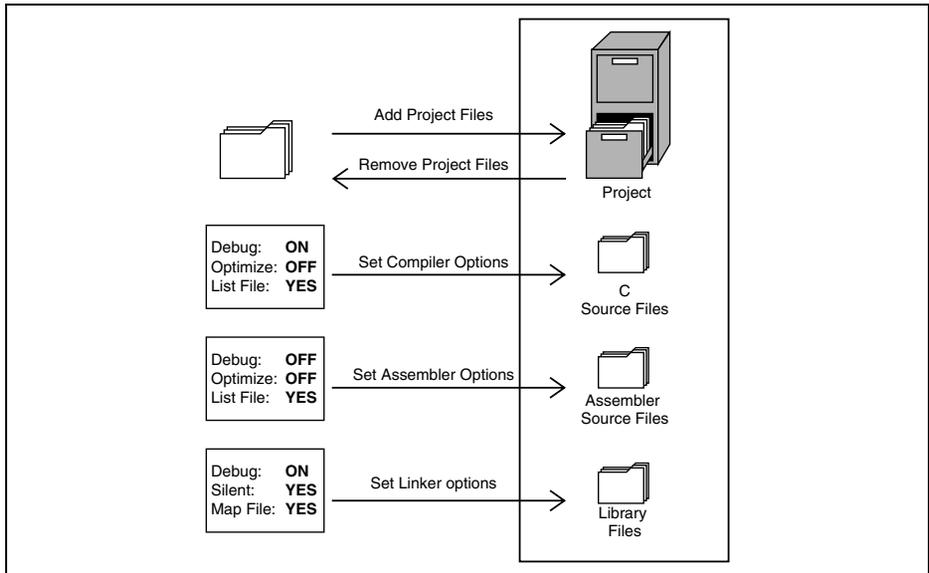


Figure 2.2: Editing a Project

2.2.1 Adding Files to a Project

Before you can build your application you must first inform the High-performance Embedded Workshop, which files it, is composed of.

☞ To add a files to a project:

1. Select [**Project->Add Files...**], select [**Add Files...**] from the “Workspace” window’s pop-up menu (see figure 2.3), or press **INS** when the “Workspace” window is selected.

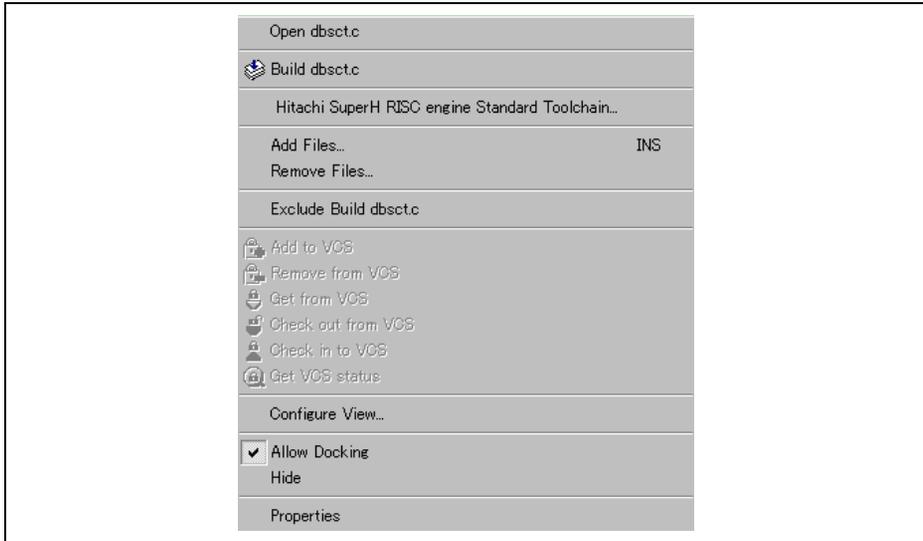


Figure 2.3: Project Pop-up Menu

2. The “Add” dialog will be displayed.
3. Select the file(s), that you want to add and then click “Add”.

There are a number of other ways to add new files to the project. These are described below:

- Clicking right button on an open file in the editor window displays a pop-up menu option (figure 2.4). If the file is already in the project then the “Add File to Project” menu option is disabled. Selecting the “Add File to Project” then adds the file to the current project.

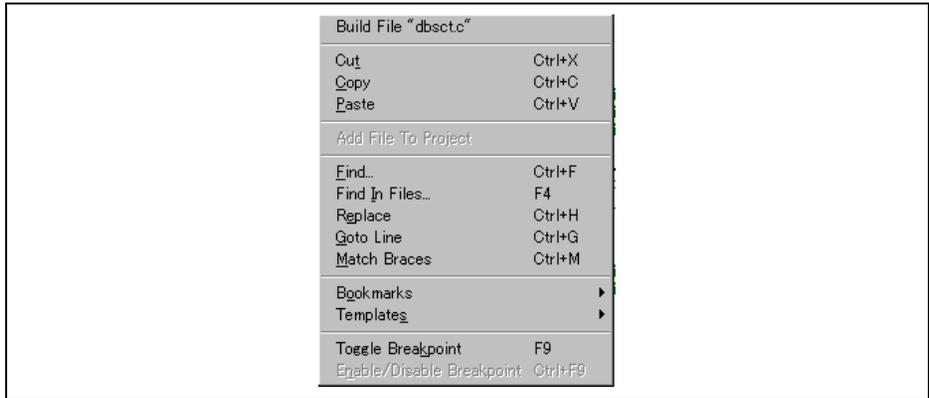


Figure 2.4: Editor Window Pop-up Menu

- In the HEW it is also possible to “Drag and Drop” files from Windows Explorer onto the workspace window. These files will be automatically added to the project and are displayed in the folder in which they were dragged to.

Note: If you add a file to a project when it is an unrecognized file type then it will still be added to the project. Certain functions will be disabled with reference to this file. When this file is double clicked in the workspace window instead of opening the file in the editor the open operation is passed to Windows operating system. The default open operation is then carried out as if the file was opened in Windows Explorer. To view the current defined extensions use the “File Extensions” dialog (see the section on file extensions later in this chapter).

2.2.2 Removing Files from a Project

Files can be individually removed from a project, selections of files can be removed or all files can be removed.

☞ To remove files from a project:

1. Select [**Project->Remove Files...**], or select [**Remove Files...**] from the “Projects” tab’s pop-up menu in the Workspace window (see figure 2.5). The “Remove Project Files” dialog will be displayed (figure 2.6).

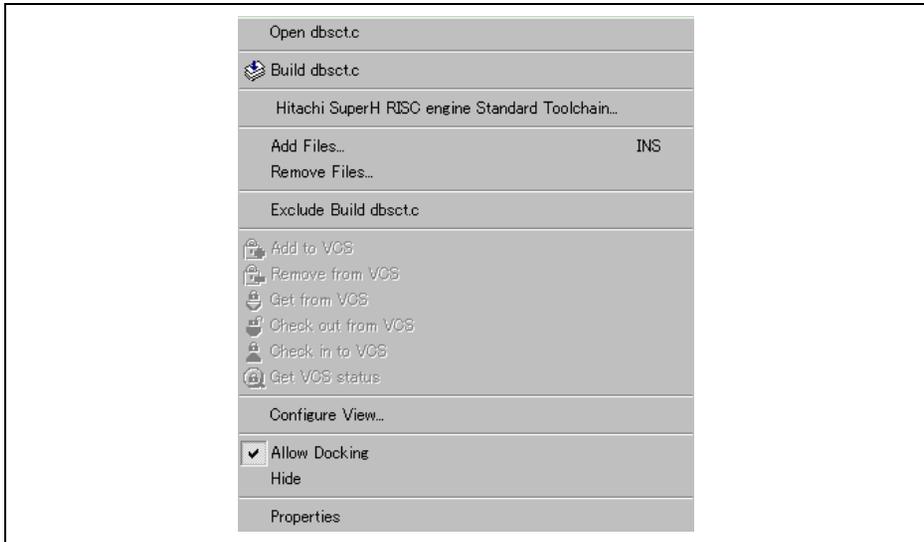


Figure 2.5: Projects Tab Pop-up Menu

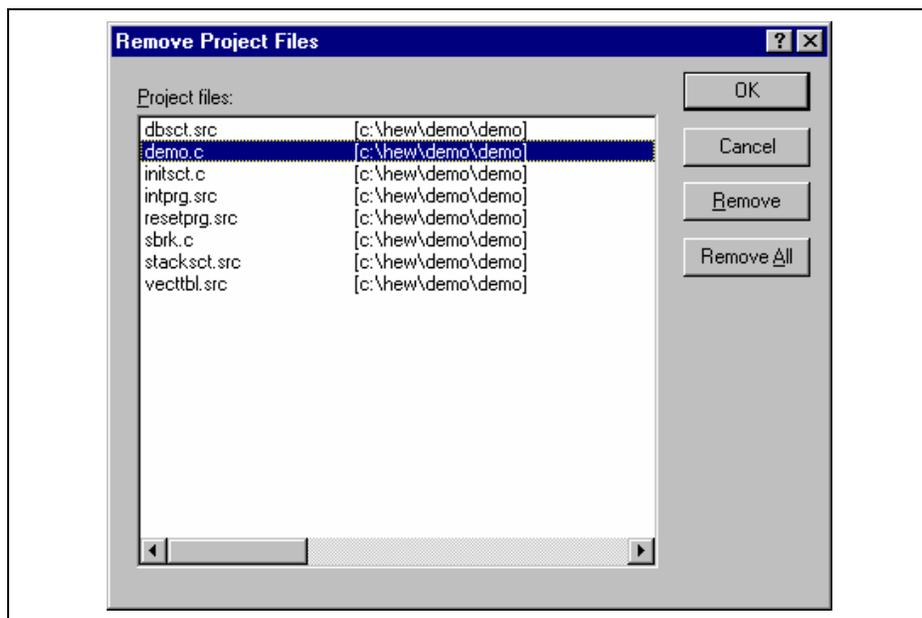


Figure 2.6: Remove Project Files Dialog

2. Select the file or files which you want to remove from the “Project files” list.
 3. Click the “Remove” button to remove the selected files or click “Remove All” to remove all project files.
 4. Click “OK” to remove the files from the project.
- To remove selected files from a project:
1. Select the files, which you want to remove, in the “Projects” tab of the “Workspace” window. Multiple files can be selected by holding down the **SHIFT** or **CTRL** key.
 2. Press the **DEL** key. The files will be removed.

2.2.3 Excluding a Project File from Build

A file in a project can be individually excluded from build on a configuration by configuration basis.

- To exclude a file in a project from build:
 1. Push the right mouse button on a file, which you want to be excluded from build, in the “Projects” tab of the “Workspace” window.
 2. Select [**Exclude Build file**], where <file> is the selected file, from the pop-up menu (figure 2.5). Then a red cross will be put on the file’s icon, and the file will be excluded from build.

2.2.4 Including a Project File in Build

An excluded file can be included in the project again.

- To include a file which has been excluded from build:
 1. Push the right mouse button on a file, which has been excluded from build, on the “Projects” tab of the “Workspace” window.
 2. Select [**Include Build file**], where <file> is the selected file, from the pop-up menu. Then a red cross will be removed from the file’s icon, and the file will be included in build.

2.3 File Extensions and File Groups

The HEW can identify files by their extension. The system defines certain extensions depending upon the tools, which are being used. For example, if you are using a compiler then the .c extension will be in the “C source file” group and be used as input to the compiler phase (figure 2.1, Typical Build Process). Additionally, the HEW allows you to define your own extensions. For example, if the project you are developing uses assembler source files the default extension may be .src. If you would like to use a different extension instead of .src (e.g. .asm) then you can define a new extension and request that the HEW treats it in the same way as a .src file.

File extensions and file groups can be viewed and modified via the “File Extensions” dialog (figure 2.7). This is invoked by selecting [Project->File Extensions...]. This dialog displays all of the extensions and file groups, which are defined within the current workspace.

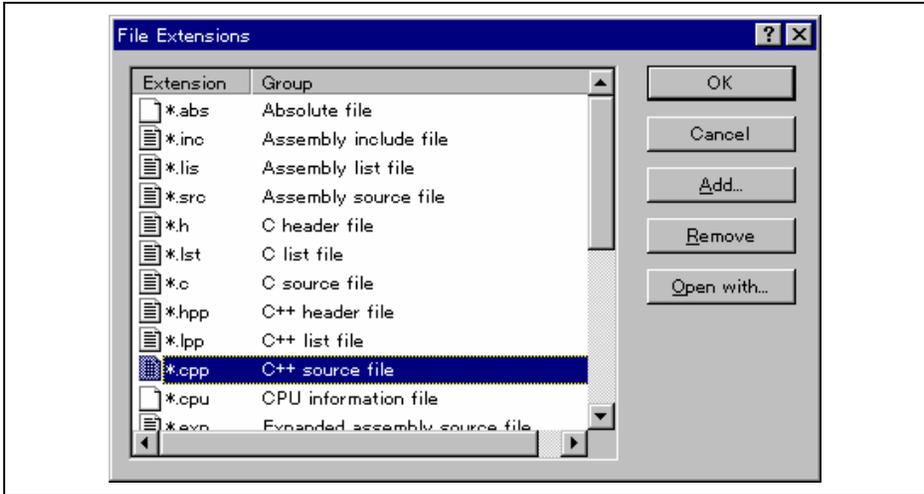


Figure 2.7: File Extensions Dialog

The “File Extensions” list shown in figure 2.7 is divided into two columns. On the left are the file extensions themselves, whilst on the right are the file groups. Many file extensions can belong to the same group. For example, assembler source files may have several extensions in a single project (e.g. .src, .asm, .mar etc) as shown in figure 2.8.

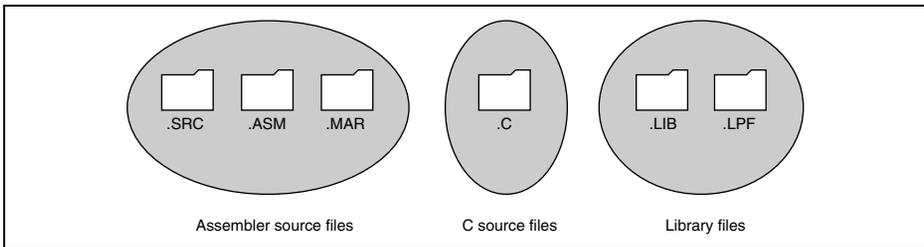


Figure 2.8: File Extensions and Groups

When creating a new extension you should consider whether the extension belongs to a group, which is already defined, or whether you need to create a new file group. If you are adding a

completely new type of file then you will want to create a new file group. This process is described below.

- To create a new file extension in a new file group:
1. Select [**Project->File Extensions...**] from the menu bar. The “File Extensions” dialog will be displayed (figure 2.7).
 2. Click the “Add...” button. The “Add File Extension” dialog will be displayed (figure 2.9).
 3. Enter the extension, which you want to define into the “File extension” field. It is not necessary to type the period (.) character. The drop list contains all extensions that are undefined in the current project. Selecting one of these extensions will add the text to the file extension field automatically.
 4. Select the “Extension belongs to a new group” option and enter a description, which defines this new file group.
 5. At this stage it is possible to change the associated application. There are four available choices in the “Open” with drop list. These are listed below:
 - Editor
 - None
 - Other
 - Windows default

If the editor is selected, the open file function in the workspace window causes the file to be opened in the HEW editor. If none is selected then the open operation is disabled when the open file function is attempted. Selecting “Other” allows you to configure an another tool for the open file operation. See *“To associate an application with a file group”* for more details. If the “Windows default” option is selected then the open file function in the workspace window passes the open file to the Windows operating system. This then selects the default behavior for this file extension as defined in Windows Explorer.

6. Click “OK” to add the extension to the “File Extensions” list.

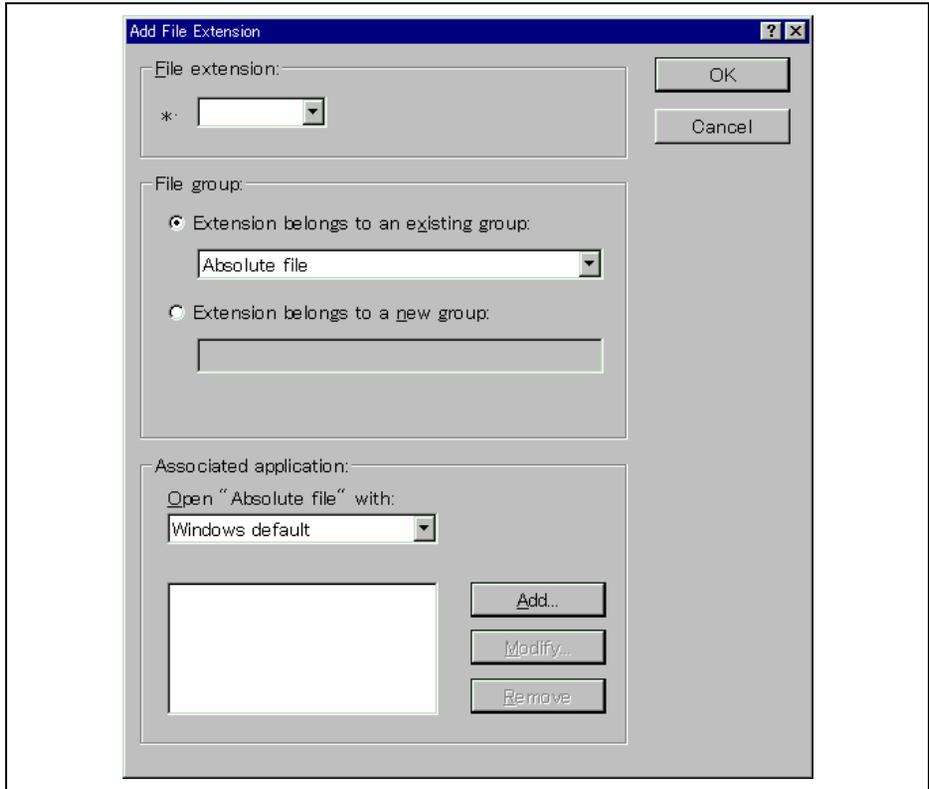


Figure 2.9: Add File Extension Dialog (New Group)

If you want to create a new extension because your project uses a different extension from those accepted by the HEW. For example, a phase might by default use the extension `.asm` but the HEW only recognizes `.src`. Then you need to create a new extension and add it to an existing file group. This process is described below.

- To create a new file extension in an existing file group:
 1. Select [**Project->File Extensions...**] from the menu bar. The “File Extensions” dialog will be displayed (figure 2.7).
 2. Click the “Add...” button. The “Add File Extension” dialog will be displayed (figure 2.10).
 3. Enter the extension, which you want to define into the “File extension” field. It is not necessary to type the period (.) character. The drop list contains all extensions that are undefined in the current project. Selecting one of these extensions will add the text to the file extension field automatically.
 4. Select the “Extension belongs to an existing group” option and select which group you would like to add this new extension.
 5. Click “OK” to add the extension to the “File Extensions” list.

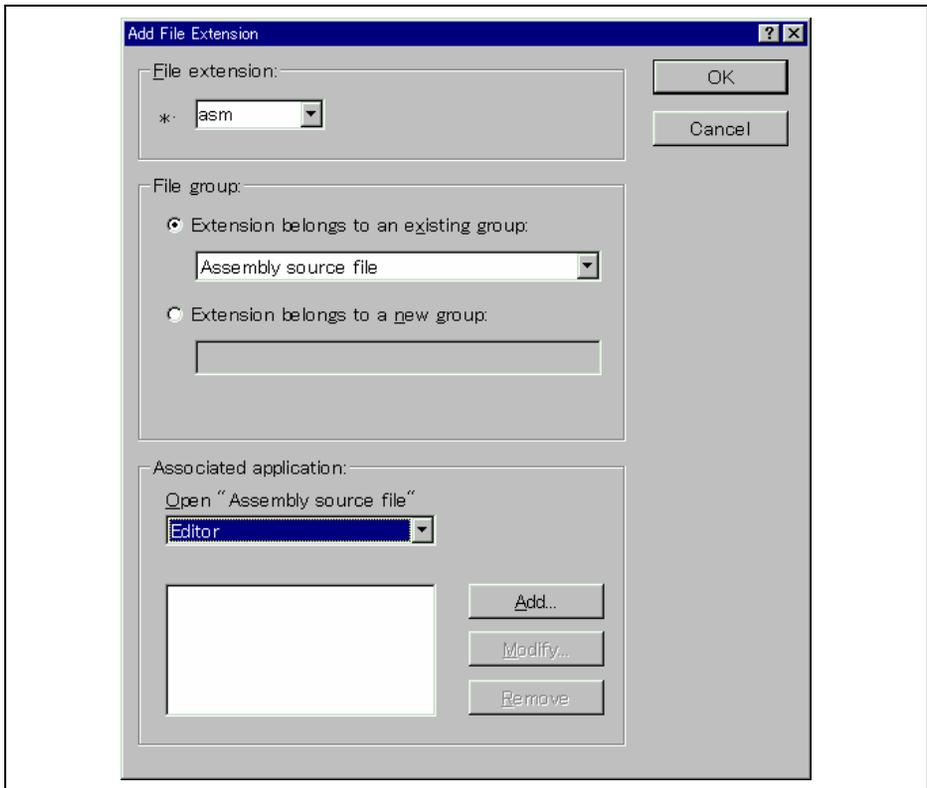


Figure 2.10: Add File Extension Dialog (Existing Group)

In addition to opening a file with the editor, the “File Extensions” dialog allows you to associate any application with any file group so that when you double click on a file in the “Projects” tab of the “Workspace” then the appropriate application is launched with the file. Figure 2.11 shows the association between a word processor and the extension .DOC.

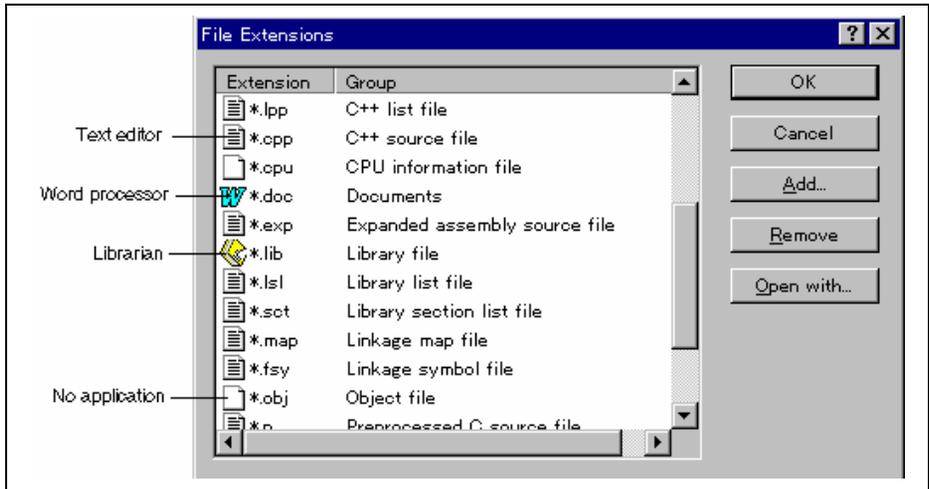


Figure 2.11: File Groups and Applications

- To associate an application with a file group:
 1. Select the file group to be associated from the “File Extensions” dialog (figure 2.11).
 2. Click the “Open with...” button. The “Modify File Extension” dialog will be displayed (figure 2.12).

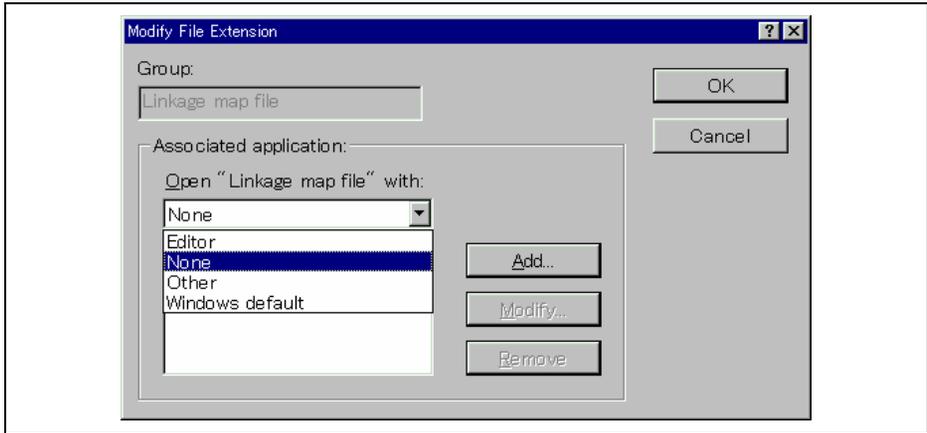


Figure 2.12: Modify File Extension Dialog

3. Select “None” to remove any association, select “Editor” to open this type of file in the internal/external editor or select “Other” if you want to open this type of file with a specific application. If you select “Other” then you can select from any previously defined application from the drop-down list or specify a new application.
4. Click “Add...” to define a new application. The “Add Application” dialog will be displayed (figure 2.13).

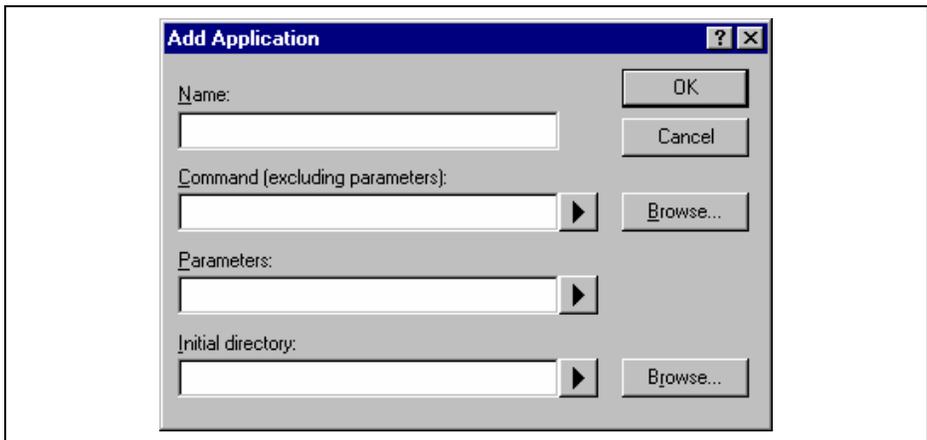


Figure 2.13: Add Application Dialog

5. Enter the name of the tool into the “Name” field. Enter the full path to the tool in the “Command” field (do not include any parameters). Enter the parameters that are required to open a file in the “Parameters” field. Be sure to use the \$(FULLFILE) placeholder to specify the location file (see appendix C, “*Placeholders*”, for more information on placeholders and their uses). Enter the initial directory, in which you would like the application to run, into the “Initial directory” field. Click “OK” to create the application.
6. Click “Modify...” to modify an application. The “Modify Application” dialog will be displayed. This dialog is the same as the “Add Application” dialog described above except that the “Name” field is read only. Modify the settings as desired and then click “OK”.
7. Click “OK” to set the application for the selected file group.

2.4 Specifying How to Build a File

Once you have added the necessary files to the project the next step is to instruct the HEW on how to build each file. To do this, you will need to select a menu option from the “Options” menu. The contents of this menu depend upon which tools you are using. For example, if you are using a compiler, assembler and linker then there will be three menu options, each one referring to one of the tools.

☞ To set options for a build phase:

1. Select the options menu and find the phase whose options you would like to modify. Select this option.
2. A dialog will be invoked which allows you to specify the options.
3. After making your selections, click “OK” to set them.

To obtain further information, use the context sensitive help button or select the area in which you need assistance and then press **F1**.

2.5 Build Configurations

The HEW allows you to store all of your build options into a build configuration (figure 2.14). This means that you can “freeze” all of the options and give them a name. Later on, you can select that configuration and all of the options for all of the build phases will be restored. These build configurations also allow the user to specify debugger settings for a build configuration. This means that each configuration can be targeted at a different end platform. (See Debugger Part in this manual, for further information).

Figure 2.14 shows three build configurations; “Default”, “MyDebug” and “MyOptimized”. In the first configuration, “Default”, each of the phases (compile and assemble) are set to their standard settings. In the second configuration, “MyDebug”, each of the files are being built with debug information switched on. In the third configuration, “MyOptimized”, each of the files are being built with optimization on full and without any debug information. The developer of this project can select any of those configurations and build them without having to return to the options dialogs to set them again.

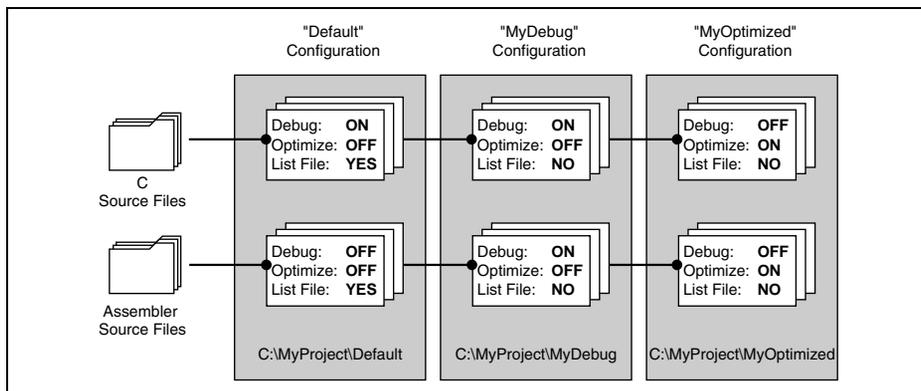


Figure 2.14: Configurations and File Options

2.5.1 Selecting a Configuration

The current configuration can be set in two ways:

Either:

1. Select it from the drop down list box (figure 2.15) in the toolbar.

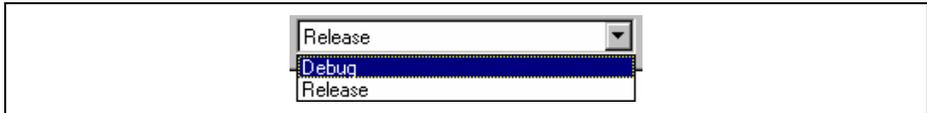


Figure 2.15: Toolbar Selection

or:

1. Select [**Options->Build Configurations...**]. This will invoke the “Build Configurations” Dialog (figure 2.16).

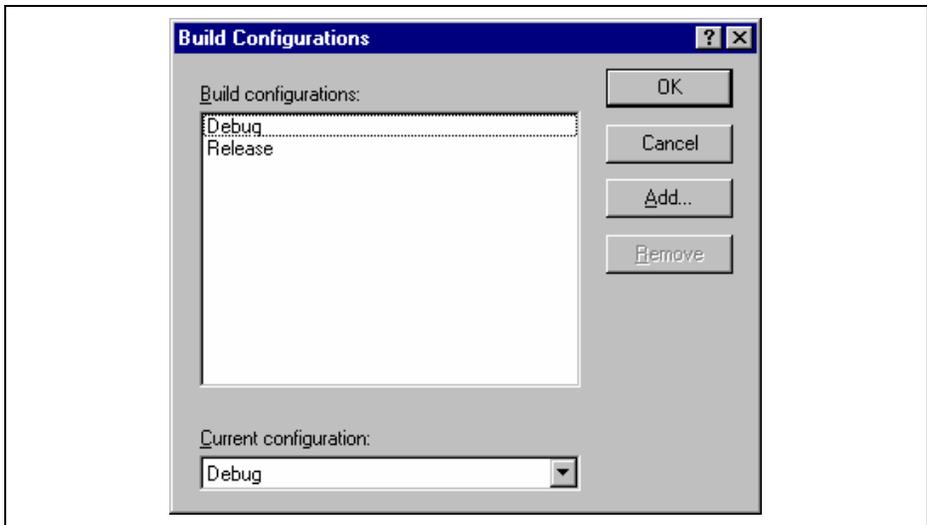


Figure 2.16: Build Configurations Dialog

2. Select the configuration that you want to use from the “Current configuration” drop down list.
3. Click “OK” to set the configuration.

2.5.2 Adding and Deleting Configurations

You can add a new configuration by copying settings from another configuration or delete a configuration. These three tasks are described below.

➤ To add a new configuration:

1. Select [**Options->Build Configurations...**] to display the “Build Configurations” dialog (figure 2.16).
2. Click the “Add...” button. The “Add Configuration” dialog will be invoked (figure 2.17).

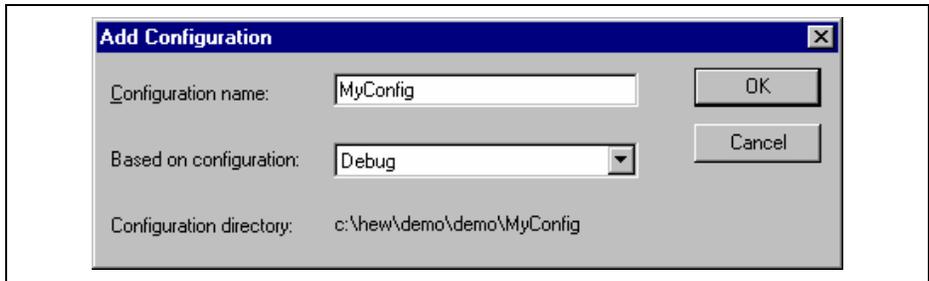


Figure 2.17: Add Configuration Dialog

3. Enter the new configuration name into the “Configuration name” field. As you enter the new configuration name, the directory underneath changes to reflect the configuration directory that will be used. Select one of existing configurations, from which you want to copy a configuration, out of the drop-down list of the “Based on configuration” field. Click “OK” on both dialogs to create the new configuration.

➤ To remove a configuration:

1. Select [**Options->Build Configurations...**] to display the “Build Configurations” dialog (figure 2.16).
2. Select the configuration that you want to remove and then click the “Remove” button.
3. Click “OK” to close the “Build Configurations” dialog.

2.6 Building a Project

The outline of the build process is shown in figure 2.1.

2.6.1 Building a Project

The build option only compiles or assembles those files that have changed since the last build. Additionally, it will rebuild source files if they depend upon a file that has changed since the last build. For instance, if the file “test.c” #include’s the file “header.h” and the latter has changed since the last build, the file “test.c” will be recompiled.

- ☞ To perform a build:
Select [**Build->Build**] or click the build toolbar button () or press **F7** or click the right mouse button on a project icon in the “Projects” tab of the “Workspace” window and select [**Build**] from the pop-up menu.

The build all option compiles and assembles all source files, irrespective of whether they have been modified or not, and links all of the new object files produced.

- ☞ To perform a build all:
Select [**Build->Build All**], or click the build all toolbar button () or click the right mouse button on a project icon in the “Projects” tab of the “Workspace” window and select [**Build All**] from the pop-up menu.

Both the build and the build all will terminate if any of the project files produce errors.

2.6.2 Building Individual Files

The High-performance Embedded Workshop lets you build project files individually.

- ☞ To build an individual file:
 1. Select the file which you want to build from the project window.
 2. Select [**Build->Build File**], click the build file toolbar button () or press **CTRL+F7** or click the right mouse button on a file icon in the “Projects” tab of the “Workspace” window and select [**Build <file>**] from the pop-up menu.

2.6.3 Stopping a Build

The High-performance Embedded Workshop allows you to halt the build process.

- ☞ To stop a build:
 1. Select [**Build->Stop Build**] or click the stop build toolbar button (). The build will be stop after the current file has been built.
 2. Wait until the message “Build Finished” appears in the “Output” window before continuing.
- ☞ To forcibly terminate a current tool
 1. Select [**Build->Terminate Current Tool**]. The HEW will attempt to stop the tool immediately.

Note: Do NOT assume that any output from the tool you terminated is valid. It is recommended that you delete any output files produced and ensure that the phase is executed again.

2.6.4 Building Multiple Projects

The High-performance Embedded Workshop lets you build multiple projects and configurations at once.

- ☞ To build multiple projects:
 1. Select [**Build->Build Multiple**]. The figure displayed in figure 2.18.
 2. The build multiple gives you the choice of which projects and configurations should be built. To select which projects and configurations need to be built select the check box next to the project – configuration combination you want to build. For example, in figure 2.18 if you wanted to build the entire “hewtest2” project you would check the “hewtest2-Debug” and the “hewtest2-Release” selections and leave all other check boxes unchecked.
 3. When you are happy with your chosen selection click the build button and the HEW will then build the projects and configurations you have chosen.
 4. If you want to build all the projects which you choose, you click the build all button.
 5. Results from the build are displayed in the build window in the same way as the normal build process.

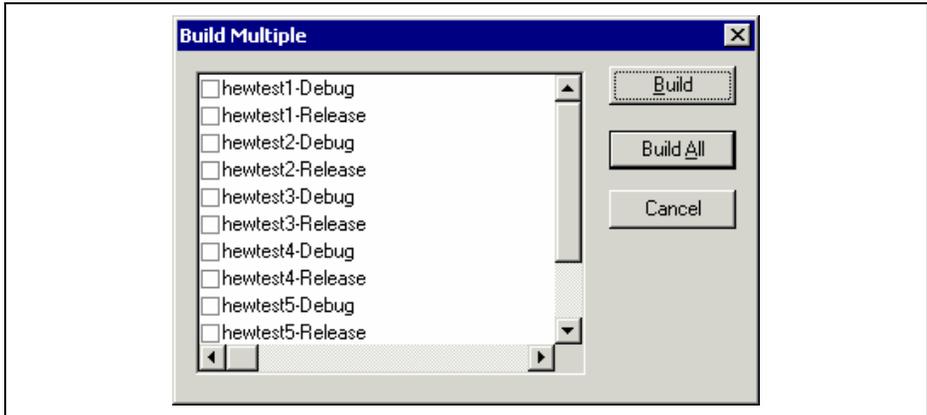


Figure 2.18: Build Multiple Dialog

2.6.5 The Output Window

When a tool executes (i.e. compiler, assembler, linker etc.) its output is displayed in the “Output” window. If any of the tools produce any errors or warnings then they are displayed along with the source file name and the line number at which the error is located. To quickly locate a specific bug, double click on a given error/warning to invoke the current editor.

2.6.6 Controlling the Content of the Output Window

It is often useful to display low-level information (such as the command line options that are being applied to a file) during a build. The HEW allows you to specify whether or not you want such options displayed in the “Output” window during a build, build all or build file operation via the “Tools Options” dialog.

- ☞ To view or hide extra information during a build:
 1. Select [**Tools->Options...**]. The “Options” dialog will be displayed.
 2. Select the “Build” tab (figure 2.19).
 3. Set the three check boxes in the “Show” group as follows. “Command line” controls whether the command line is shown as each tool is executed. “Environment” controls whether the environment is shown as each tool is executed. “Initial directory” controls whether the current directory is shown as each tool is executed.

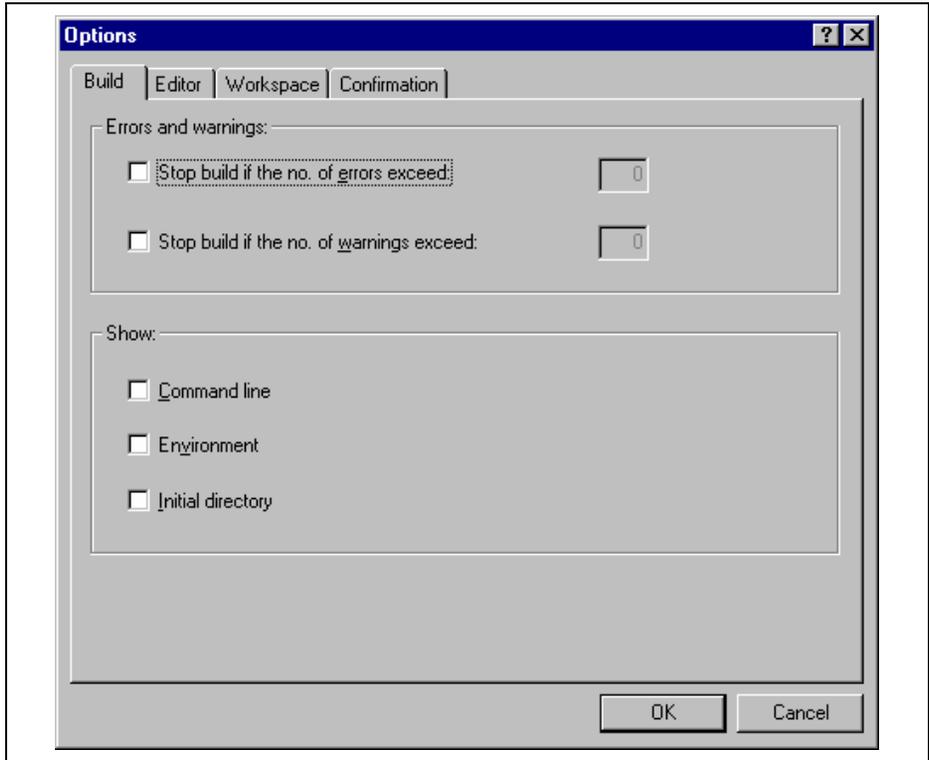


Figure 2.19: Options Dialog Build Tab

2.7 File Dependencies

A typical project will contain dependencies between files, for example, one C file may “#include” one or more header files. In complex projects, source files will include (or depend upon) others and this can quickly become difficult to manage. However, the HEW provides a dependency scanning mechanism whereby all files in a project are checked for dependencies. Once complete, the project window will display an up-to-date list with all the project file dependencies.

➡ To update a project’s dependencies:

Select [**Build->Update All Dependencies**] or click the right mouse button on a project icon in the “Projects” tab of the “Workspace” window and select [**Update All Dependencies**] from the pop-up menu.

Initially, the dependencies for all files are contained within the “Dependencies” folder (figure 2.20.i).

2.8 Configuring the Workspace Window

If you click the right mouse button anywhere inside the “Projects” tab of the “Workspace” window, a pop-up menu will be invoked. Select the “Configure View...” menu option to modify the way in which information is displayed. The following four sections detail the effect of each option on the “Configure View” dialog.

2.8.1 Show Dependencies under Each File

If you select “Show dependencies under each file”, the dependent files are shown under the including source file as a flat structure, i.e. the files themselves become folders (figure 2.20.ii). If this option is not selected then a separate folder contains all dependencies (figure 2.20.i).

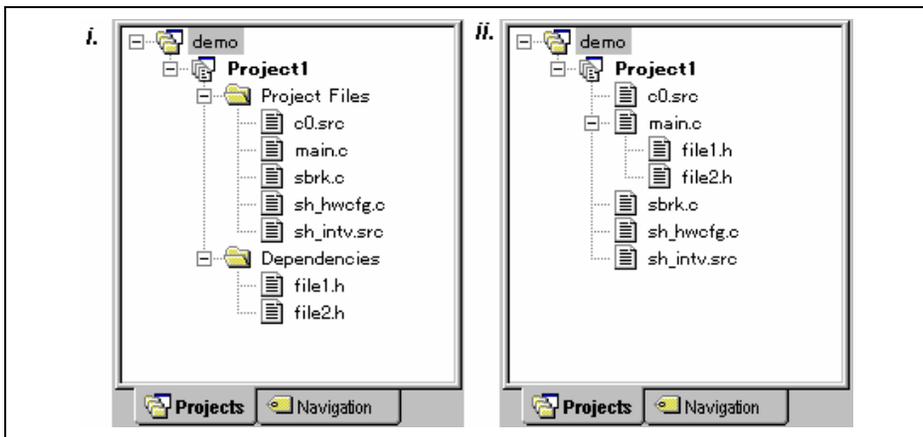


Figure 2.20: Dependencies under Each File

2.8.2 Show Standard Library Includes

By default, any dependent files found in standard include paths will not be shown (figure 2.21.i). For example, in C code, if you write an include statement such as “#include <stdio.h>” then `stdio.h` will not be listed as a dependent file. To view such system include files, select the “Show standard library includes” option (figure 2.21.ii).

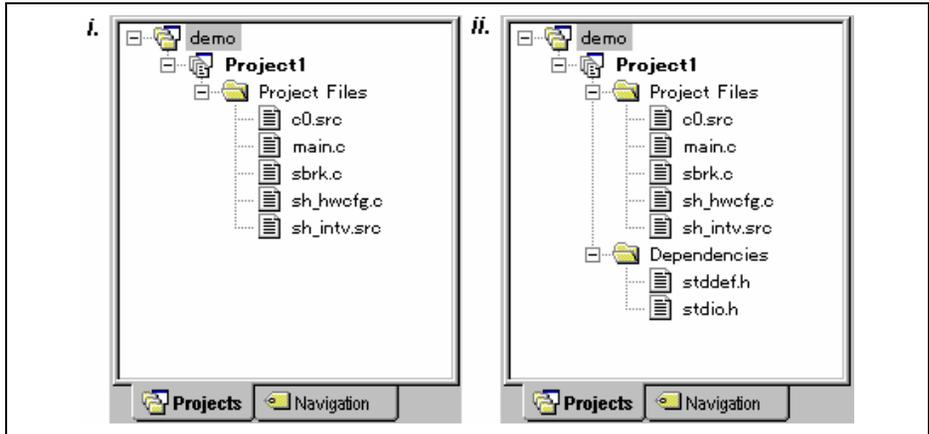


Figure 2.21: Standard Library Includes

2.8.3 Show File Paths

If “Show file paths” is selected, all of the files in the project window are shown with their full path, i.e. from a drive letter (figure 2.22).

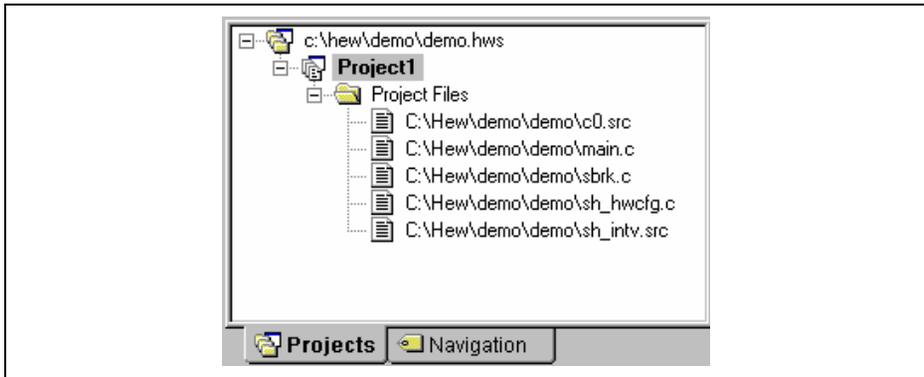


Figure 2.22: File Paths Shown

2.9 Setting the Current Project

A workspace can contain more than one project but only one of the projects can be active at any time. This active project is the one which build actions and debug operations can be performed on. It is possible to change the builder or debugger options for the project. An active project is displayed in bold.

- ☞ To set a project as the current project:
 1. Select the project from the “Projects” tab of the “Workspace” window.
 2. Click the right mouse button to display the pop-up menu and select the **[Set as Current Project]** option.
 or:
 1. Select the project, which you want to make active from the **[Project->Set Current Project]** sub-menu.

2.10 Inserting a Project into a Workspace

When a workspace is created, it contains only one project but, after it is created, you can insert new or existing projects into a workspace.

- To insert a new project into a workspace:
1. Select **[Project->Insert Project...]**. The “Insert Project” dialog will be displayed (figure 2.23).
 2. Set the “New Project” option.
 3. Click OK. The “Insert New Project” dialog will be invoked.
 4. Enter the name of the new workspace into the “Name” field. This can be up to 32 characters in length and contain letters, numbers and the underscore character. As you enter the project name the HEW will add a subdirectory for you automatically. This can be deleted if desired.
 5. Click the “Browse...” button to graphically select the directory in which you would like to create the project. Alternatively, you can type the directory into the “Directory” field manually.
 6. The “Project type” list displays all of the available project types (e.g. application, library etc.). Select the type of project that you want to create from this list.
 7. Click “OK” to create the project and insert it into the workspace.

Note: When a new project is being inserted, the CPU family and tool chain cannot be specified as these properties are already defined by the workspace (i.e. all projects within the same workspace target the same CPU family and toolchain).

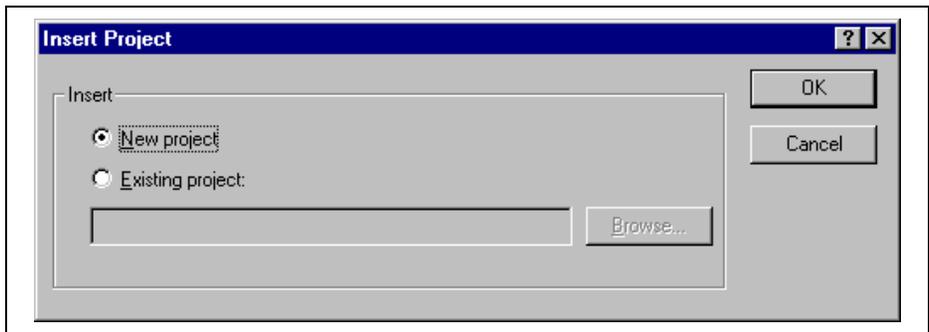


Figure 2.23: [Insert Project] Dialog

- ☞ To insert an existing project into a workspace:
 1. Select [**Project->Insert Project...**]. The “Insert Project” dialog will be displayed.
 2. Set the “Existing Project” option.
 3. Enter the full path of the project database file (.HWP file) into the edit field or click “Browse...” to search for it graphically.
 4. Click “OK” to insert the existing project into the workspace.

Note: When an existing project is being inserted into a workspace, the CPU family and tool chain upon which that project is based must match those of the current workspace. If they do not then the project cannot be inserted into the workspace.

2.11 Specifying Dependencies between Projects

The projects within a workspace can be dependent upon one another so that when one project is built, all its dependent projects are built first. This is useful if another project uses one of the others in the workspace. For example, imagine that a workspace contains two projects. The first project is a library that is included by an application project. In this case the library must have been built and up to date before the second application can build correctly. To achieve this situation we can specify the library as a dependent (i.e. child) project of the application project. This would then allow the library to be built first if it is out-of-date.

When a dependent project is built the HEW attempts to match the configuration in the dependent project with that of the current project. This means that if the current configuration is “Debug” then the HEW will attempt to build the “Debug” configuration in the dependent project. If this matched configuration does not exist then the HEW will use the configuration that was last used in the dependent project.

- ☞ To make projects depend upon another:
 1. Select [**Project->Dependent Projects**]. The “Dependent Projects” dialog will be displayed.(figure 2.24)
 2. Select the project to which you would like to add dependents to. When you do this, the “Dependent projects” list will display all of the projects in the workspace (excluding the selected project).
 3. The “Dependent projects” list has a check box for each project listed. Set the associated check boxes to make those projects depend upon the selected project.
 4. Click “OK” to confirm the new project dependencies.

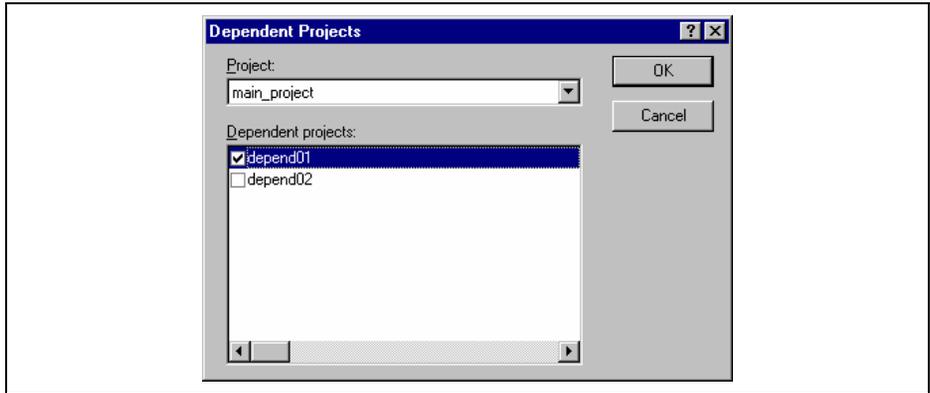


Figure 2.24 Dependent Projects dialog

2.12 Removing a Project from a Workspace

- ☞ To remove a project from a workspace:
 1. Select the project from the “Projects” tab of the “Workspace” window and click the right mouse button to invoke a pop-up menu.
 2. Select the **[Remove Project]** option.or:
 1. Select the project from the “Projects” tab of the “Workspace” window.
 2. Press the **DEL** key.

Note: You cannot remove the current project from the workspace.

2.13 Loading/Unloading a Project to/from a Workspace

- ☞ To load a project to a workspace:
 1. Select the project that has been unloaded from the “Projects” tab of the “Workspace” window.
 2. Click the right mouse button to invoke a pop-up menu and select the **[Load Project]** option.
- ☞ To unload a project from a workspace:
 1. Select the active project from the “Projects” tab of the “Workspace” window.
 2. Click the right mouse button to invoke a pop-up menu and select the **[Unload Project]** option.

Note: It is possible to select several projects at the same time and load or unload all of them. This is more efficient than loading or unloading each of the projects individually.

2.14 Relative Projects Paths in the Workspace

In the High-performance Embedded Workshop when you add a project to the workspace you can choose to add the project to the workspace using a relative path. This allows you to position a project above the workspace directory and it will still be relocated correctly if you relocate the HEW workspace. The project is always relative to the workspace so if the project is one directory above the workspace before it is moved the HEW will try to find the project in the same relative location after the relocation procedure. This is especially useful if you are using a project shared between more than one workspace.

In older versions of the HEW this project would not have been relocated and would have still tried to access the original project path. The older version of the HEW could only relocate the projects, which were in a subdirectory of the workspace directory. This is still the standard behavior for the High-performance Embedded Workshop.

☞ To change a projects relative path flag:

1. Select the project in the workspace window.
2. Right click and then select properties.
3. Click the “Project relative file path” checkbox to switch on or off the relative file path feature. (figure 2.25)
4. Click “OK”.

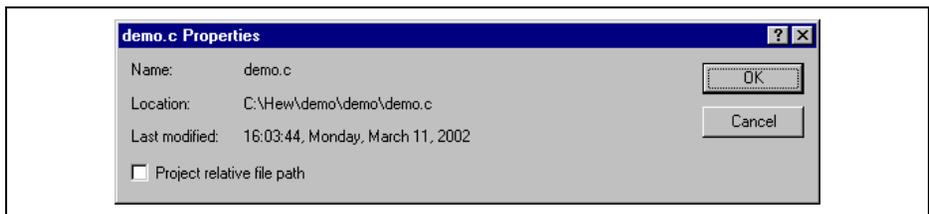


Figure 2.25: Properties Dialog

3. Advanced Build Features

This chapter explains the more advanced build concepts.

3.1 The Build Process Revisited

Chapter 2, “*Build Basics*” began by describing the build process in terms of a compiler, an assembler and a linker (figure 2.1). This will be the case for most installations of the High-performance Embedded Workshop. However, if you want to begin changing the build process (e.g. adding and removing phases) then it is important to understand more about the way in which a build functions.

3.1.1 What is a Build?

Building a project means applying a set of tools upon certain input files in order to produce the desired output. Thus, we apply a compiler upon C/C++ source files in order to create object files, we apply an assembler upon assembler source files in order to create object files and so forth. At each step or “phase” of the build, we apply a different tool upon a different set of input files.

Figure 3.1 presents another view of the build process.

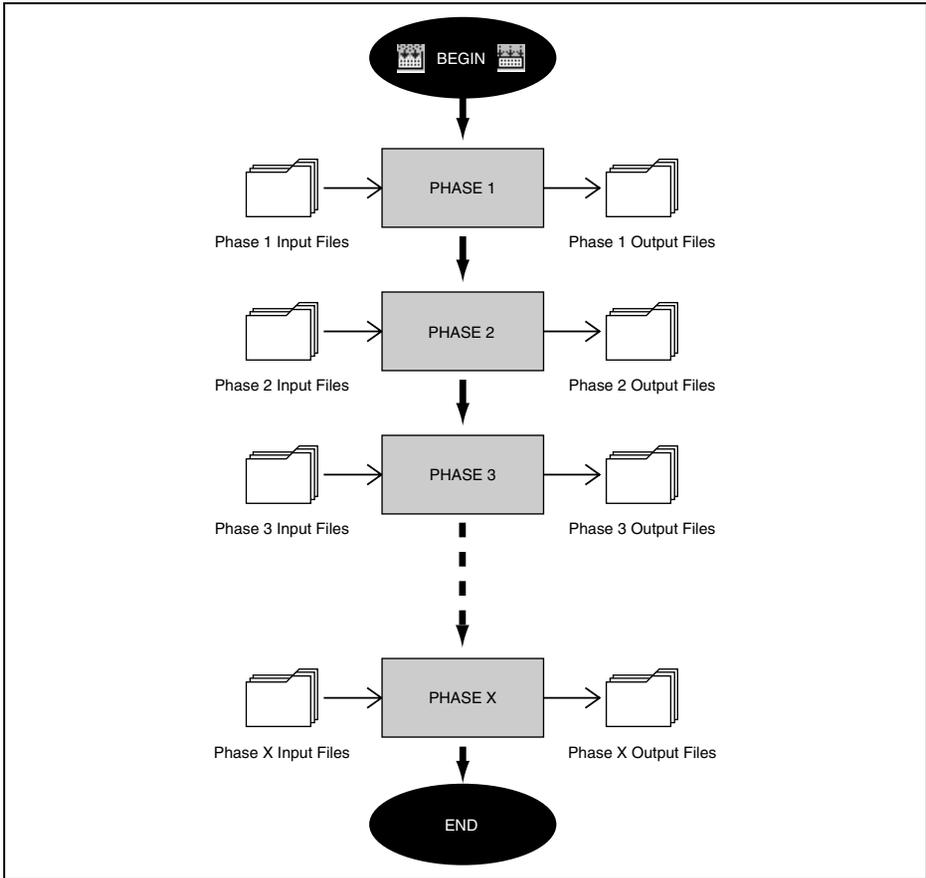


Figure 3.1: Build Process

The High-performance Embedded Workshop provides the ability to change this build process via its “Build Phases” dialog, which can be, accessed via the **[Options->Build Phases...]** (figure 3.2). On the left-hand side are the phases that are defined in the current project (Figure 3.2 shows a standard set of build phases). The remainder of this chapter details the various functions that the “Build Phases” dialog provides.

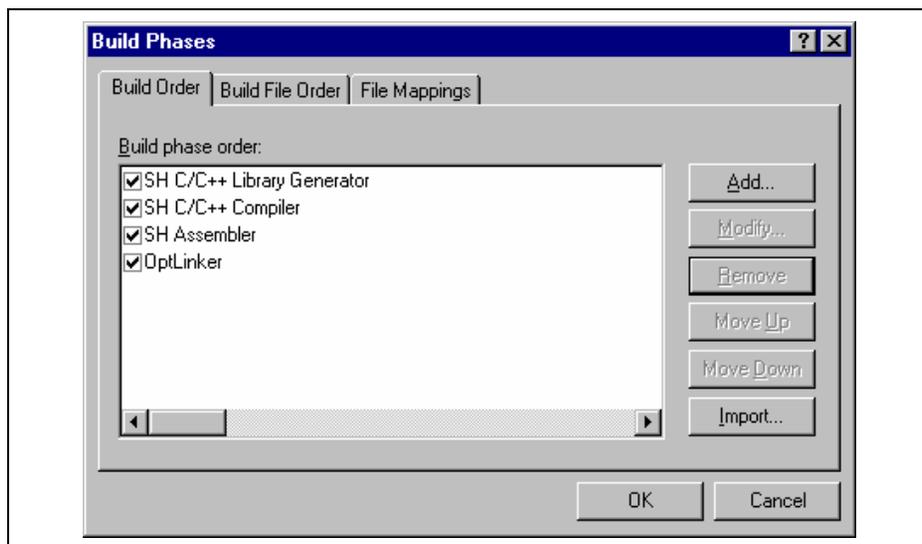


Figure 3.2: Build Phases Dialog

3.2 Creating a Custom Build Phase

If you want to execute another tool before, during or after a standard build process then this can be achieved by creating your own (i.e. custom) build phase.

Select [**Options->Build Phases...**] to invoke the “Build Phases” dialog (figure 3.2) and then click the “Add...” button. This will invoke the new build phase wizard dialog (figure 3.3a).

The first step (as shown in figure 3.3a) asks whether you want to create an entirely new phase or whether you want to add a system phase. A system phase is a “ready made” phase which is already defined within the toolchain you are using (e.g. compiler, assembler, linker, librarian, etc.) or a utility phase (e.g. file copy, complexity analyzer etc.).

The “Add an existing system phase” button is inactive if no more system phases are available. Select the “Create a new custom phase” button to create your own build phase.

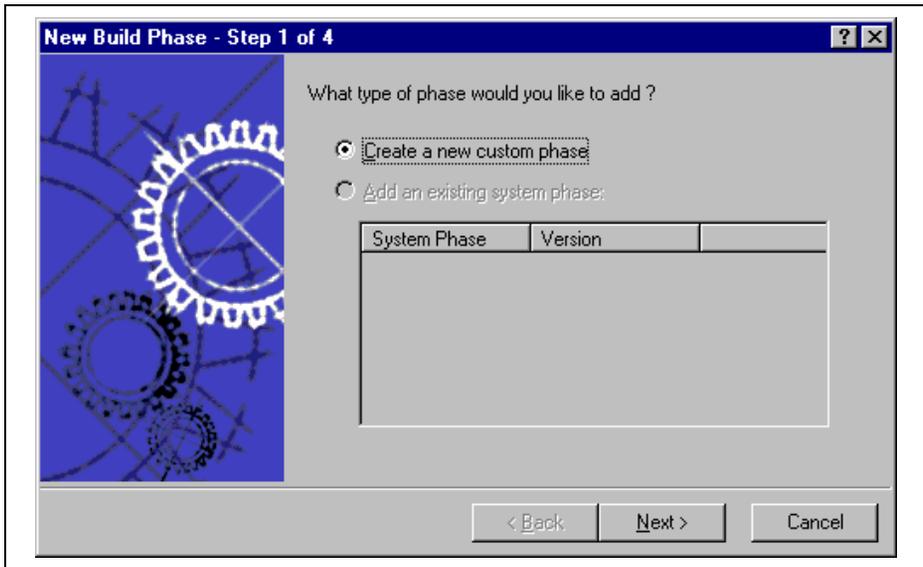


Figure 3.3a: New Build Phase Dialog (Step 1)

The second step (figure 3.3b) asks what type of phase you would like to create. There are two choices: multiple or single. When a multiple phase is executed, the command is applied to each file in the project of a certain file group. For example, if you set the input file group to be C source files then the command will be executed once for each C source file in the project. A single phase is executed once at most during a build.

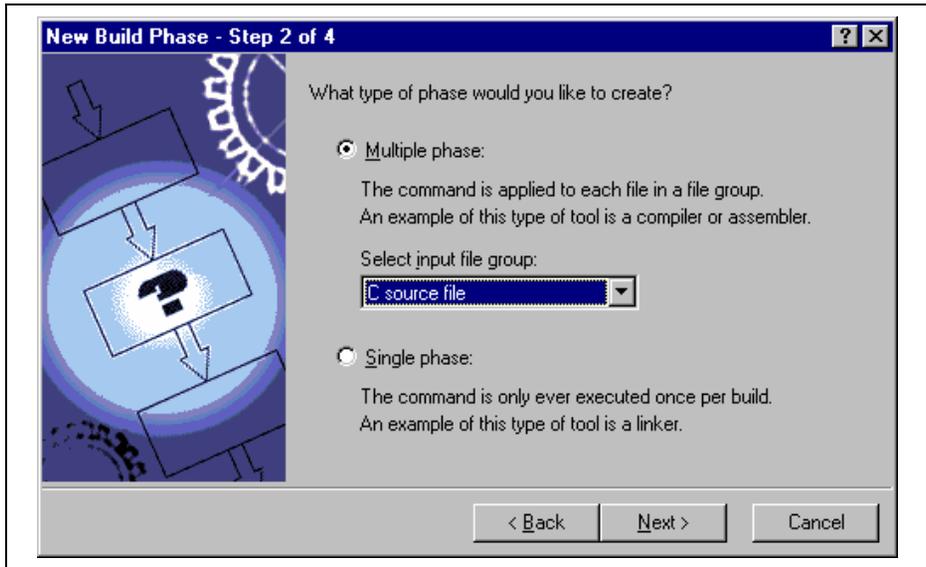


Figure 3.3b: New Build Phase Dialog (Step 2)

The input file group list contains the current file groups defined for the project. It is possible to define multiple input file groups by selecting the “Multiple Groups...” entry in the input file group list. Selecting this list entry displays the dialog in figure 3.3c.

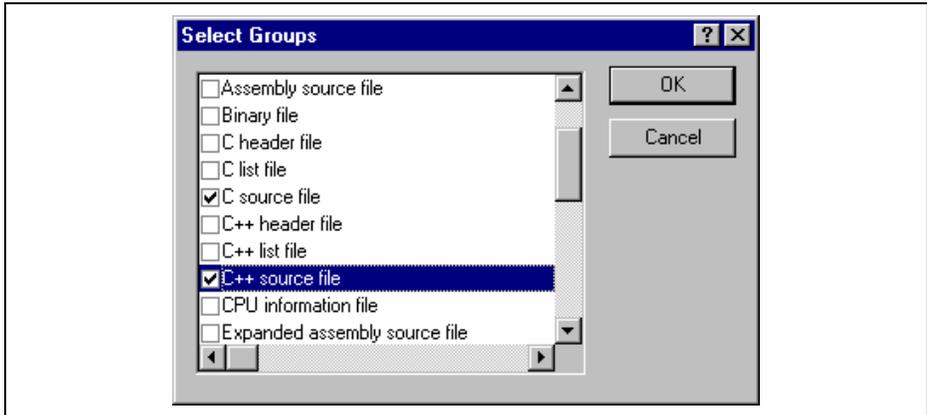


Figure 3.3c: Modify multiple input file groups

Once this choice has been made the input file group selection is displayed as “Multiple Groups...” This dialog allows the user to choose multiple input file groups for the custom phase being added to the project. To select a file group check the box next to the file groups name. One or more file groups can be selected in this dialog.

The third step (figure 3.3d) requests the fundamental information about the new build phase. Enter the name of the phase into the “Phase name” field. Enter the location of the program file into the “Command” field (do not insert any command line options as these options are specified via the **[Options]** menu of the HEW menu bar). Specify the default options for the phase (i.e. what options you would like new files to take when added to the project) into the “Default options” field. If you have a preferred directory in which you would like this program to run from (i.e. where you want the current working directory to be set to before the tool is executed) then enter it into the “Initial directory” field.

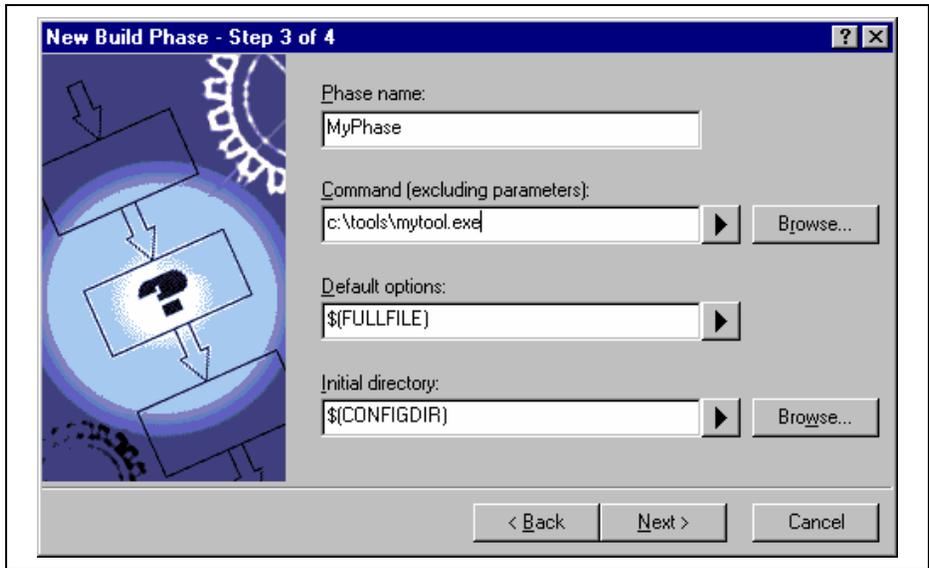


Figure 3.3d: New Build Phase Dialog (Step 3)

The fourth and final step (figure 3.3e) allows you to specify any environment variables, which the phase requires.

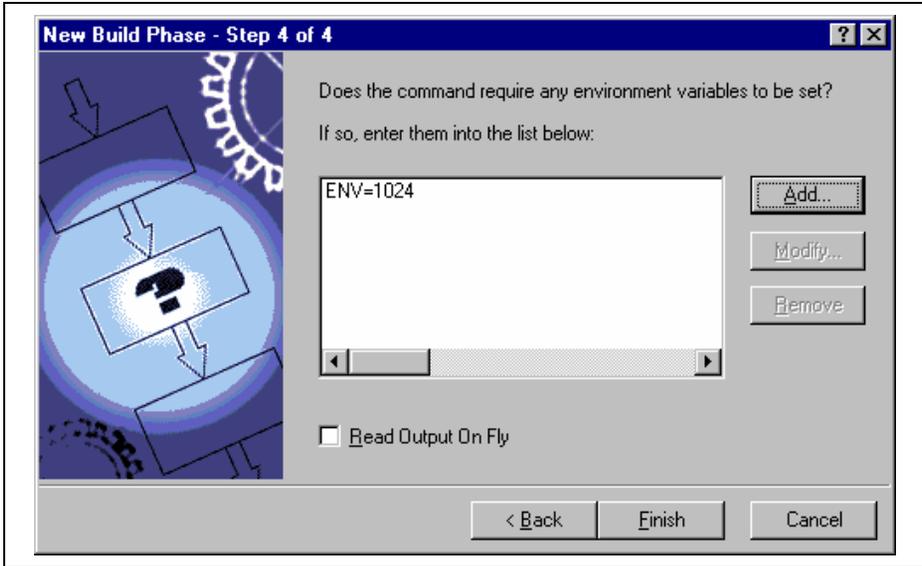


Figure 3.3e: New Build Phase Dialog (Step 4)

To add a new environment variable click the “Add...” button (the dialog shown in figure 3.4 will be invoked). Enter the variable name into the “Variable” field and the variable’s value into the “Value” field and then click “OK” to add the new variable to the list of the fourth step. To modify an environment variables select the variable in the list and then click the “Modify...” button. Make the required changes to the “Variable” and “Value” fields and then click “OK” to add the modified variable to the list. To remove environment variables select the variable that you want to remove from the list and then click the “Remove” button.

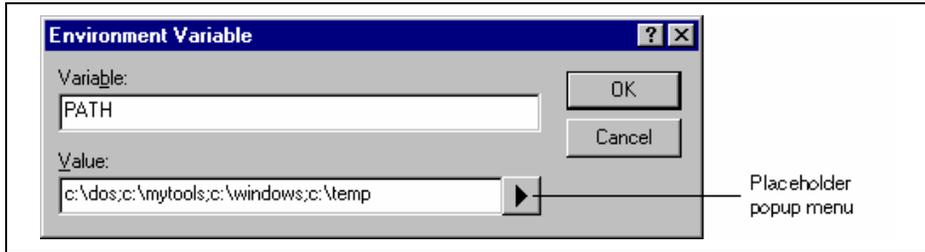


Figure 3.4: Environment Variable Dialog

If the tool you are adding can display its output as the tool is running then use the ‘Read Output On Fly’ option. This will display the tool output as each line of output happens. If this option is set to off then the HEW will store all output, which is being displayed by the tool, and display it in the output window when the tool has finished its operation. This can be a problem when the tool is running an operation that might take many minutes, as it is difficult to see the progress of the current execution.

Note: Using ‘Read Output On Fly’ can cause problems when using certain tools on certain operating systems. If you are having problems with tools locking up or freezing in HEW then uncheck the ‘Read Output On Fly’ option.

Click the “Finish” button to create the new phase. By default the new phase is added to the bottom of the “Build Phase Order” list in the “Build Order” tab of the “Build Phases” dialog (Figure 3.2).

3.3 Ordering Build Phases

In a standard build (shown in figure 3.5), you could add a phase at four different positions: before the compiler, before the assembler, before the linker or after the linker. You may place your own custom phases or move system phases to any position in the build order. It is important to remember that if the output of your custom phase can be input into another phase then the phase order must be correct if the build is to behave as intended.

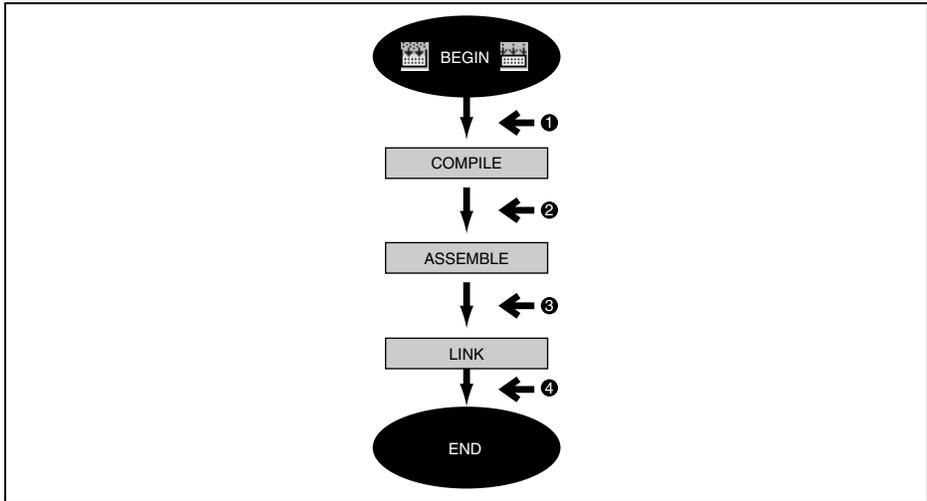


Figure 3.5: Typical Build Process

The build phase dialog provides facilities for ordering build phases via the “Build Phases” dialog. It has two tabs, which are concerned with the ordering of phases: “Build Order” and “Build File Order”.

3.3.1 Build Phase Order

The “Build Order” tab (figure 3.6) displays the current order in which phases will be executed when the build () or build all () operation is selected. The check box to the left of each phase indicates whether or not it is currently enabled. By clicking this box, the phase can be toggled on or off.

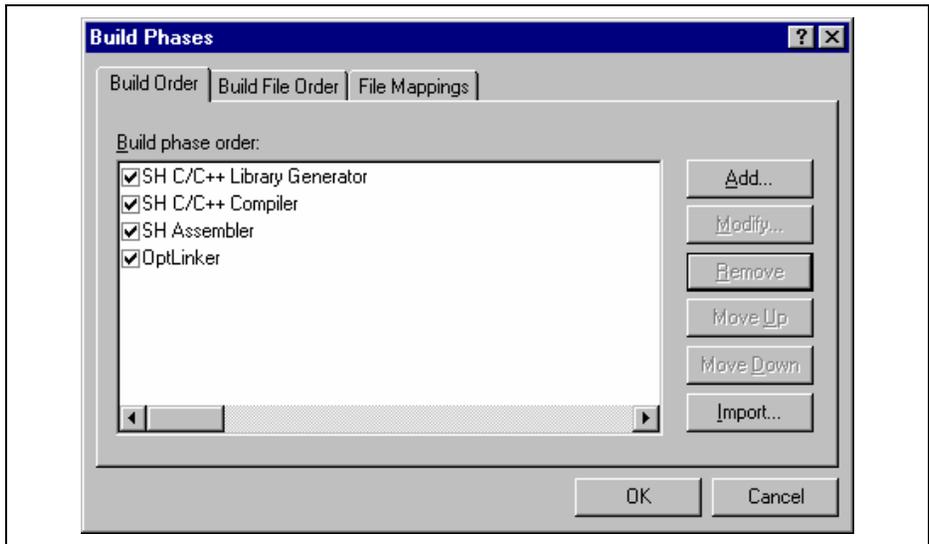


Figure 3.6: Build Phases Dialog Build Order Tab

In addition the following operations can be performed:

- To remove a phase:
 1. Select the phase that you would like to remove.
 2. Click the “Remove” button.
- To view the properties of a system phase:
 1. Select the system phase that you would like to examine.
 2. Click the “Modify...” button.
- To move a phase:
 1. Select the phase that you would like to move.
 2. Click the “Move Up” or “Move Down” button.
- To import a phase:

High-performance Embedded Workshop User Manual

1. Click the import button. A dialog is displayed which allows the user to browse to an existing project to import a custom phase from.
2. Choose the location of the project you wish to import a custom phase from. Once selected a dialog is displayed which lists the custom phases in the imported project.
3. Selecting a phase name and then clicking properties displays the custom phase details. This allows you to decide whether the phase does the functionality you require.
4. Once you have decided which phase to import highlight it in the list and then click OK. The phase will then be added to the build phases dialog at the bottom of the build order.

- To modify a custom phase:
 1. Select the custom phase that you would like to modify.
 2. Click the “Modify...” button. The modify phase dialog will be invoked with the “Command” tab selected (figure 3.7).
 3. Change the contents of the fields as appropriate.
 4. Set the “Don’t check for input file(s) existence before executing” check box if you don’t want the HEW to abort the execution of the phase if any of the input files don’t exist.

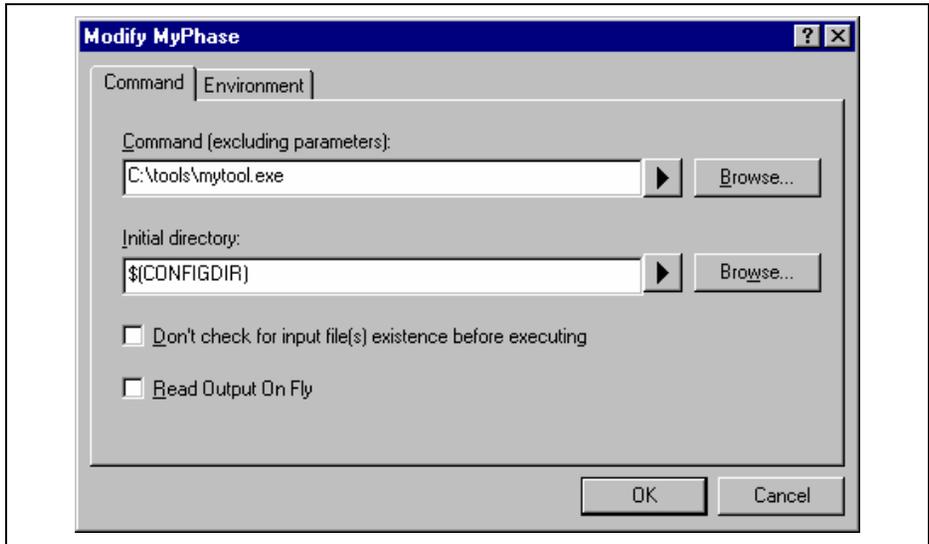


Figure 3.7: Modify Phase Dialog Command Tab

5. Select the “Environment” tab (figure 3.8) to edit the environment settings for the phase.
6. Use the “Add...”, “Modify...” and “Remove” buttons to add, modify and remove environment variables. The operation is the same as discussed in the previous section.
7. Click “OK” when all modifications have been made.

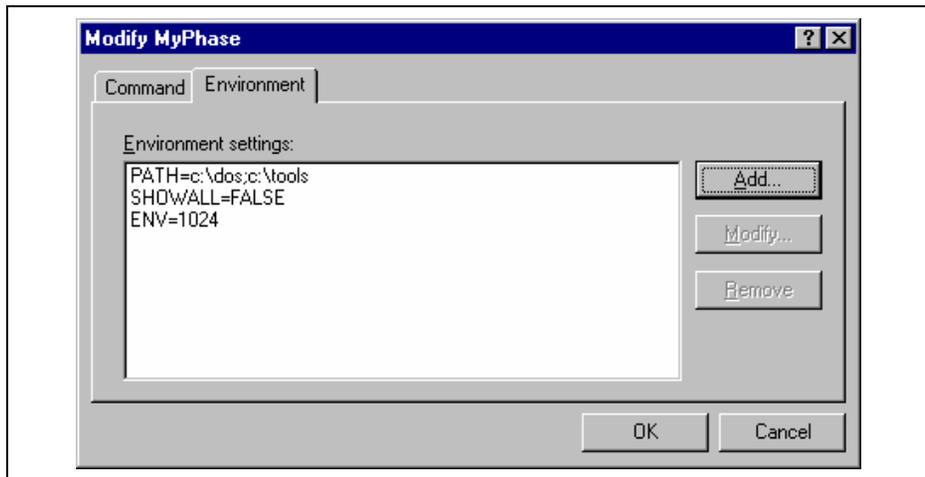


Figure 3.8: Modify Phase Dialog Environment Tab

3.3.2 Build File Phase Order

If you were to select a C source file from the “Workspace” window and then activate [**Build->Build File**] (or press ) you would expect the file to be compiled. Likewise, if you were to select an assembly source file from the workspace window and then activate [**Build->Build File**] you would expect the file to be assembled. The connection between file group and which phase(s) to execute is managed by the “Build File Order” tab of the “Build Phases” dialog (figure 3.9).

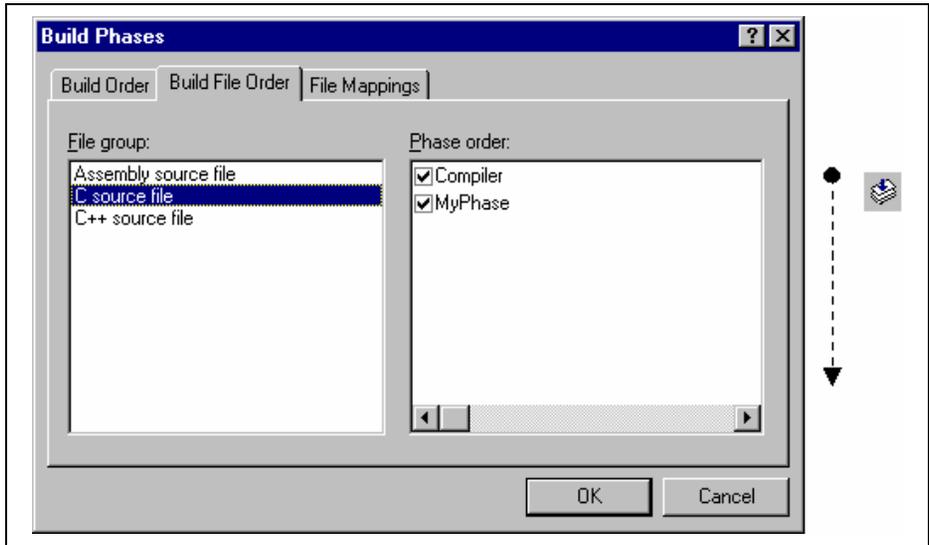


Figure 3.9: Build Phases Dialog Build File Order Tab

The list displays all of the current phases that will be executed when the build file operation is selected upon the file group shown in the “File group” list box. In figure 3.9 the “C source file” file group is selected and the “Compiler” and “MyPhase” phases are associated with it.

Entries in the “Phase order” list, of the “Build File Order” tab, are added automatically as new entries are added to the “Build Order” tab. For example, if you were to add a phase which takes C source files as input then this phase will be automatically added to the list of phases to execute when a build file operation is applied to a C source file. If you don’t want a certain phase to execute when [**Build->Build File**] is selected then clear the check box to the left of the phase name in the “Phase order” list.

3.4 Setting Custom Build Phase Options

Once you have defined a custom phase, you will want to specify the command line options that should be used when it is executed. Each defined phase has a menu option on the **[Options]** menu. To specify options for that phase select it. The dialog that will be invoked depends upon whether the custom phase selected was a multiple or single phase (according to the selection of phase type in figure 3.3b).

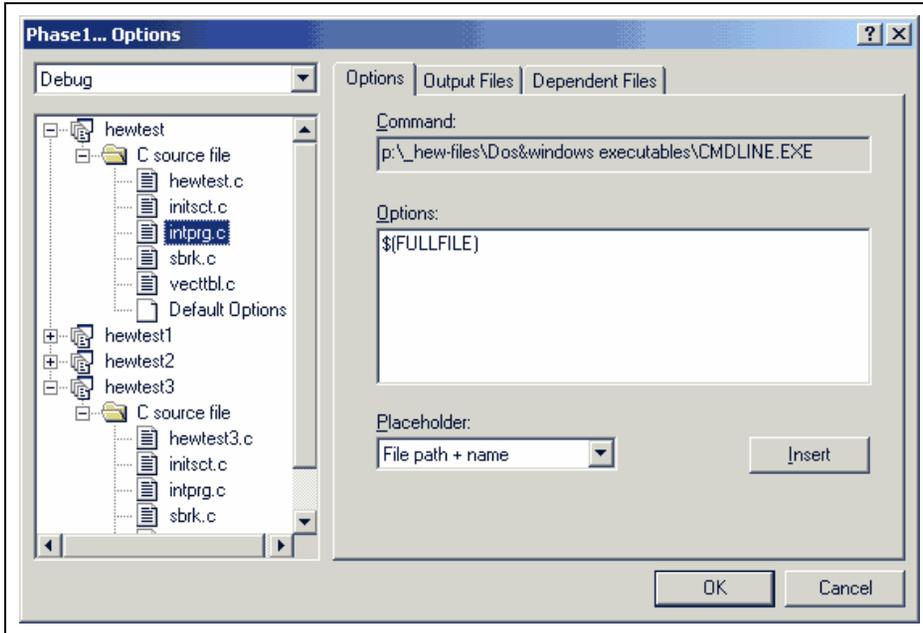


Figure 3.10: Custom Options Dialog

The dialog in figure 3.10 is a custom phase options dialog. The implementation of which is slightly different depending on whether you are using a multiple or single shot phase. On the left-hand side is the project and file list. It is possible to select multiple projects and files in the same way as Windows explorer to modify the options for more than one selection. On the right-hand side are the 3 options tabs. This is where you set the options that you want to apply to the selected file(s). You can also choose which configuration information is being viewed from the configuration list on the upper left of the dialog box. Each configuration is listed along with a special entry named “Multiple configurations...”. If you select multiple configurations then a

dialog is displayed which allows you to select more than one configuration. This method is used throughout HEW for modifying multiple configurations at once.

3.4.1 Options Tab

The “Options” tab (figure 3.11) allows you to define the command line options that will be passed to the phase. The “Command” field displays the command, which was entered when you defined the phase (figure 3.3d). Enter into the “Options” field the command line arguments that you would like to pass to the command. If you want to insert a placeholder, select the relevant placeholder from the “Placeholder” drop-down list box and then click the “Insert” button. For a detailed description of placeholders see appendix C, “*Placeholders*”.

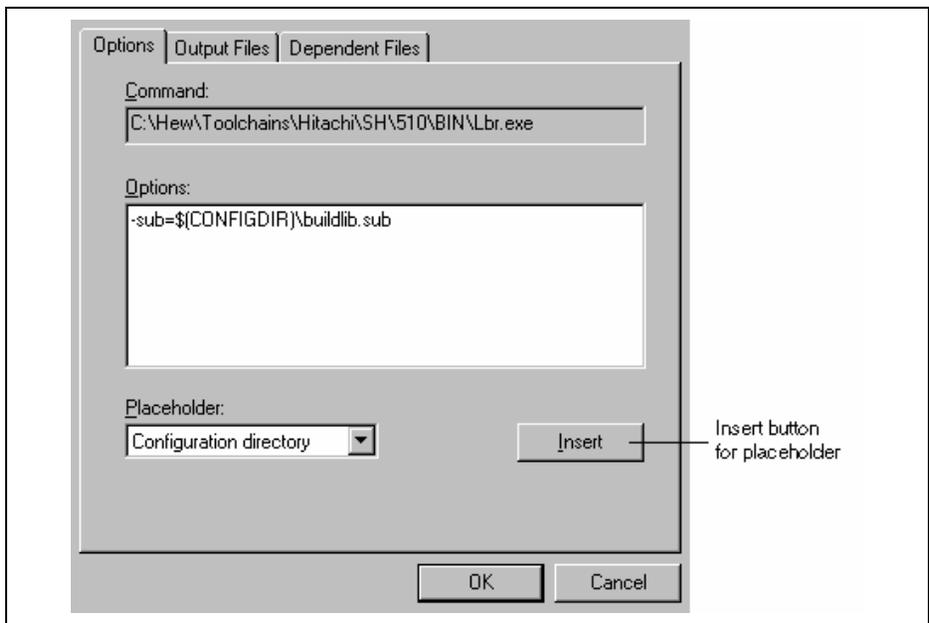


Figure 3.11: Custom Options Options Tab

3.4.2 Output Files Tab

The “Output Files” tab (figure 3.12) is where you can specify the output file or files that will be produced by the phase. Before each file is passed into this phase, the HEW checks that the output files are of a less recent date than the input file. If so, the phase will be executed for that file (i.e.

input files have been modified since the output file or files were last produced). If the files are up to date then the phase will not be executed.

Note: If no output files are specified, the phase will execute regardless.

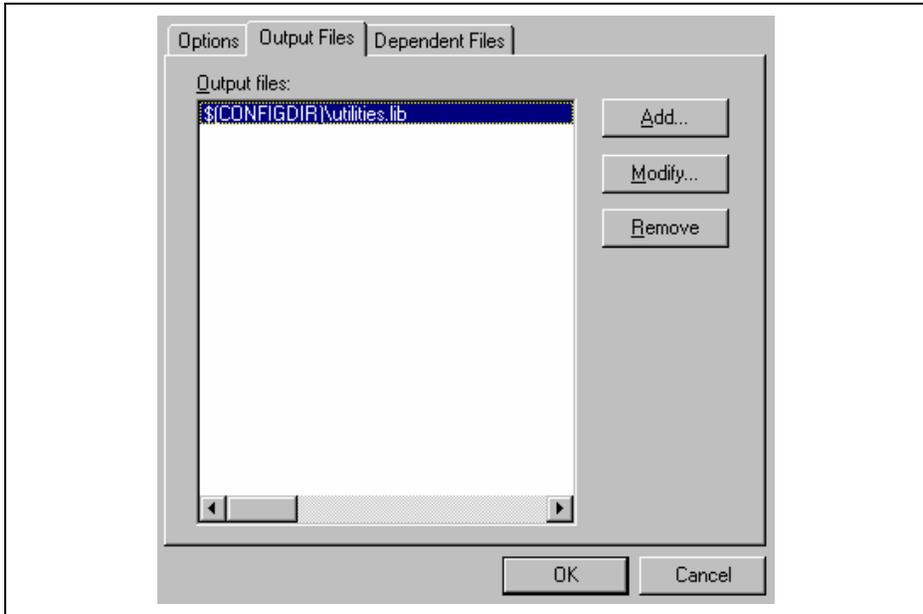


Figure 3.12: Custom Options Output Files Tab

- To add an output file:
 1. Click "Add...". The "Add Output File" dialog will be invoked (figure 3.13).
 2. Enter the file path or browse to it using the "Browse..." button.
 3. Click "OK" to add this output file to the list.

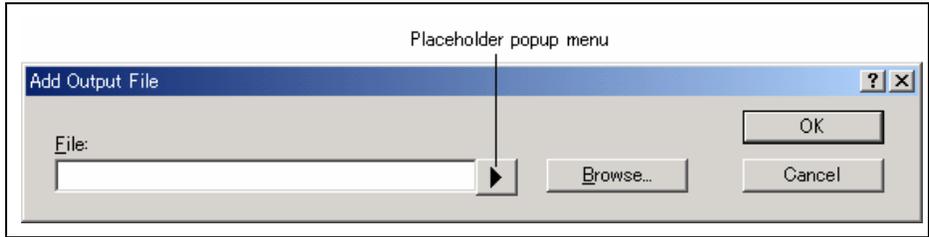


Figure 3.13: Add Output File Dialog

- To modify an output file:
 1. Select the output file that you would like to modify.
 2. Click “Modify...”. The “Modify Output File” dialog, which is the same as figure 3.13 except the title, will be invoked.
 3. Modify the fields as required and then click the “OK” button to add the modified entry back to the list.
- To remove an output file:
 1. Select the output file that you would like to remove.
 2. Click the “Remove” button.

3.4.3 Dependent Files Tab

The “Dependent Files” tab (figure 3.14) is where you can specify the dependent files that are needed by the phase. Before each file is passed into this phase, the HEW checks that the dependent files are of a more recent date than the input file. If so, the phase will be executed for that file (i.e. dependent files have been modified since the input file or files were last modified). If not, the phase is not executed for the files.

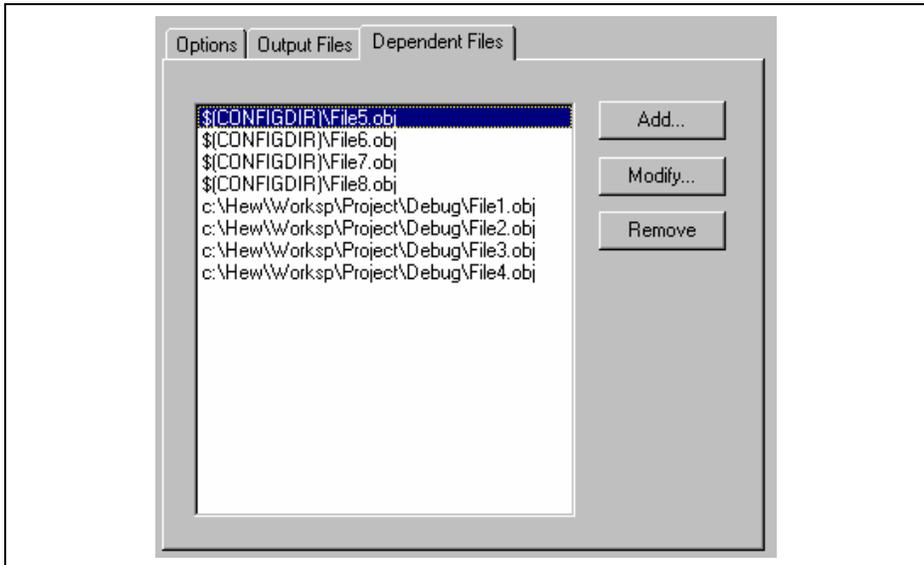


Figure 3.14: Dependent Files Tab in Custom Options

- To add a dependent file:
 1. Click “Add...”. The “Add Dependent File” dialog will be invoked (figure 3.15).
 2. Enter the file path or browse to it using the “Browse...” button.
 3. Click “OK” to add this output file to the list.

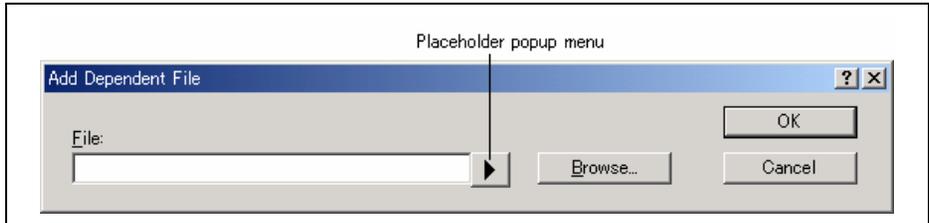


Figure 3.15: Add Dependent File Dialog

- To modify a dependent file:
 1. Select the dependent file that you would like to modify.
 2. Click “Modify...”. The “Modify Dependent File” dialog, which is the same as figure 3.15 except the title, will be invoked.
 3. Modify the fields as required and then click the “OK” button to add the modified entry back to the list.
- To remove a dependent file:
 1. Select the dependent file that you would like to remove.
 2. Click the “Remove” button.

3.5 File Mappings

By default, the files input to a phase are only taken from the project, i.e. all project files of the type specified in the “Select input file group” drop-down list on the “New Build Phase” dialog (figure 3.3b). If you would like a phase to take files output from a previous phase (i.e. intermediate files) then you must define this in the “File Mappings” tab of the “Build Phases” dialog (figure 3.16).

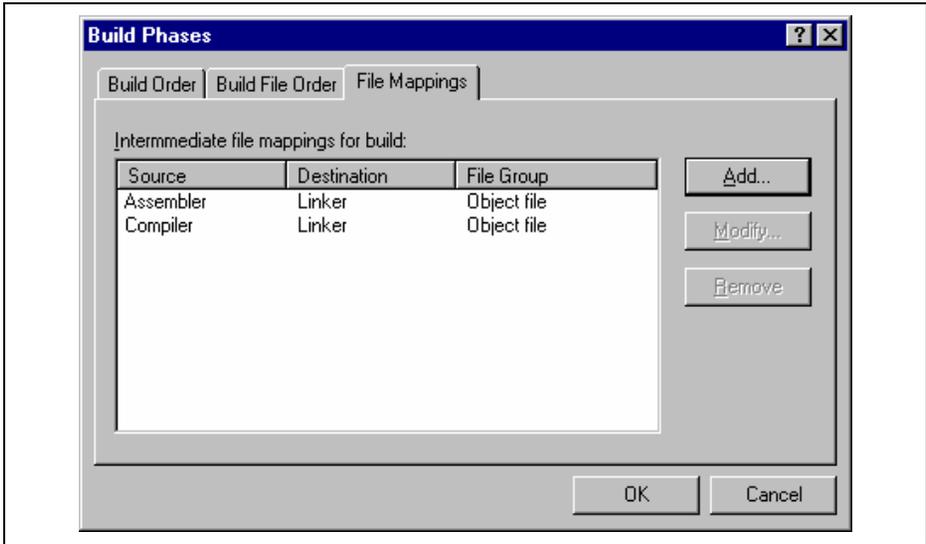


Figure 3.16: Build Phases Dialog File Mappings Tab

A file mapping states that you would like the HEW to pass output files of a certain type produced by one phase (referred to as the source phase) to another phase (referred to as the destination phase). Such intermediate files are passed in addition to the project files.

- To add a file mapping:
 1. Click “Add...”. The “Define File Mapping” dialog will be invoked (figure 3.17).
 2. Select the file group, which you want to pass between the phases from the “File group” drop-down list box.
 3. Select the source phase (i.e. which phase generates the files) from the “Source phase” drop-down list box.
 4. Select the destination phase (i.e. which phase takes these files) from the “Destination phase” drop-down list box.
 5. Click “OK” to create the new mapping.

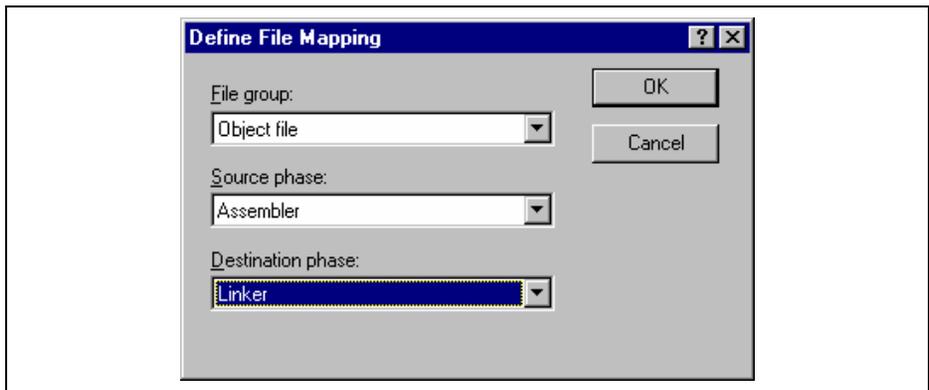


Figure 3.17: Define File Mapping Dialog

- To modify a file mapping:
 1. Select the mapping to be modified.
 2. Click “Modify...” button. The “Define File Mapping” dialog will be invoked (figure 3.17).
 3. Modify the options as necessary.
 4. Click “OK” to commit the changes.

3.6 Controlling the Build

By default, the High-performance Embedded Workshop will execute all of the phases in a build and only stop if a fatal error is encountered. You can change this behavior by setting the controls on the “Build” tab of the “Options” dialog (figure 3.18).

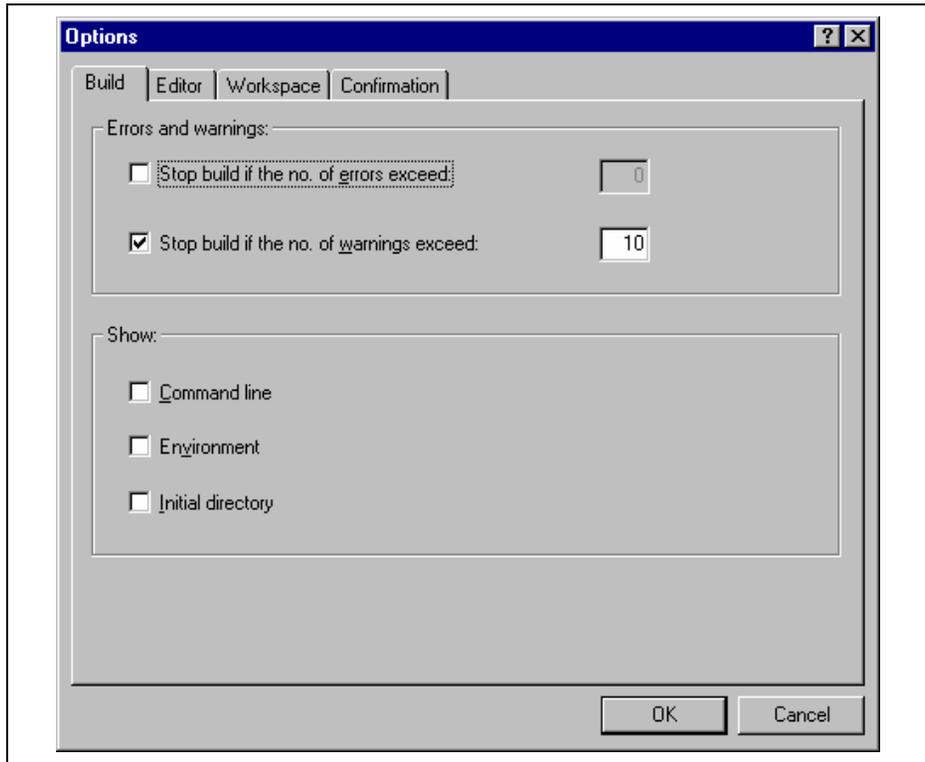


Figure 3.18: Options Dialog Build Tab

Select [**T**ools->**O**ptions...] to invoke the dialog. If you want to stop the build when a certain number of errors are exceeded then set the “Stop build if the no. of errors exceed” check box and then specify the error count limit in the edit field to the right. If you want to stop the build when a certain number of warnings are exceeded then set the “Stop build if the no. of warnings exceed” check box and then specify the warning count limit in the edit field to the right.

Note: Irrespective of what these controls are set to, the build will always halt if a fatal error is encountered.

In addition to specifying error and warning count limits, the “Build” tab also allows you to request that the command line, environment and initial directory of each execution should be displayed. Check the appropriate check boxes as necessary.

3.7 Logging Build Output

If you would like to write the results of each build to file then invoke the “Customize” dialog by selecting [Tools -> Customize...] and select the “Log” tab (figure 3.19). Set the “Generate log file” check box and then enter the full path of the log file into the “Path” field or browse to it graphically by clicking the “Browse...” button.

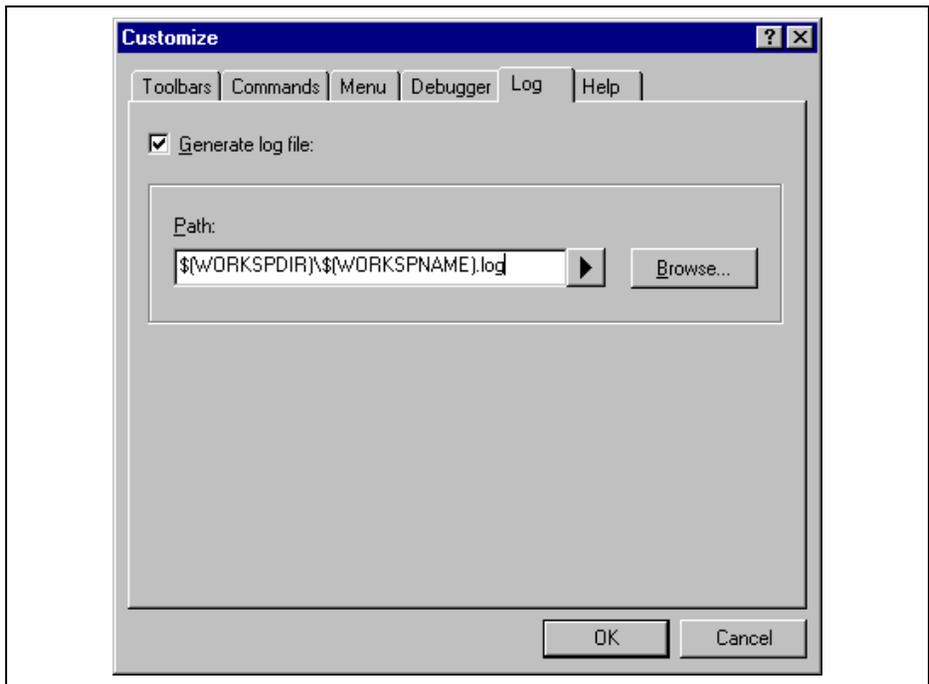


Figure 3.19: Tools Customize Dialog Log Tab

3.8 Changing Toolchain Version

If two or more versions of the same toolchain are registered in the HEW, you can choose a version of the toolchain on the “Change Toolchain Version” dialog shown in Figure . To invoke the dialog, select [**Tools->Change Toolchain Version...**]. Choose one of the versions from the “Available versions” drop-down list and click the “OK” button to enforce your choice.

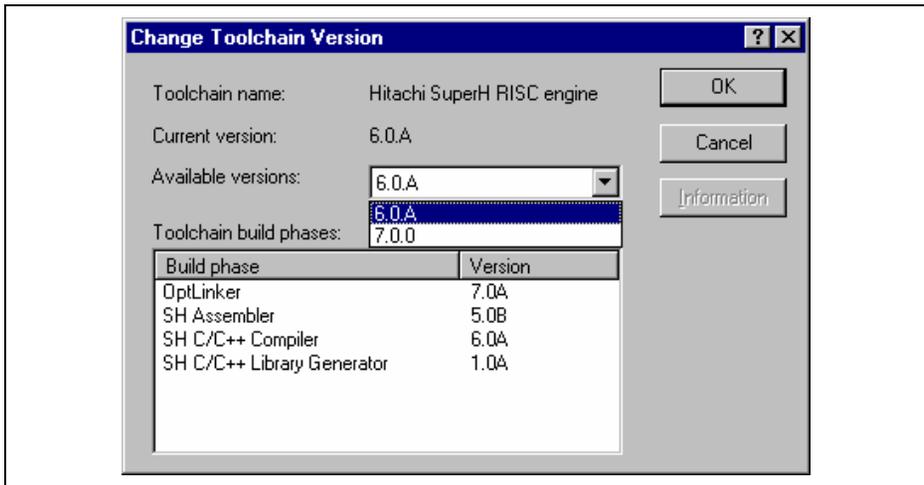


Figure 3.20: Change Toolchain Version Dialog

To show information of toolchain components select a tool from the “Toolchain build phases” list on the “Change Toolchain Version” dialog and click the “Information” button. Then a tool information dialog (figure 3.21) will show you the information of the tool. Click the “Close” button to close the dialog.

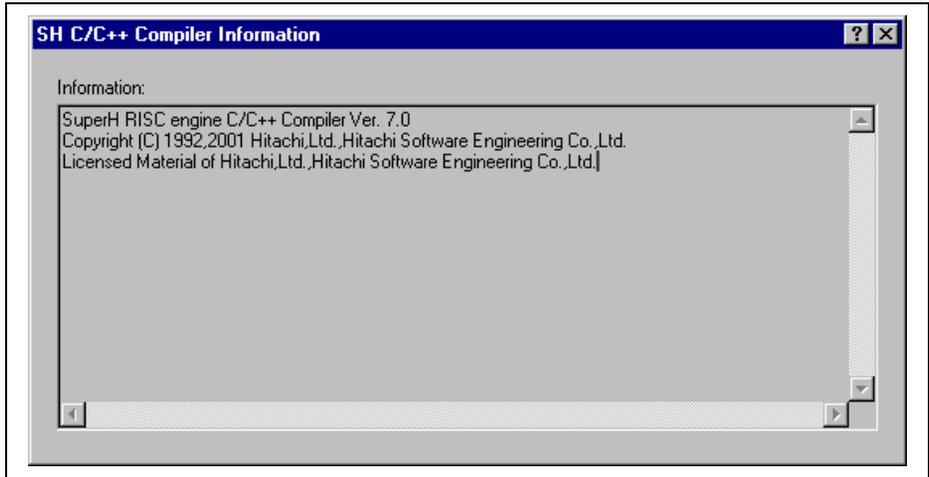


Figure 3.21: Toolchain Information Dialog

3.9 Using an External Debugger

The High-performance Embedded Workshop can launch an external debugger tool. If you want to use another debugger then you must add it to the Tools menu (as described in chapter 6, “*Customizing the Environment*”).

The “Debugger” tab of the “Customize” dialog (figure 3.22) is where the HDI-related information is configured. You may wish to use an older version of the debugger if certain targets are not currently supported in the new environment. Invoke it by selecting [**T**ools->**C**ustomize...] and then selecting the “Debugger” tab.

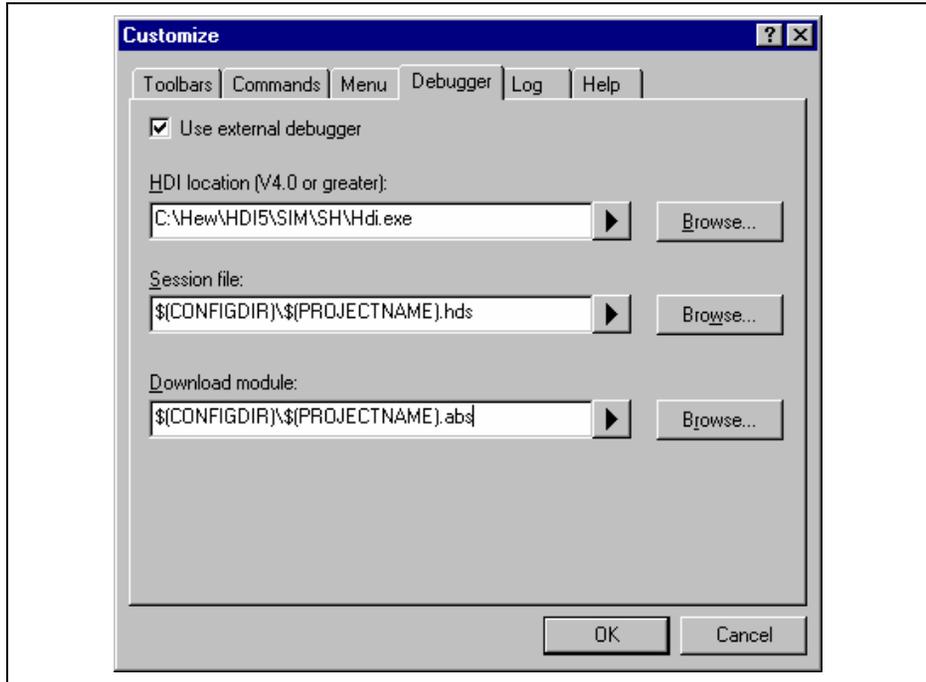


Figure 3.22: Customize Dialog Debugger Tab

When an external debugger is used, check 'Use external debugger' and then set the following items. Firstly, the location of the HDI executable must be specified. This must be version 4.0 or greater otherwise the behavior is not guaranteed. The second item of data is the session file. This tells HDI which session to load when it is launched. Finally, the location of the download module is required. This allows the HEW to automatically switch to HDI when the download module changes after a build.

Click the “Launch External Debugger” toolbar button to invoke HDI with the specified session file:



After a build, if the download module has been updated, the HEW will switch back to HDI to enable immediate debugging. Whilst using HDI, double clicking in any source window will switch back to the HEW with the source file open at the line which was double clicked.

3.10 Generating a Makefile

The HEW allows you to generate a makefile, which can be used to build parts of your workspace without HEW. This is particularly useful if you want to send a project to a user who does not have the HEW or if you want to version control an entire build, including the make components.

➤ To generate a makefile:

1. Ensure that the project, which you want to generate a makefile for, is the current project.
2. Ensure that the build configuration that you want to build the project with is the current configuration.
3. Select **[Build>Generate Makefile]**.
4. Once this menu has been selected a dialog is displayed which asks the user what parts of the workspace need to be added to the make file. (See figure 3.23.)
5. Select the radio button which is relevant for your make file and then click OK.

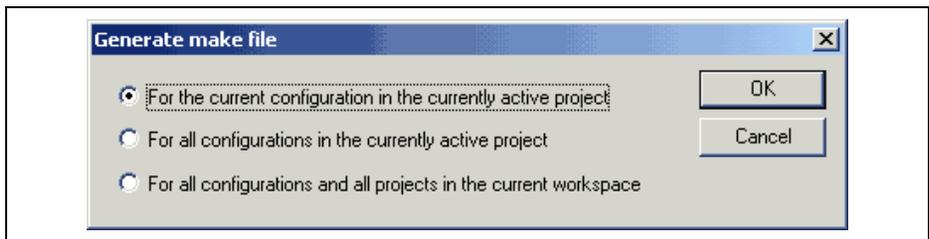


Figure 3.23: Generate makefile Dialog

The HEW will create a subdirectory “make” within the current workspace directory and then generate the makefile into it. It is named after the selection, with a .mak extension for example the current project and configuration (e.g. project_debug.mak). The executable HMAKE.EXE, located in the HEW installation directory, is provided for you to execute the makefiles generated by the HEW. It is not intended to execute makefiles, which have been user modified.

➤ To execute a makefile:

High-performance Embedded Workshop User Manual

1. Open a command window and change to the “make” directory where the makefile was generated.
2. Execute HMAKE. Its command line is HMAKE.EXE <makefile>.

Note: The degree portability of a generated makefile is entirely dependent upon how portable the project itself is. For example, any compiler options, which include full paths to an output directory or include file directory, will mean that, when given to another user with a different installation, the build will probably fail. In general use placeholders wherever possible – using a full, specific path should be avoided when possible.

4. Using the Editor

This chapter describes how to use the editor that is provided with the High-performance Embedded Workshop.

4.1 The Editor Window

The editor window (figure 4.1) contains the file windows that are being viewed or edited. Only one window is active at anytime. This window is called the active window (or current window) and its title bar will appear a different color from that of the others (“dbstct.c” is the active window in figure 4.1). All text operations such as typing, pasting text and so forth only affect the active window. To switch to another source file window (i.e. to make some other window the active window) there are a number of methods:

- Click on it if it is visible.
- Press **CTRL+TAB** to cycle through the windows one after another.
- Select the window by name from the “Window” menu.
- Select its tab at the bottom of the editor window.

When a file has been edited, an asterisk (*) is appended to the window’s title bar. The asterisk remains there until the file is saved. The asterisk is also removed if all of the edited changes are undone in the current window.

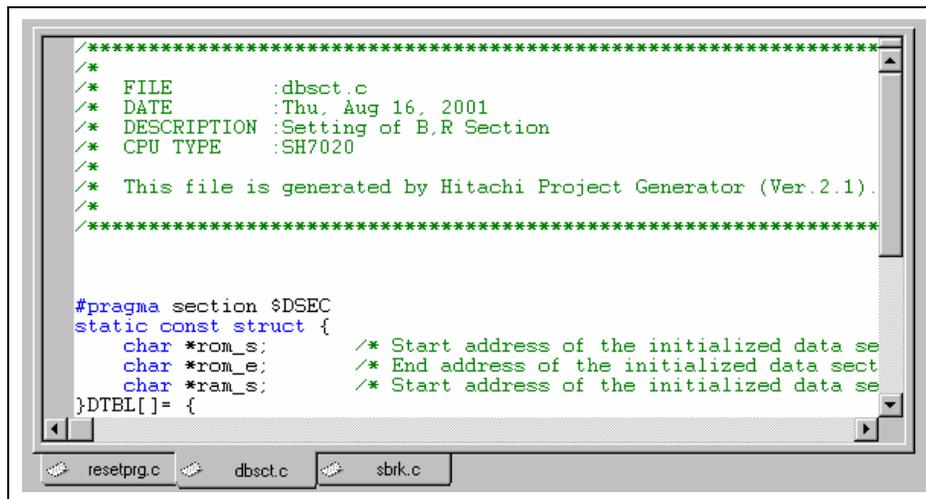


Figure 4.1: Editor Window

4.2 Working with Multiple Files

The file area is where you will work with the files of your project. The editor allows you to have many files open at one time, to switch between them, to arrange them in different configurations and to edit them in whichever order you want to. The operations that you can perform upon the windows are typical of most Windows® applications and they can be found under the [**Window**] menu:

- [**Window->Cascade**]
Arrange all open windows so that they overlap, with the top left of each window visible.
- [**Window->Tile Horizontally**]
Arrange all open windows in sequence (horizontally) so that they occupy the entire editor window with no overlapping edges.
- [**Window->Tile Vertically**]
Arrange all open windows in sequence (vertically) so that they occupy the entire editor window with no overlapping edges.
- [**Window->Arrange Icons**]
Line up all minimized windows at the bottom of the editor window.
- [**Window->Close All**]
Close all open editor windows.

The files within the editor can be displayed in a “notebook” style. This means that each file has a separate tab associated with it to aid in navigating between files.

- ☞ To show files in notebook:
1. Select [**Tools->Options...**]. The “Tools Options” dialog box will be displayed. Select the “Editor” tab.
 2. Set the “Show files in notebook” check box as appropriate.
 3. Click “OK” for the new settings to take effect.

4.2.1 The Editor Toolbars

The editor has four related toolbars: Editor, Search, Bookmarks and Templates. They provide a shortcut to the functions of the editor, which you will use most often. The following sections describe each buttons function.

4.2.2 Editor Toolbar Buttons



New File

The new file button creates a new source file window with a default name. When you save the file, you can specify your own filename.



Open File

Click this button if you want to open a file. It invokes a standard file chooser - select the file which you want to open and then click “Open”.



Save File

Saves the active source file.



Save All Files

Saves all of the files in the editor.



Print File

To print the contents of the current window, click this button.



Cut

Clicking this button will remove the current text selection and place a copy of it onto the Windows® clipboard (it can be pasted back to a file with a paste operation).



Copy

This button allows you to copy the current text selection into the Windows® clipboard.



Paste

The paste button copies the contents of the clipboard into the active window at the position of the insertion cursor.



Find

Click this button if you want to find a certain text string in the current file. It invokes a find dialog box where you can specify the search parameters.



Find in Files

To search several files for a text string then click this button. All find results are displayed in the “Find in Files” tab of the “Output” window. For further information, refer to the “*Searching and Navigating Through Files*” section later in this chapter.



Match Braces

The match braces button highlights text between braces of type { }, [] and (). This is particularly useful when attempting to find out the structure of C/C++ code blocks which are opened with { and closed with }. To use it, select the open brace to match from, or place the cursor before it, and then click this button. For further information on brace matching, refer to the “Brace Matching” section later in this chapter.



Insert Template

To insert a pre-defined template at the current cursor position, click this toolbar button. The “Insert Template” dialog box will be invoked. Select a template name and then click OK. For further information on templates, refer to the “*Templates*” section later in this chapter.



Toggle Bookmark

The High-performance Embedded Workshop editor provides standard bookmark capabilities. To set a bookmark, select the line to mark and click this button (a green mark will then appear in the blank on the left side of the editor window). To remove a bookmark, select the line to remove a bookmark and click this button (the mark in the blank on the left side of the editor window will disappear). For further information on bookmarks, refer to the “*Bookmarks*” section later in this chapter.

4.2.3 Search Toolbar Buttons



Find in Files

To search several files for a text string then click this button. All find results are displayed in the “Find in Files” tab of the “Output” window. For further information, refer to the “*Searching and Navigating Through Files*” section later in this chapter.



Find

Click this button if you want to find a certain text string in the current file. It invokes a find dialog box where you can specify the search parameters.



Find Next

Finds the next occurrence of the current search string.



Find Previous

Finds the previous occurrence of the current search string.

4.2.4 Bookmarks Toolbar Buttons



Toggle Bookmarks

Sets a bookmark at the current line or clears a bookmark at the current line.



Next Bookmark

Jumps to the next bookmark in the current file from the current line.



Previous Bookmark

Jumps to the previous bookmark in the current file from the current line.



Clear All Bookmarks

Clears all bookmarks in the current file.

4.2.5 Templates Toolbar Buttons



Define Template

Specify template text for subsequent insertion.



Insert Template

Insert the template selected in the drop-down list at the current cursor position.

4.3 Standard File Operations

4.3.1 Creating a New File

☞ To create a new editing window:

Select [**File->New**] or click the new file toolbar button () or press **CTRL+N**.

The window will be given an arbitrary name by default. You can provide a new name when you save the file.

4.3.2 Saving a File

☞ To save the contents of an editing window:

1. Ensure that the window, whose contents you want to save, is the active window.
2. Select [**File->Save**] or click the save file toolbar button () or press **CTRL+S**.
3. If the file has not been saved before, a file save dialog box will be displayed. Enter a filename, specify a directory and then click OK to create the file with the name given, in the directory specified.
4. If the file has been saved before, then the file will be updated (no dialog box will be displayed).

☞ To save the contents of an editing window under a new name:

1. Ensure that the window, whose contents you want to save, is the active window.
2. Select [**File->Save As...**].
3. A file save dialog box will be displayed. Enter a filename, specify a directory and then click OK to create the file with the name given, in the directory specified.

4.3.3 Saving all Files

- To save the contents of every open editor window:
 1. Select [**File->Save All**] or click the save all files toolbar button .
 2. If any of the files has not been saved before, a file save dialog box will be displayed. Enter a filename, specify a directory and then click OK to create the file with the name given, in the directory specified.
 3. If any of the files have been saved before, then the file will be updated (no dialog box will be displayed).

4.3.4 Opening a File

- To open a file:
 1. Select [**File->Open...**] or click the open file toolbar button  or press **CTRL+O**.
 2. An open file dialog box will be displayed. Use the directory browser (on the right) to navigate to the directory in which the file you want to open is located. Use the “Files of type” combo box to select the type of file you want to open (or set it to “All Files (*.*)” to see every file in a directory).
 3. Once you have located the file select it and click “Open”.

The High-performance Embedded Workshop keeps track of the last five files that you have opened and adds them to the file menu under the “Recent Files” sub-menu. This gives you a shortcut to opening files which you have used recently.

- To open a recently used file:

Select the [**File->Recent Files**] menu option and from this sub-menu select the desired file.

You can also open a file via the “Projects” tab of the “Workspace” window. Either double click the file you want to open or select it, click the right mouse button (to invoke a pop-up menu) and then choose the [**Open <file>**] menu option (where <file> is the name of the file selected).

4.3.5 Closing Files

- To close individual files select one of the following methods:
 - Double click on the editor window's system menu (located at the top left of each window when not maximized).
 - Click on the editor window's system menu (located at the top left of each window when not maximized) and select the "Close" menu option.
 - Ensure that the window that you want to close is the active window and then press **CTRL+F4**.
 - Ensure that the window that you want to close is the active window and then select [**File->Close**].
 - Click on the close button (located at the top right of each window when not maximized).
- To close all windows at once:
Select [**Window->Close All**].

4.4 Editing a File

The High-performance Embedded Workshop editor supports standard editing functionality. This is available through the usual methods (i.e. the menu, toolbar and keyboard shortcuts) and is additionally supported via a pop-up menu (or local menu) that is local to each editor window. To invoke it, place the pointer in an open window and click the right mouse button. Table 4.1 outlines the basic operations that are provided by the editor.

Table 4.1 Basic Editing Operations

Operation	Effect	Action
Cut	Removes highlighted text and places it on the Windows® clipboard	Click the cut toolbar button Select [Edit->Cut] Select [Cut] - local menu Press CTRL+X
Copy	Places a copy of the highlighted text into the Windows® clipboard	Click the copy toolbar button Select [Edit->Copy] Select [Copy] - local menu Press CTRL+C
Paste	Copies the contents of the Windows® clipboard into the active window at the position of the insertion cursor	Click the paste toolbar button Select [Edit->Paste] Select [Paste] - local menu Press CTRL+V
Delete	Removes highlighted text (it is not copied to the Windows® clipboard)	Select [Edit->Clear] Select [Clear] - local menu Press Delete
Select All	Selects (i.e. highlights) the entire contents of the active window	Select [Edit->Select All] Select [Select All] - local menu
Undo	Reverses the last editing operation	Select [Edit->Undo] Select [Undo] - local menu Press CTRL+Z
Redo	Repeats the last “undone” editing operation	Select [Edit->Redo] Select [Redo] - local menu Press CTRL+Y

4.5 Searching and Navigating through Files

The High-performance Embedded Workshop editor provides find, replace and file navigation functionality. The following three sections detail how to use these features.

4.5.1 Finding Text

- ☞ To search for text in the current file:
 1. Ensure that the window, whose contents you want to search, is the active window.
 2. Position the insertion cursor at the point from which you want to start your search.
 3. Select [Edit->Find...], press **CTRL+F**, select [Find...] from the editor window's local menu or click the find toolbar button (🔍). The "Find" dialog box will be displayed (figure 4.2).



Figure 4.2: Find Dialog

4. Enter the text that you want to search for into the "Find what" field, or select a previous search string from the drop-down list box. If you select text before invoking the find operation, the selected text will be automatically placed into the "Find what" field.
5. If you would like to search for character string as a whole word then check the "Match whole word only" check box. When this option is not selected, the search will be for any string that is matched by the search string.
6. If you would like your search to be case sensitive (i.e. to distinguish between upper and lower case letters) then check the "Match case" check box.
7. If your search string uses regular expressions then check the "Regular expressions" check box. Refer to Appendix B, "*Regular Expressions*" for further information.
8. The "Direction" radio buttons allow you to select the direction of the search. Selecting "Down" means that the search will be performed from the insertion cursor towards the bottom of the file. Selecting "Up" means that the search will be performed from the insertion cursor towards the top of the file.

9. Click the “Find Next” button to begin the search. Click “Cancel” to stop the find action.

The High-performance Embedded Workshop editor also allows you to search for a string across many files.

4.5.2 Finding Text in Multiple Files

☞ To search for text in many files:

1. Select [**Edit->Find in Files...**], select [**Find in Files...**] from the editor window's local menu or click the find in files toolbar button (). The "Find in Files" dialog box will be displayed (figure 4.3).

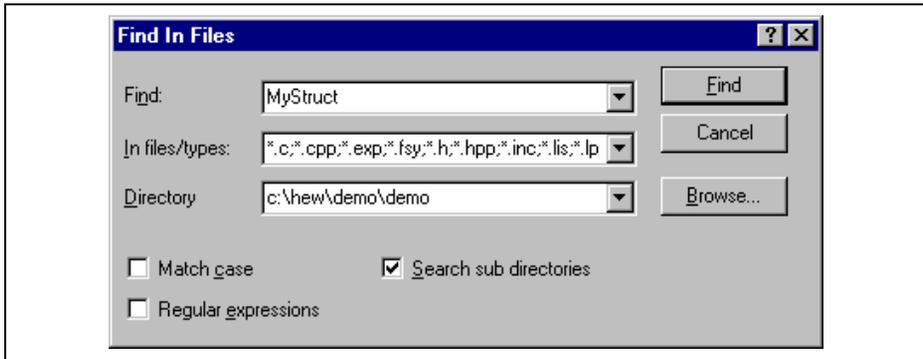


Figure 4.3: Find in Files Dialog

2. Enter the text that you want to search for into the "Find" field, or select a previous search string from the drop-down list box. If you select text before invoking the find operation, the selected text will be automatically placed into the "Find" field.
3. Enter the file extensions of the files you would like to search into the "In files/types" field. If several extensions are specified be sure to separate them with a comma (e.g. *.c,*.h).
4. Enter the directory in which you would like to search files into the "Directory" field. Alternatively you may browse to the desired directory graphically if you click the "Browse..." button.
5. If you would like to search the directory specified and all directories below it then check the "Search sub directories" check box. If you just want to search the single directory specified in the "Directory" field then ensure that this check box is not checked.
6. If you would like to search for character string as a whole word then check the "Match case" check box. When this option is not selected, the search will be for any string that is matched by the search string.
7. If you would like your search to be case sensitive (i.e. to distinguish between upper and lower case letters) then check the "Match case" check box.
8. Click "Find" to begin the search. Any matches found will be displayed in the "Find in Files" tab of the "Output" window. To jump to an instance of the string, double click on the desired entry in the "Output" window.

4.5.3 Replacing Text

Replacing text is similar to finding text, as discussed in the previous section. The difference is that when the text is found you have the option to replace it with other text.

☞ To replace text in a file:

1. Ensure that the window, whose contents you want to replace, is the active window.
2. Position the insertion cursor at the point from which you want to start your search.
3. Select [**Edit->Replace...**], press **CTRL+H** or select [**Replace...**] from the editor window's local menu. A replace dialog box will be displayed (figure 4.4).
4. Enter the text that you want to search for into the "Find what" field, or select a previous search string from the drop-down list box. If you select text before invoking the replace operation, the selected text will be automatically placed into the "Find what" field.
5. Enter the text that you want to replace the search string with into the "Replace with" field, or select a previous replace string from the drop-down list box.

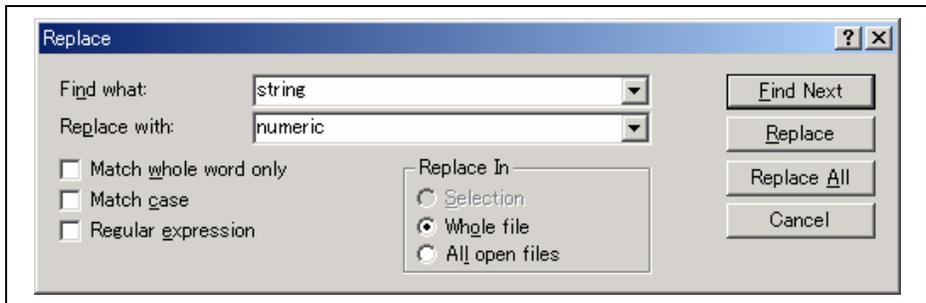


Figure 4.4: Replace Dialog

6. If you would like to search for character string as a whole word then check the "Match whole word only" check box. When this option is not selected, the search will be for any string that is matched by the search string.
7. If you would like your search to be case sensitive (i.e. to distinguish between upper and lower case letters) then check the "Match case" check box.
8. If your search string uses regular expressions then check the "Regular expressions" check box. Refer to appendix B, "*Regular Expressions*" for further information.
9. If you clicked "Find Next", the editor will search for the first occurrence of the search string. Click "Replace" if you want to replace it. Click "Replace All" to replace all occurrences or click "Cancel" to stop the replace action. If you select "Selection" in "Replace In", selected range of the text is replaced. If you select "whole file", the whole

files are replaced. If you select all open files, all files that are currently open in the editor have the replace operation carried out on them.

4.5.4 Jumping to a Specified Line

- ☞ To jump to a line in a file:
 1. Ensure that the window, whose contents you want to replace, is the active window.
 2. Select [**Edit->Goto Line...**], press **CTRL+G**, or select [**Goto Line...**] from the editor window's local menu. A goto line dialog box will be displayed (figure 4.5).
 3. Enter into the dialog box the number of the line that you want to go to, and then click "OK".
 4. The insertion cursor will be placed at the start of the line number specified.

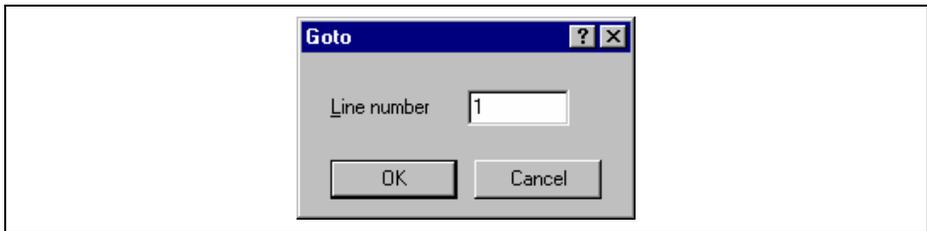


Figure 4.5: Goto Dialog

4.6 Bookmarks

When working with many large files at a time, it can become difficult to locate specific lines or areas of interest. Bookmarks enable you to specify lines that you want to jump back to at a subsequent time. One example of its use is in a large C file where you may want to set a bookmark on each function definition. Once a bookmark has been set, it exists until it is removed or the file is closed.

- ☞ To set a bookmark:
 1. Place the insertion cursor on the line to mark.
 2. Select [**Edit->Bookmarks->Toggle Bookmark**], press **CTRL+F2**, select [**Bookmarks ->Toggle Bookmark**] from the local menu or click the toggle bookmark toolbar button .
 3. A green mark appears in the blank on the left side of the line to indicate the presence of an active bookmark.
- ☞ To remove a bookmark:
 1. Place the insertion cursor on the marked line.

2. Select [**Edit->Bookmarks->Toggle Bookmark**], press **CTRL+F2**, select [**Bookmarks->Toggle Bookmark**] from the local menu or click the toggle bookmark toolbar button .
3. The mark will be removed and the line will return to normal text.

- ☞ To jump to the next bookmark in a file:
 1. Ensure that the insertion cursor is somewhere within the file to be searched.
 2. Select [**Edit->Bookmarks->Next Bookmark**], press **F2** or select [**Bookmarks->Next Bookmark**] from the local menu or click the next bookmark toolbar button ()
- ☞ To jump to the previous bookmark in a file:
 1. Ensure that the insertion cursor is somewhere within the file to be searched.
 2. Select [**Edit->Bookmarks->Previous Bookmark**], press **SHIFT+F2** or select [**Bookmarks->Previous Bookmark**] from the local menu or click the previous bookmark toolbar button ()
- ☞ To remove all bookmarks in a file:
 1. Ensure that the window, whose bookmarks you want to remove is the active window.
 2. Select [**Edit->Bookmarks->Clear All Bookmarks**] or select [**Bookmarks->Clear All Bookmarks**] from the local menu or click the clear all bookmarks toolbar button ()

4.7 Printing a File

- ☞ To print a file:
 1. Ensure that the window, whose contents you want to print, is the active window.
 2. Select [**File->Print...**], or click the print toolbar button () or press **CTRL+P**.

4.8 Configuring Text Layout

The following sections detail how to set-up the layout of the text within the editor windows.

4.8.1 Page Set-up

When you print a file from the High-performance Embedded Workshop editor, the settings in the print dialog box affect the way in which the file is printed (e.g. double or single sided). Control over how the text is formatted on the page can also be controlled via the page set-up option. This allows you to specify the margins (top, bottom, left and right) of your printouts. It is often necessary to set this because some printers cannot print to the edges of an A4 page. Furthermore, some users have their own layout requirements (e.g. a large left hand margin so that code can be placed in an A4 binder).

➡ To set-up the page margins:

1. Select [**File->Page Setup...**]. The “Page Setup” dialog will be invoked (figure 4.6).
2. Enter into the edit fields the margins required (set the “inch” or “mm” radio buttons to set the measurements).
3. Click “OK” for the new settings to take effect.

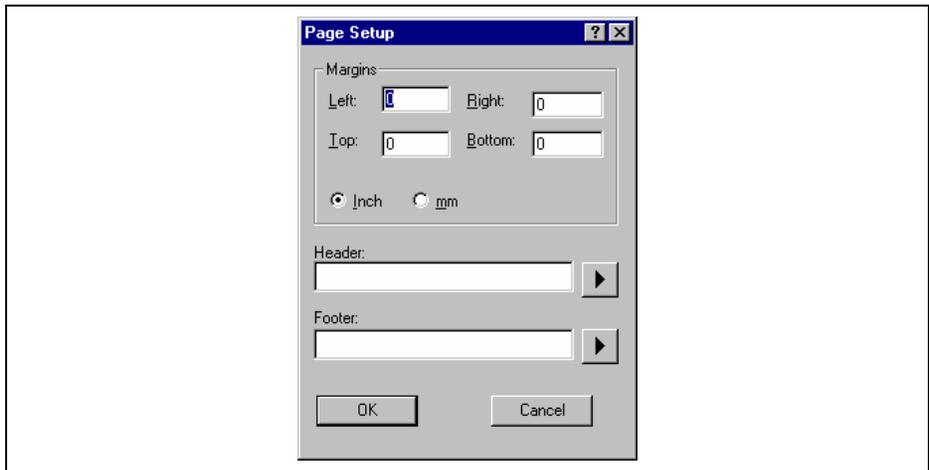


Figure 4.6: Page Setup Dialog

- ☞ To set-up the page header and footers:
 1. Select [**File->Page Setup...**]. The “Page Setup” dialog will be invoked (figure 4.6).
 2. Enter into the header and footer edit fields the text required to be displayed. All normal placeholders are available along with page numbering, text justification and date fields. These are all expanded before the page is to be printed.
 3. Click “OK” for the new settings to take effect.

4.8.2 Changing Tabs

➡ To change tab size:

1. Select [**Tools->Options...**]. The “Options” dialog will be displayed. Select the “Editor” tab (figure 4.7)).
2. Enter into the “Tab size” field the number of desired tabs.
3. Click “OK” for the tab setting specified to take effect.

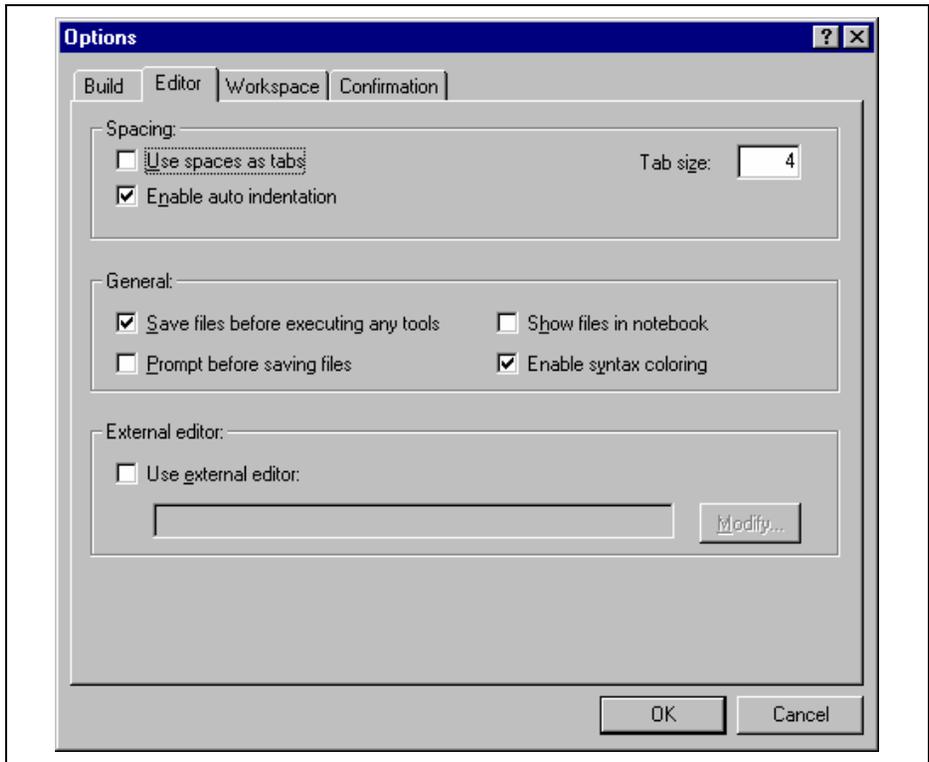


Figure 4.7: Options Dialog Editor Tab

When a **TAB** key is pressed in the editor a tab character is usually stored in the file. However, sometimes it is preferable to store spaces instead. The representation of tab characters can be controlled via the “Options” dialog.

➡ To use spaces as tabs:

1. Select [**Tools->Options...**]. The “Options” dialog will be displayed. Select the “Editor” tab (figure 4.7).
2. Set the “Use spaces as tabs” check box as appropriate.
3. Click “OK” for the tab setting specified to take effect.

4.8.3 Auto Indentation

When you press return in a standard editor the insertion cursor will move to the next line down, at the first column (i.e. against the left hand side of a window). Auto Indentation is a feature which, when return is pressed, places the insertion cursor on the next line (as before) but under the first non-white space character of the previous line. This enables you to type neat C/C++ or assembler code faster as you don’t have to type leading spaces or tabs yourself.

Figure 4.8 illustrates two examples. The first (i) shows the effect of pressing return when the auto indentation feature is disabled - the insertion cursor returns to the left-hand side of the window on the next line. When the line “int z=20” is typed, it is not aligned with the previous two lines. The second example (ii) shows the effect of pressing return when auto indentation is enabled - the insertion cursor drops underneath the “i” of the previous line. Now, when the line “int z=20” is typed, it is automatically aligned (i.e. automatically indented).

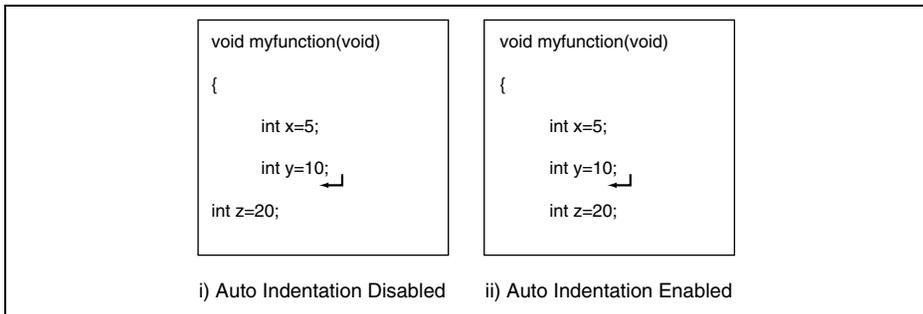


Figure 4.8: Effect of Auto Indentation

- ☞ To enable/disable Auto Indentation:
1. Select [**Tools->Options...**]. The “Options” dialog will be displayed. Select the “Editor” tab (figure 4.7).
 2. Set the “Enable auto indentation” check box accordingly.
 3. Click “OK” for the setting of the auto indentation check box to take effect.

4.9 Splitting a Window

The High-performance Embedded Workshop editor allows you to split a text window into two. Figure 4.9 shows the split bar button which is located just underneath the maximize button at the top right hand corner of any text window.

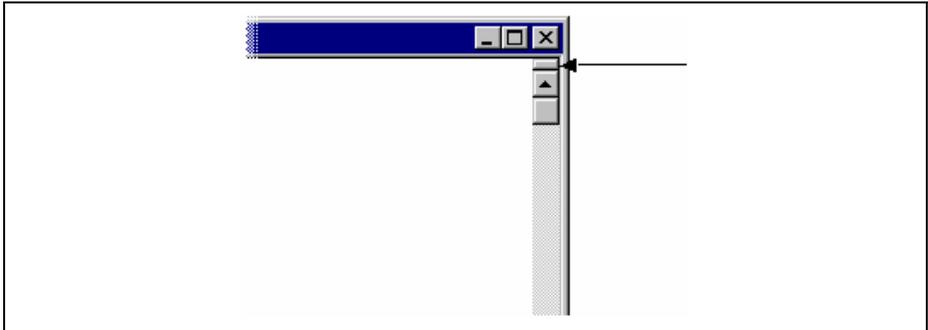


Figure 4.9: Split Bar Button

- To split a window:
Double click on the split bar button to split the window in half or click on the split bar button, keep the button pressed, move the mouse down and then release the mouse button at the point you want to split the window.
- To adjust the position of the split bar:
Click on the split bar itself, keep the button pressed then move the bar to the new position and then release the button.
- To remove the split bar:
Double click on the split bar or move the split bar to the top or bottom of the window.

4.10 Configuring Text

The following sections detail how to change the appearance of the text displayed in the editor windows.

4.10.1 Changing the Editor Font

The High-performance Embedded Workshop allows you to specify the font to be used in its internal editor. All editor windows, regardless of the file type, use the same font.

☞ To change the editor font:

1. Select [**Tools->Format Views...**]. The “Format Views” dialog will be displayed. Select the Source icon in the tree (figure 4.10).
2. Select the desired font from the “Font” list.
3. Select the size of the font from the “Size” list.
4. Click “OK” to confirm the new editor settings.

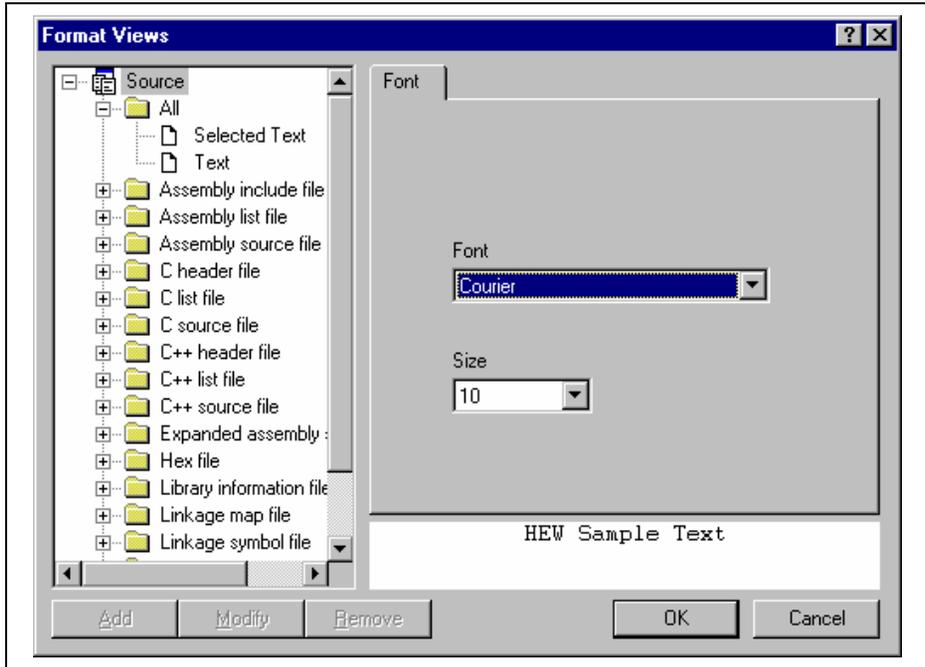


Figure 4.10: Format Views Dialog Font Tab

4.11 Syntax Coloring

To enhance code readability, the HEW editor can display specific strings (i.e. keywords) in different colors. For instance, C source code comments could be shown in green and C types (e.g. int) could be shown in blue.

The coloring method used can be specified on a file group by file group basis. For example, you can define different color schemes for a C source files, text files, map files or even your own files.

➤ To change existing colors:

1. Select [Tools->Format Views...]. The “Format Views” dialog will be displayed.
2. Select the item underneath the icon in the tree you wish to modify the colour for. This should be the file type (e.g. C source file) and correct keyword group (e.g. identifier or pre-processor).
3. Select the “Colour” tab.

4. Modify the “Foreground” and “Background” color lists as desired. The color “System” refers to the current window foreground and background settings in control panel.
5. Click “OK” for the new colors to take effect.

☞ To create new keyword groups:

1. Select [**Tools->Format Views...**]. The “Format Views” dialog will be displayed.
2. Select the file type in the tree to which you wish to add the new keyword group.
3. Click “Add...” underneath the tree. The “Add Category” dialog box will be displayed (figure 4.11). Enter the name of the keyword group in the “Category Title” field, then click “OK” to create the new keyword group.

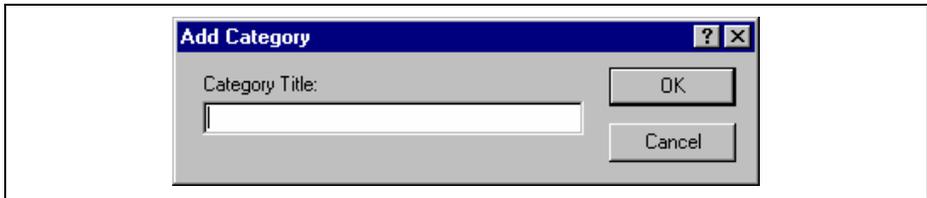


Figure 4.11: Add Category Dialog

☞ To create new keywords:

1. Select [**Tools->Format Views...**]. The “Format Views” dialog will be displayed.
2. Select the item underneath the source view icon in the tree you wish to modify the syntax highlighting for. This should be the file type (e.g. C source file) and correct keyword group (e.g. identifier or pre-processor).
3. Select the “Keywords” tab (figure 4.12).

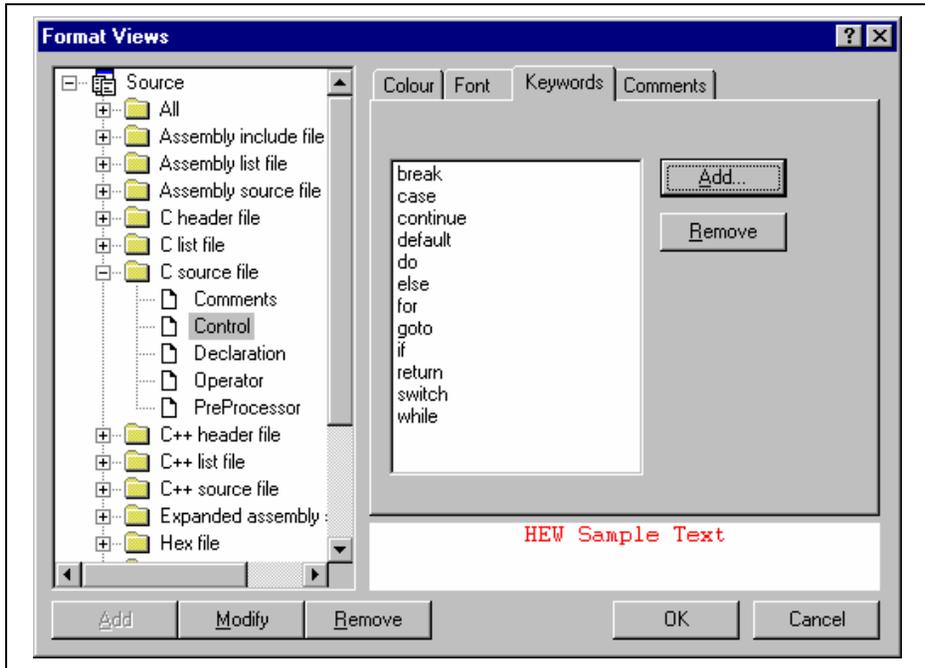


Figure 4.12: Format Views Dialog Keywords Tab

4. Click the “Add...” button to add a keyword. Then the “Add Keyword” dialog (figure 4.13) will be launched. Specify a keyword in the “Keyword” field and click “OK” to close the dialog. To remove a keyword, select the keyword and click the “Remove” button.

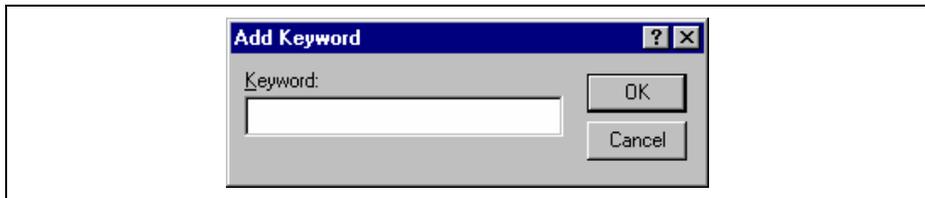


Figure 4.13: Add Keyword Dialog

When you create a new file, syntax coloring will not be active as a new file does not initially have an extension (new files are named arbitrarily by the editor without an extension). In order to activate syntax coloring, you must save the new file with a name, which has one of the above extensions.

- ☞ To disable/enable syntax coloring:
 1. Select [**Tools->Options...**]. The “Options” dialog will be displayed. Select the “Editor” tab (figure 4.7).
 2. Set the “Enable syntax coloring” check box as necessary and then click “OK”.

4.12 Templates

When developing software it is often necessary to enter the same text repeatedly, for instance, when typing a function definition, for loop or a comment block for a function. The High-performance Embedded Workshop editor allows you to specify a block of text (or template) which can be inserted into the currently active editor window. Thus, once a template has been defined, it can be automatically inserted without the need to re-enter it manually.

4.12.1 Defining a Template

➡ To define a template:

1. Select [**Edit->Templates->Define Templates...**], select [**Templates->Define Templates...**] from the local menu, press **CTRL+T** or click on the define template toolbar button (). The dialog shown in figure 4.14 will be displayed.
2. Click “Add”. A dialog is displayed that asks you to enter your chosen template name. This name must be unique otherwise a duplicated template name message will be displayed and the template will not be added.
3. If you want to modify an existing template use the “Template name” drop down menu to select which template you want to modify.
4. Enter the desired text into the “Template text” text area. You can copy text from another editor window and then paste it into this dialog using **CTRL+V**.
5. Enter the following keywords to insert special information when the template is inserted:

Menu Entry	Placeholder	Replaced With
Time	\$(TIME)	Current time
Date as DMY	\$(DATE_DMY)	Current date, in dd/mm/yy form
Date as MDY	\$(DATE_MDY)	Current date, in mm/dd/yy form
Date as YMD	\$(DATE_YMD)	Current date, in yy/mm/dd form
Date as Text	\$(DATE_TEXT)	Current date in text form
Line	\$(LINE)	First line number of template insertion
User	\$(USER)	Current windows user
File	\$(FULLFILE)	Name of the file
Filename	\$(FILE)	Name and full path of the file
Project Name	\$(PROJNAME)	Current project name
Workspace Name	\$(WORKSPNAME)	Workspace name

Cursor position	\$(^)	Insertion cursor – Positions the cursor in this position after template has been inserted
-----------------	-------	---

- Enter the \$(^) character to specify where the insertion cursor is to be placed after the template has been inserted. If this is not specified then the insertion cursor will be placed after the last character in the template (as in a normal paste operation).

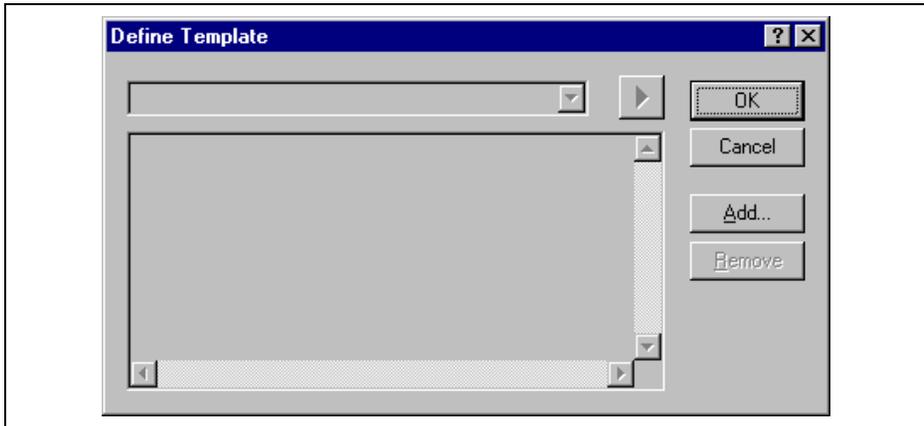


Figure 4.14: Define Template Dialog

4.12.2 Deleting a Template

- ☞ To delete a template:
 - Select [**Edit->Templates->Define Templates...**], select [**Templates->Define Templates...**] from the local menu, press **CTRL+T** or click on the define template bookmark toolbar button (). The dialog shown in figure 4.14 will be displayed.
 - Use the Template name drop down list to select the name of the template you wish to remove and then click the “Remove” button.
 - Clicking “OK” saves the template changes and dismisses the dialog.

4.12.3 Inserting a Template

- ☞ To insert a template:
 - Select a template in the toolbar, then click the insert template toolbar button (), select [**Edit-> Templates->Insert Template...**] or select [**Templates-> Insert Template...**] from

the local menu. The dialog is dismissed and the chosen template is added to the current editor window.

4.12.4 Brace Matching.

Complicated source code can often become unwieldy, especially when blocks of C code are deeply nested within each other or when complex logic statements are expressed within an 'if' clause. To help in such situations, the High-performance Embedded Workshop editor provides a match brace feature which highlights text between braces of type { }, () and [].

➡ To find a matching brace:

1. Either highlight the open brace to match from or place the cursor before it.
2. Click the match braces toolbar button , press **CTRL+B**, select **[Edit->Match Braces]** or select **[Match Braces]** from the local menu.

To check the structure of an entire file, place the cursor at its start and then repeatedly invoke the match brace operation. The editor will successively highlight each pair of braces in turn until there are no more to match.

4.13 Editor Column management

The editor in HEW has the ability to manage columns apart from the main editor column. These can be added and used by debugger feature. You can choose the column to display/undisplay.

```

tutorial.c
#include <no_float.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void main(void);
void sort(long *a);
void change(long *a);
extern void srand(unsigned int);

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

    srand(1);

    printf("### Data Input ###\n");

    for( i=0; i<10; i++ ){
        j = rand();
        if( j < 0){
            j = -j;
        }
        a[i] = j;
        printf("a[%d]=%ld\n",i,a[i]);
    }
    sort(a);
    printf("*** Sorting results ***\n");
    for( i=0; i<10; i++ ){
        printf("a[%d]=%ld\n",i,a[i]);
    }
    min = a[0];
}

```

Figure 4.15: Editor columns

- ☞ To switch off a column in all source files:
 1. Right click on the editor window.
 2. Click the “Define Column Format...” menu item.
 3. The “Global Editor Column States” dialog is displayed.
 4. The “Check status” shows whether the column is enabled or not. If it is checked it is enabled if the check box is gray this means that in some files the column is enabled and in other files it is not.
 5. Click “OK” for the new column settings to take effect.

- To switch off a column in one source files:
1. Right click on the editor window, which you wish to remove a column from, and the editor pop-up is displayed.
 2. Click the Columns menu item and a cascaded menu item appears. Each column is displayed in this pop-up menu. If the column is enabled it has a tick next to its name. Clicking the entry will toggle whether the column is displayed or not.

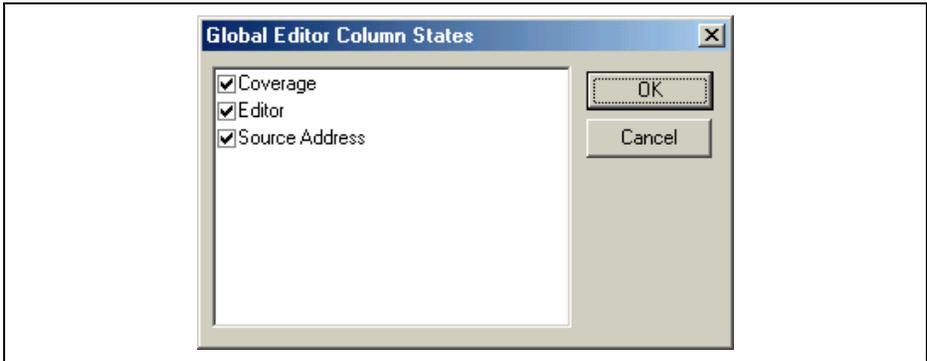


Figure 4.16: Global column state Dialog

5. Tools Administration

You control the components, which can be used by the High-performance Embedded Workshop via the “Tools Administration” dialog (figure 5.1), which is invoked via [**Tools->Administration...**]. Modification of the “Tools Administration” dialog box is only possible when no workspace is open, while only reference is possible when a workspace is open.

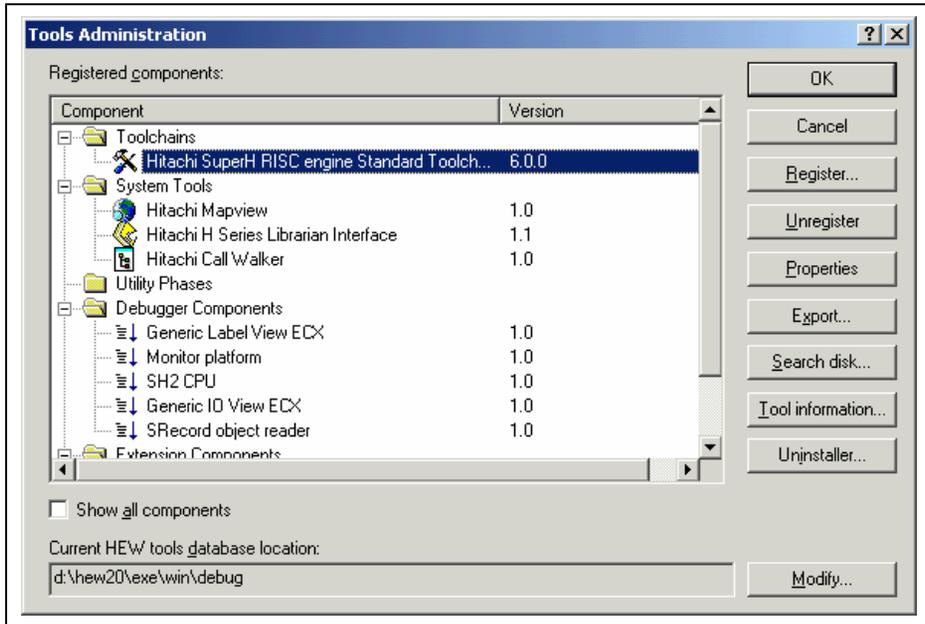


Figure 5.1: Tools Administration Dialog (Example)

There are five standard types of component:

- **Toolchain** - a set of build phases (e.g. compiler, assembler, and linker). These components provide the build capability.
- **System Tool** - an application (.EXE) which can be launched from the “Tools” menu. They are often provided as extra applications, which support the toolchain (e.g. an external debugger like the High-performance Embedded Workshop 2 (HEW2) or an interactive graphical librarian).

- **Utility Phase** - a “ready made” build phase which supports some specific build functionality (e.g. analyze complexity of source code, count lines of source code, etc.). These components provide added functionality to the build that is not toolchain specific.
- **Debugger Component** – a component that supports some specific debugger functionality (e.g. Target platform, Object reader, etc).
- **Extension Component** – a component that provides key functionality in a certain area of the HEW system. These components cannot be unregistered when installed (e.g. The HEW builder, debugger and flash support).

5.1 Tool Locations

The HEW maintains the locations of HEW compatible components automatically as each new tool is installed. After installation, the HEW stores information about the component (including its location) – this is referred to as *registration*. Although initial registration is automatic, during the course of development or if you want to manage the tools being used in your projects more effectively, you may need to register components yourself. The remainder of this chapter discusses registration and how it affects you.

5.2 HEW Registration Files (*.HRF)

When a HEW compatible component (i.e. toolchain, system tool or utility phase) is installed, part of its installation will include a file with the extension **.HRF** (figure 5.2.i). This file, named a “HEW Registration File”, describes the component to the HEW. The process of registration refers to loading a component’s .HRF file into the tools administration dialog (figure 5.2.ii).

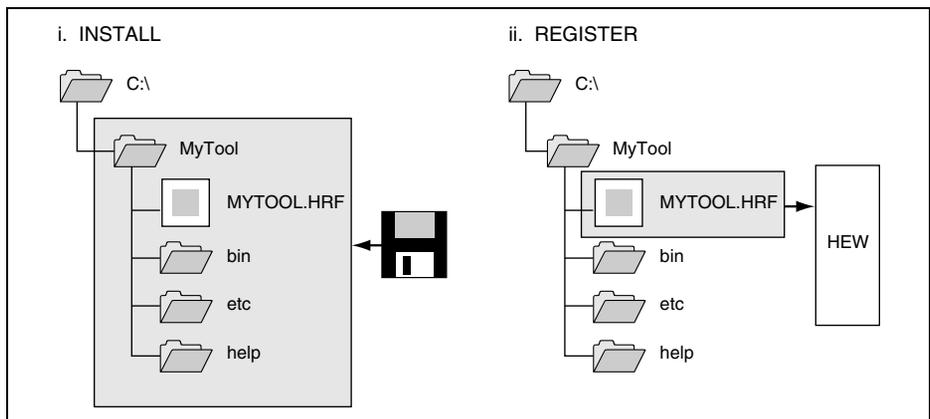


Figure 5.2: HRF File Location and Registration

In order to use a component with HEW it must first be registered. The “Tools Administration” dialog (figure 5.1) shows all currently registered components. To access it, ensure no workspaces are open and then select [**Tools** ->**Administration...**]. If you attempt to access tools administration when there is a workspace open the tools administration dialog is opened but cannot be modified. When HEW is installed by default any new tools are automatically registered.

HEW stores tool information in a tool database file. By default this is created in the HEW application directory, however if you are working in a network environment this directory may be set to another location. It is possible to change the tool directory location.

☞ To change the tools location:

1. Select [**Tools->Administration...**].
2. Click the “Modify” button for the “Current HEW tools database location” field.
3. Select the directory under which the new tool is located, then click “OK”.
4. This will switch the directory and change the tool location to the new directory. It will be necessary to scan for any new tools that may be in this location this is achieved by using the scan disk or register tool functionality.

5.3 Registering Components

The HEW will automatically attempt to register any new components installed since the last time it was invoked. However, in some circumstances you may need to register components yourself.

5.3.1 Searching Drives for Components

In some cases it is useful to search a drive for HEW compatible components. This is especially useful if the HEW installation was deleted or corrupted as it can recreate your tool information instantly.

➤ To search for components:

1. Click the “Search Disk...” button on the “Tools Administration” dialog (figure 5.1). The “Search Disk for Components” dialog will be displayed (figure 5.3).

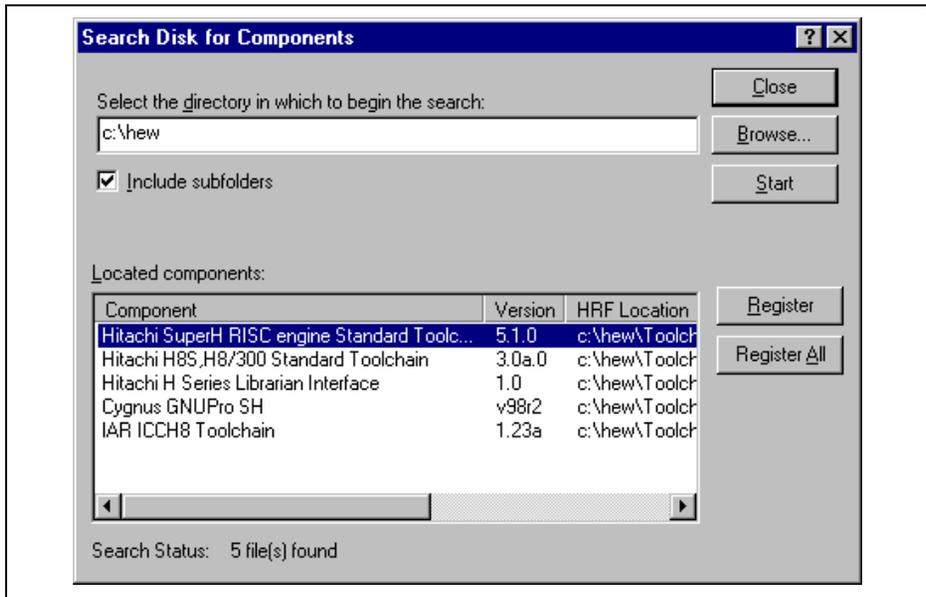


Figure 5.3: Search Disk for Components Dialog

2. Enter the directory in which you would like to search into the top field or browse to it graphically by clicking the “Browse...” button.
3. Check the “Include subfolders” check box if you would like to search the directory specified and all directories below it.

4. Click the “Start” button to begin the search. During the search, the “Start” button will change to a “Stop” button. Click the “Stop” button to halt the search at any time.
5. The results of the search are shown in the “Located components” list. Select a component and click “Register” to register an individual component or click “Register All” to register all located components.
6. Click “Close” to exit the dialog.

5.3.2 Registering a Single Component

The HEW allows you to navigate directly to a single component in order to register it. The HEW Registration File (*.HRF) is located in the root directory of a component's installation.

➡ To register a component:

1. Click the “Register...” button on the “Tools Administration” dialog. A standard file open dialog will be launched with its file filter set to “HEW Registration Files (*.hrf)”.
2. Navigate to the .HRF file of the component you would like to register, select it and then click “Select”.
3. A dialog will be invoked which displays information regarding the selected tool. Click “Register” to confirm that you want to register the tool or click “Close” to abort the operation.

5.4 Unregistering Components

The components, which are registered with the HEW, affect the way in which it behaves. For example, every compatible system tool, which is registered, will be added to the tools menu when a new project is created. Sometimes this may not be desirable. If so, open the “Tools Administration” dialog, select the component from the “Registered components” list and then click the “Unregister” button. A dialog will be invoked which asks you to confirm this action. Click “Yes” to confirm the action.

Note: Unregistering a component does not remove its installation from hard disk. It simply removes the information, which the HEW was storing about that component (i.e. it “disconnects” it from the HEW). The action can be easily reversed at anytime by registering the tool (see above). If you want to remove a component from the hard disk (i.e. uninstall a component) then refer to the section “*Uninstalling Components*” later in this chapter.

5.5 Viewing and Editing Component Properties

To view information regarding a component, select it from the “Registered components” list and then click the “Properties” button. The properties dialog will be displayed with the “General” tab selected (figure 5.4). This tab displays the name, version and location of the selected component. None of the information on this tab is editable.

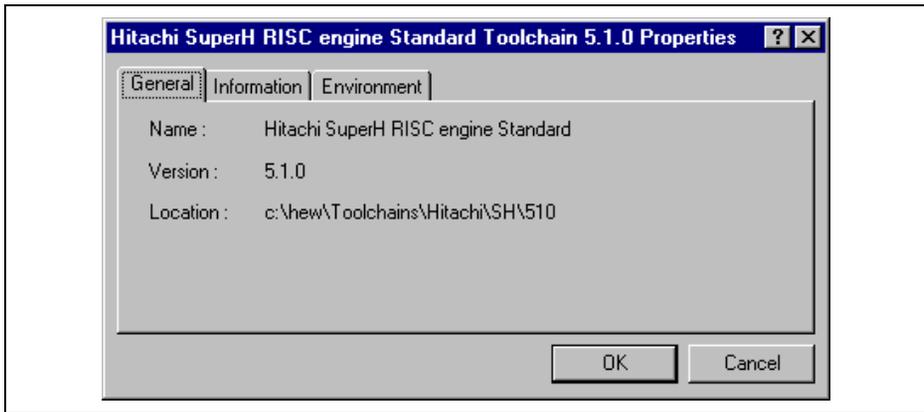


Figure 5.4: Properties Dialog General Tab

Select the “Information” tab to view any information about the component (figure 5.5). This may include copyright information, enhancements, bug fixes, user notes and so on.

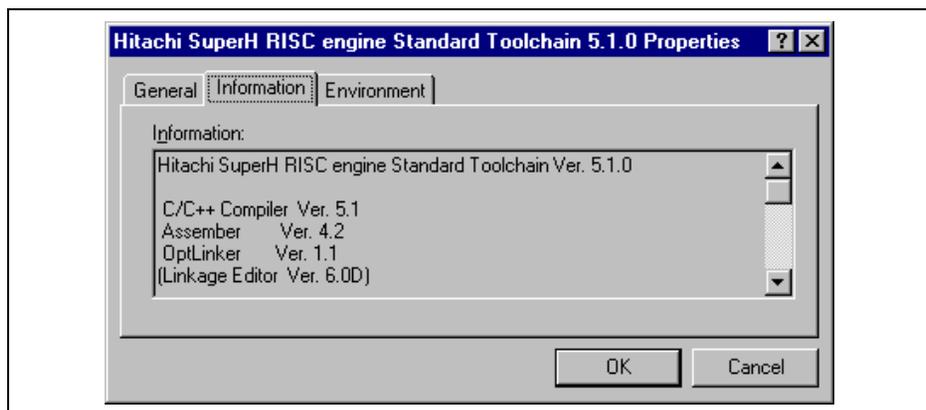


Figure 5.5: Properties Dialog Information Tab

Select the “Environment” tab, if it exists, to view and edit a component’s environment settings (figure 5.6). This dialog is most commonly used to modify the environment of a toolchain.

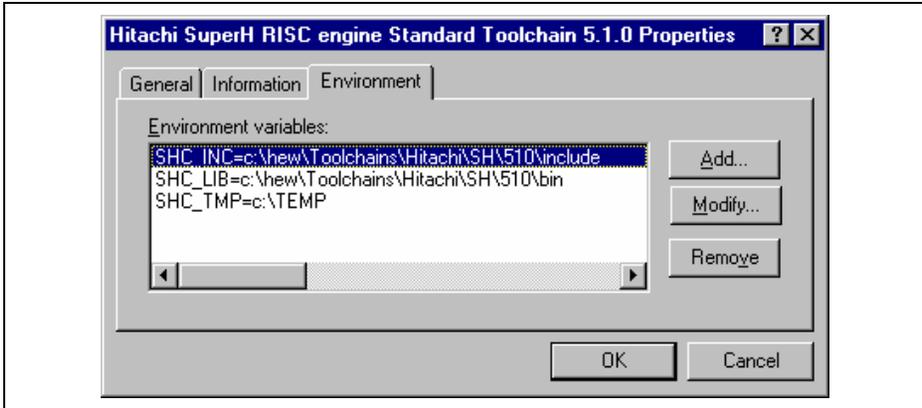


Figure 5.6: Properties Dialog Environment Tab

To add a new environment variable, click the “Add...” button (the dialog shown in figure 5.7 will be invoked). Enter the variable name into the “Variable” field, the variable’s value into the “Value” field and then click “OK” to add the new variable to the “Environment” tab. Placeholder pop-up menus are included to ensure that the environment can be specified as flexibly as possible. For a detailed description of placeholders see appendix C, “Placeholders”.

To modify an environment variable, select the variable that you want to modify from the “Environment” tab and then click the “Modify...” button. Make the required changes to the “Variable” and “Value” fields, and then click “OK” to add the modified variable to the “Environment” tab. To remove an environment variable, select it and then click the “Remove” button.

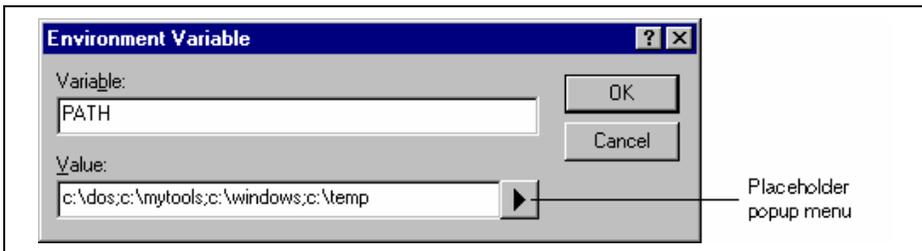


Figure 5.7: Environment Variable Dialog

5.6 Uninstalling Components

The HEW provides a built in uninstaller method, which can remove unregistered components.

☞ To uninstall a component:

1. Select **[Tools->Administration...]**.
2. Click on the uninstaller button. The “Uninstall HEW Tool” dialog is invoked (figure 5.8).

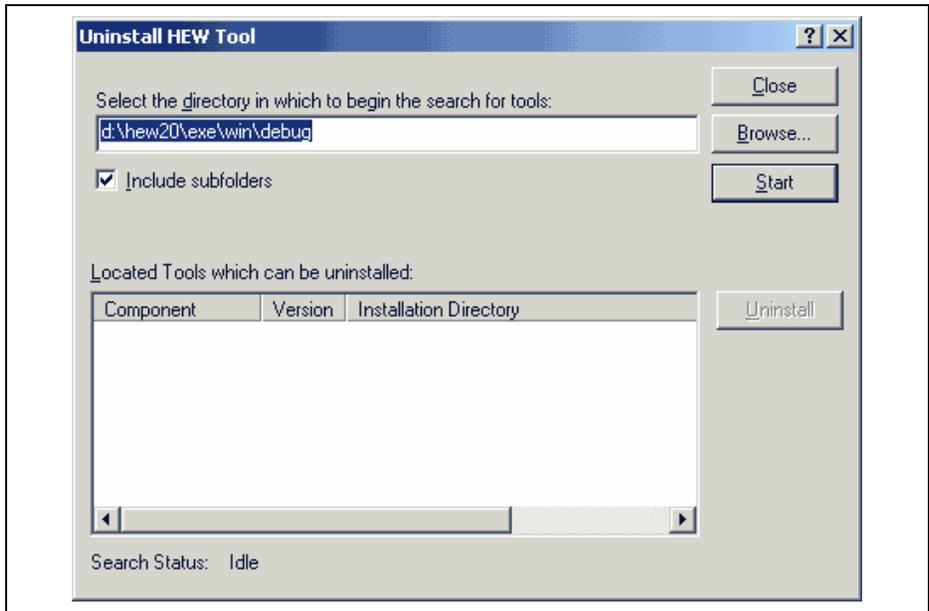


Figure 5.8: Uninstall HEW Tool

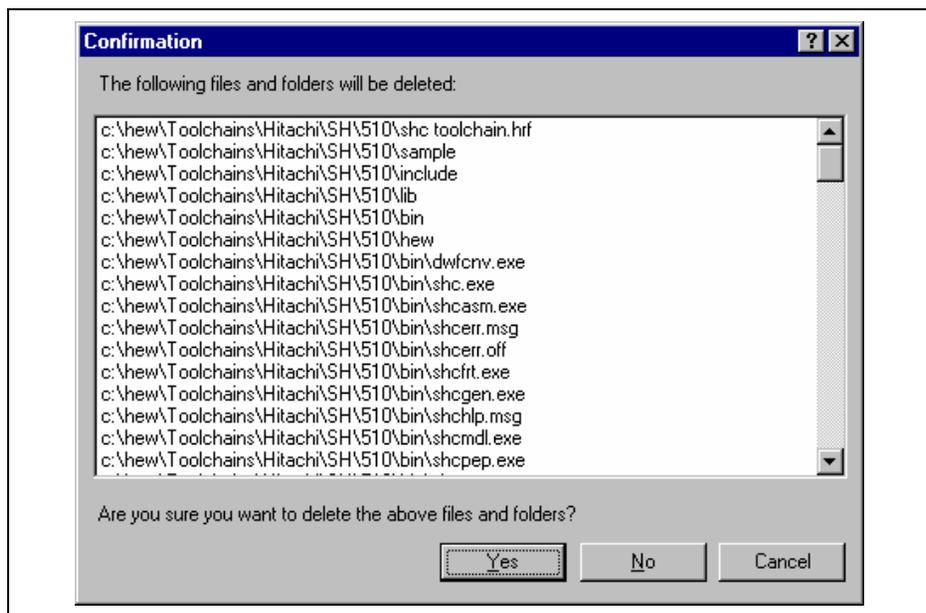
3. Enter the directory in which you would like to search into the top field or browse to it graphically by clicking the “Browse...” button.
4. Check the “Include subfolders” check box if you would like to search the directory specified and all directories below it.
5. Click the “Start” button to begin the search. During the search, the “Start” button will change to a “Stop” button. Click the “Stop” button to halt the search at any time.
6. The results of the search are shown in the “Located Tools which can be uninstalled” list. Select a component and click “Uninstall” to uninstall a component.
7. Click “Exit” to exit the dialog.

A component may only be uninstalled if it is not currently registered with the HEW. If you attempt to uninstall a tool, which is registered, then the dialog shown in figure 5.9 will be displayed. In such a case, you must return to the “Tools Administration” dialog via [**Tools->Administration...**], unregister the tool and then invoke the tool uninstaller again.



Figure 5.9: Unable to Uninstall Tool

If a tool is not registered with the HEW then the dialog shown in figure 5.10 will be displayed when the “Unregister” button is clicked. This confirmation dialog displays all of the files and folders that will be deleted. If you are certain that these files and folders can be deleted then click the “Yes” button. To abort the uninstall click the “No” or “Cancel” buttons.

**Figure 5.10: Confirmation Dialog**

5.7 Technical Support Issues

The “Tools Administration” dialog is also capable of displaying information regarding “hidden” system components. These are part of the HEW itself that cannot be unregistered/registered manually. If you check the “Show all components” check box on the tools administration dialog, extra component folders are displayed (see figure 5.11).

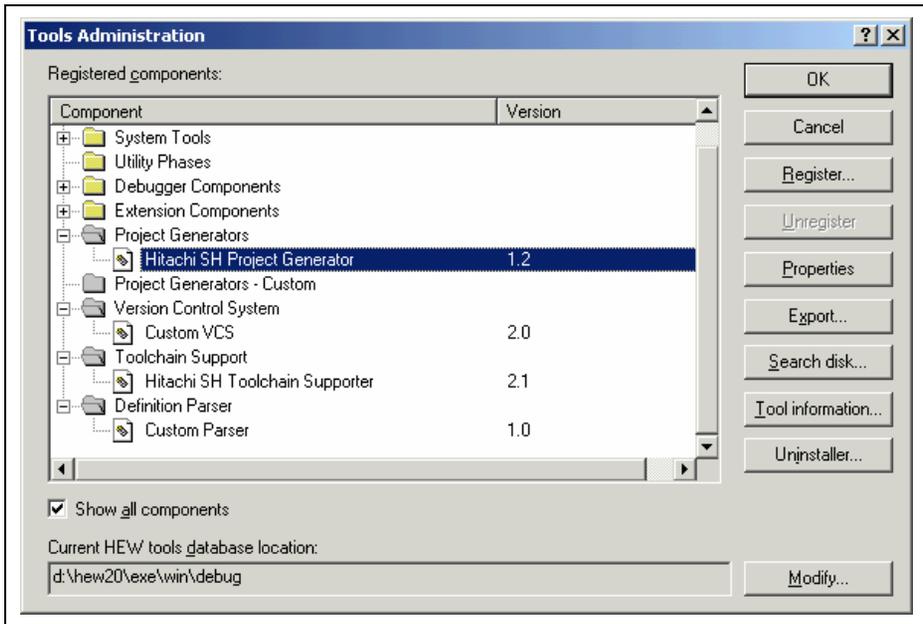


Figure 5.11: All Components Shown

When seeking technical support, you may be asked to give details about some or all of these components. To do so, open the respective folder, select a component and click the “Properties” button. The properties dialog that will be invoked behaves in the same way as discussed previously in this chapter, with the exception that there is no “Environment” tab.

The HEW also has a feature, which outputs tool information regarding the registered components to a file. This allows you to retrieve information on the entire HEW system. This information can then be sent to your technical support contact if you are experiencing problems with the HEW.

☞ To output tool information:

1. Click the [**Tools->Administration**] menu item.
2. Click the “Tool information...” button. A standard windows file save dialog is displayed.
3. Choose the file location and click OK.
4. A file is created in the chosen location with the current registered tool setup of the HEW 2.

5.8 Custom Project Types

The **[Project->Create Project Type...]** menu item in HEW allows you to create a template for your project. This menu item takes the settings of the current project and then creates a project type for you. The user can specify the name of the new type and style of the project generation wizard. Once created these project types appear in the “Tools Administration” dialog and are initially hidden in the system components part of the tools administration tree. To export one of the custom project generators select the “Export” button on the “Tools Administration” dialog. The execution environments of the custom project generators are packaged on the execution file that can be installed. When this file is executed on the target user’s machine, the custom project generator is installed.

6. Customizing the Environment

6.1 Customizing the Toolbar

The High-performance Embedded Workshop provides 2 standard toolbars as detailed in chapter 1, “Overview”. In addition to these, you may also construct your own toolbars via the “Customize” dialog (figure 6.1).

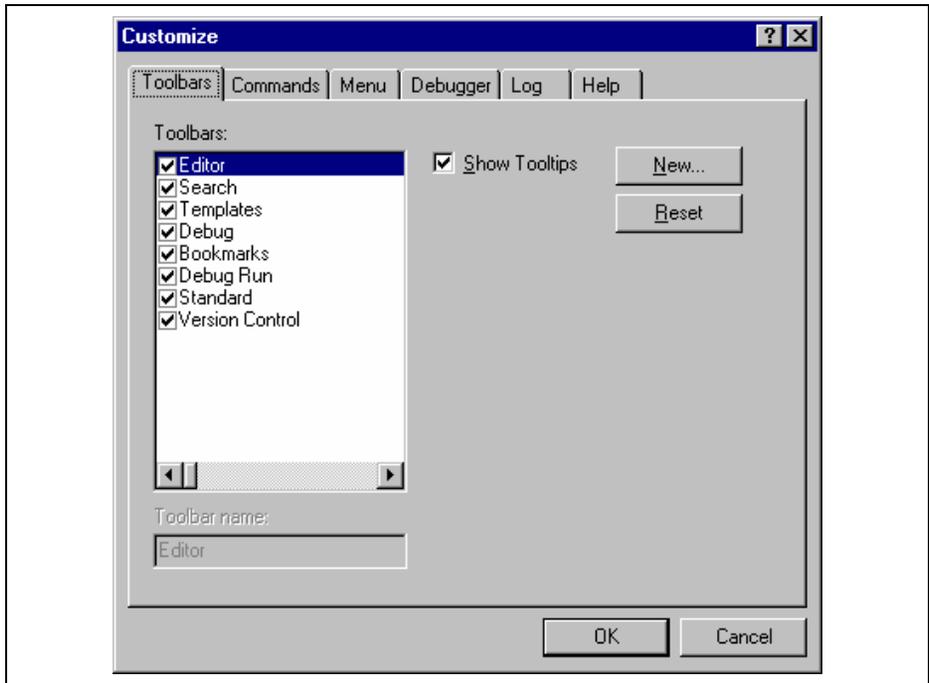


Figure 6.1: Customize Dialog Toolbars Tab

➡ To create a new toolbar:

1. Select [**T**ools->**C**ustomize...]. The dialog shown in figure 6.1 will be displayed.
2. Click the “New...” button. The dialog shown in figure 6.2 will be displayed.
3. Enter the name of the new toolbar into the “Toolbar name” field.
4. Click “OK” to create the new toolbar.

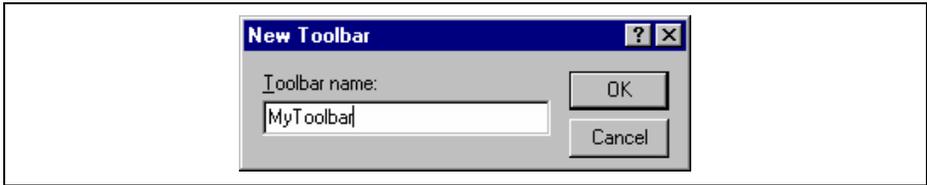


Figure 6.2: New Toolbar Dialog

When a new toolbar is created it will appear undocked (i.e. “floating”) and empty.

➡ To add buttons to a toolbar:

1. Select [Tools->Customize...]. The dialog shown in figure 6.1 will be displayed. Select the “Commands” tab (see figure 6.3).
2. Browse the available buttons by selecting the button categories from the “Categories” list. Select a button from the “Buttons” area to display information on its operation.
3. Click and drag a button from the dialog onto the toolbar.

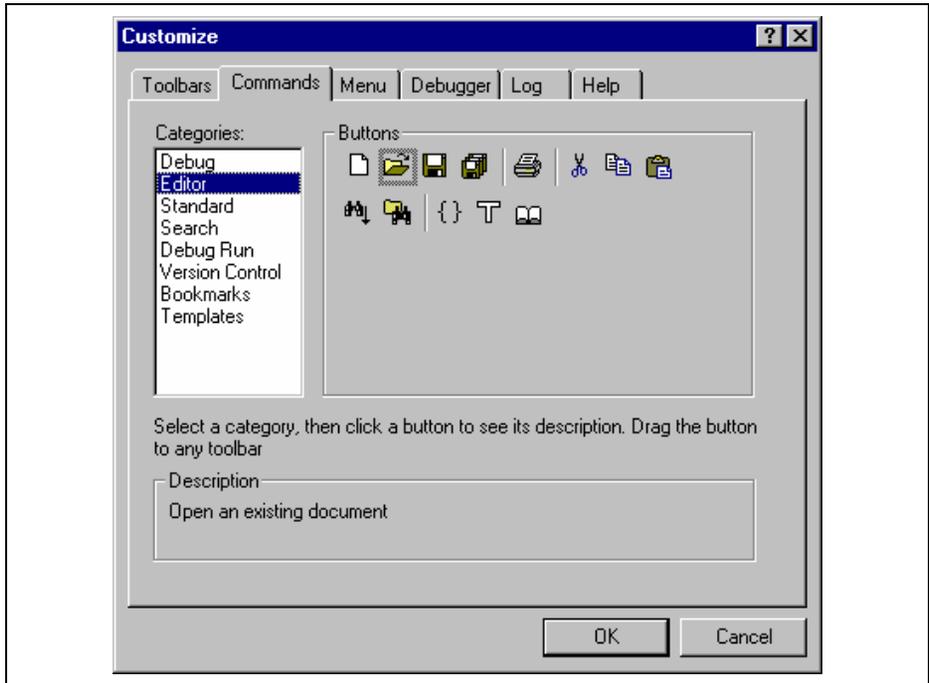


Figure 6.3: Customize Dialog Commands Tab

- To remove buttons from a toolbar:
 1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed. Select the “Commands” tab (see figure 6.3).
 2. Click and drag a button from the toolbar onto the “Buttons” area.
- To remove a user defined toolbar:
 1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed.
 2. Select the user-defined toolbar from the “Toolbars” list, and the “Reset” button in figure 6.1 changes to the “Delete” button. Then click the “Delete” button.
- To reset a standard toolbar back to its original state:
 1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed.
 2. Select the standard toolbar from the “Toolbars” list and then click the “Reset” button.
- To show or hide toolbar tooltips:
 1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed.
 2. Set the “Show Tooltips” check box as desired.
- To modify the toolbar name of a toolbar created by a user:
 1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed.
 2. In the “Toolbars” list, select a toolbar, which has been created by a user and whose name you, want to modify.
 3. Modify the name of the toolbar in the “Toolbar name” field.

6.2 Customizing the Tools Menu

The “Tools” menu can be customized to include your own menu options.

➤ To add a new menu option:

1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed. Select the “Menu” tab (see figure 6.4). The first thing for you to decide is whether you are adding a global application wide tool (“Application wide tools:”), which will be available to all of your workspaces. Or whether you wish to add a workspace wide tool (“Workspace wide tool:”), which is only valid for the current workspace. Once you have made the choice choose the relevant section of the dialog.

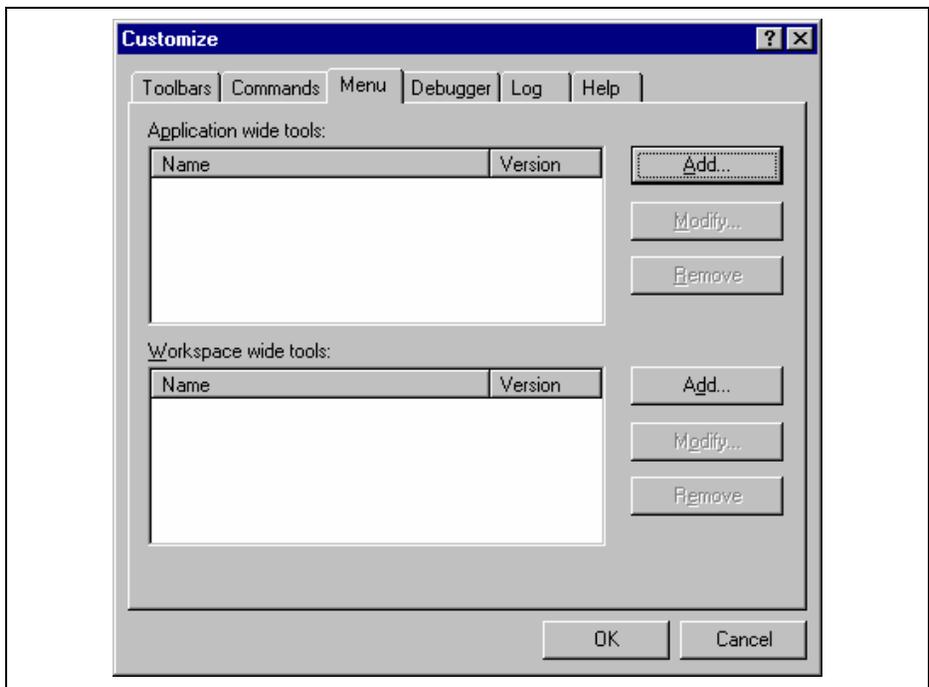


Figure 6.4: Customize Dialog Menu Tab

2. Click the “Add...” button (the dialog shown in figure 6.5 will be invoked). If you would like to add an existing system tool to the menu then select the “Select from existing system tools” radio button, choose the tool from the drop-down list and then click “OK”. Alternatively, if you would like to add a tool of your own then follow the remaining steps.

3. Enter the name of the tool into the “Name” field.
4. Enter the command, excluding arguments, into the “Command” field.
5. Enter any arguments that you would like to pass to the command into the “Arguments” field.
6. Enter an initial directory in which you would like the tool to run, into the “Initial directory” field.
7. Click “OK” to add the menu option to the “Tools” menu.

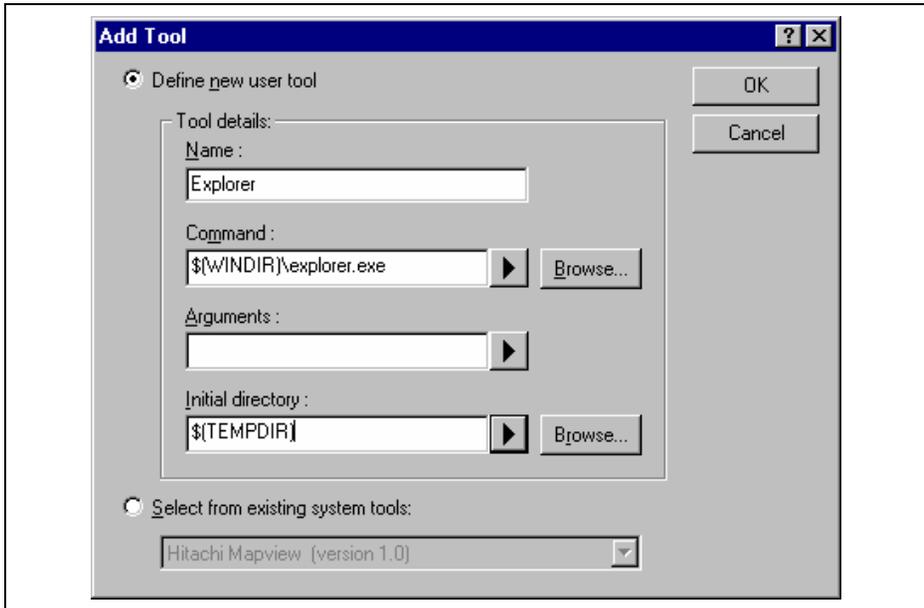


Figure 6.5: Add Tool Dialog

New menu options are added to the bottom of the list (i.e. bottom of the tools menu) by default. The order of menu options in the “Tools” menu can also be modified.

☞ To modify a menu option:

1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed. Select the “Menu” tab (see figure 6.4).
2. Select the menu option that you would like to modify and then click the “Modify...” button.
3. Make the desired changes on the “Modify Tool” dialog (figure 6.6) and then click “OK”.

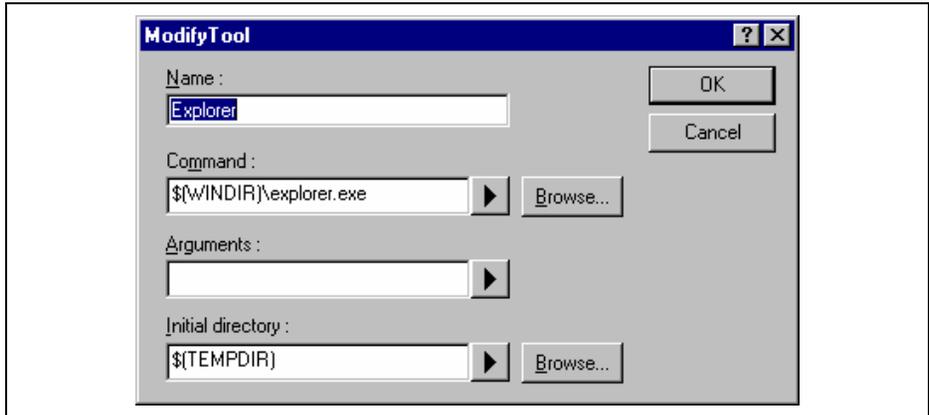


Figure 6.6: Modify Tool Dialog

- To remove a menu option:
 1. Select [**T**ools->**C**ustomize...]. The dialog shown in figure 6.1 will be displayed. Select the “Menu” tab (see figure 6.4).
 2. Select the menu option that you would like to remove and then click the “Remove” button.

6.3 Configuring the Help System

The High-performance Embedded Workshop provides context sensitive help within the editor window. In other words, if you select some text in the editor window and then press **F1**, the High-performance Embedded Workshop will attempt to locate help on that selected item. The help files, which will be searched, are listed in the “Help” tab of the “Customize” dialog.

- To add a new help file:
 1. Select [**T**ools->**C**ustomize...]. The dialog shown in figure 6.1 will be displayed. Select the “Help” tab (see figure 6.7).

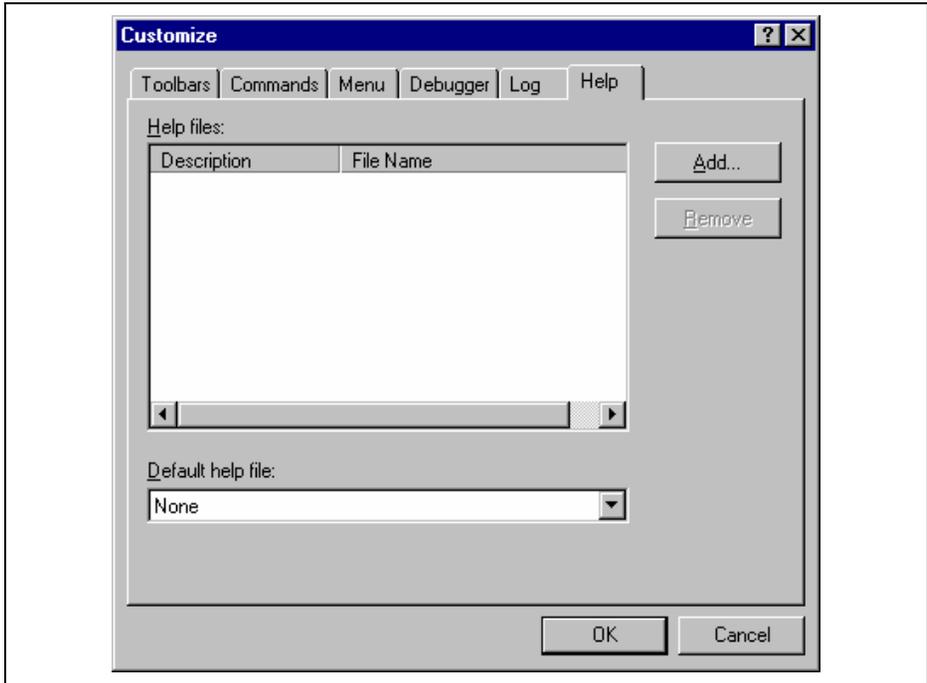


Figure 6.7: Customize Dialog Help Tab

2. Click the “Add...” button. The “Add Help File” dialog will be displayed (figure 6.8).
3. Enter a description of the help file into the “Title” field.
4. Enter the full path of the help file into the “Path” field (or browse to it graphically by clicking on the “Browse...” button).
5. Click “OK” to define the new help file.

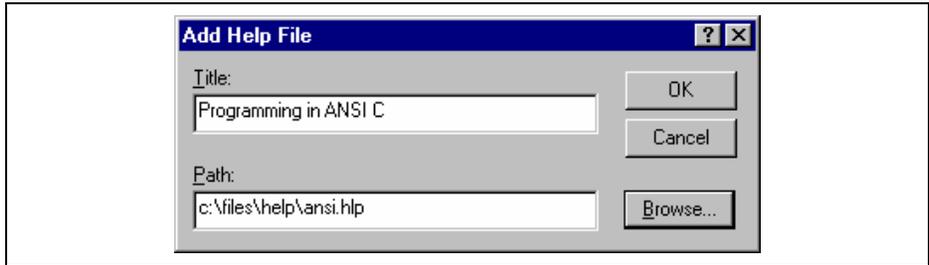


Figure 6.8: Add Help File Dialog

To make a help file the default choice, select it from the “Default help file” drop-down list or set it to “(None)” if you would like to be prompted for a help file when **F1** is pressed.

6.4 Specifying Workspace Options

The High-performance Embedded Workshop allows you to control several aspects of a workspace via the “Options” dialog (figure 6.9). To invoke it select [**Tools->Options...**], and select the “Workspace” tab.

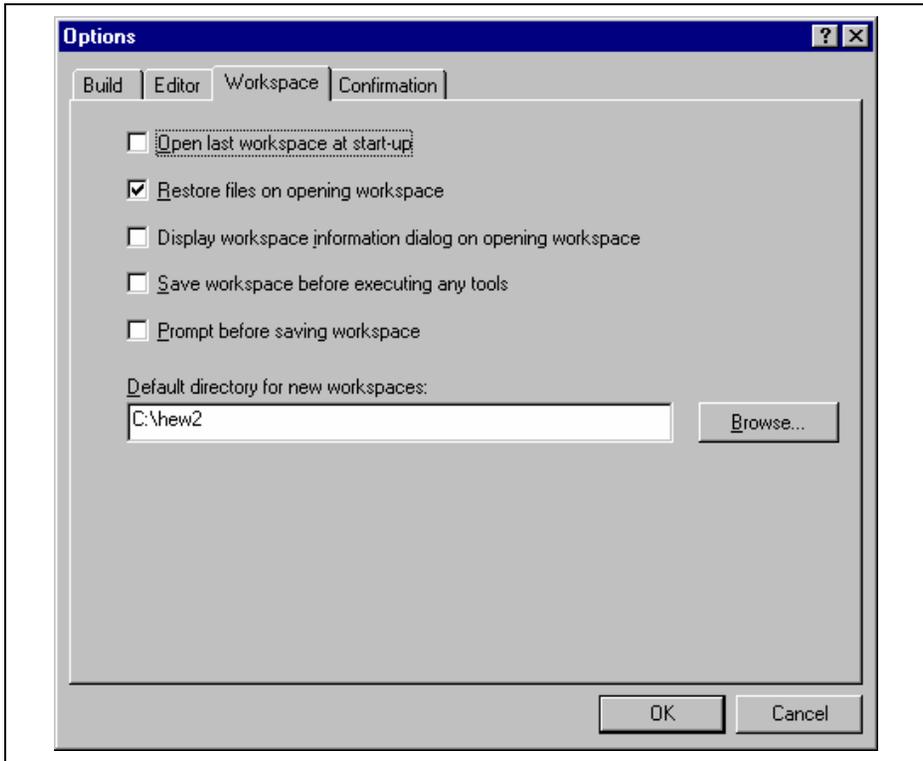


Figure 6.9: Options Dialog Workspace Tab

The following sections explain the options available on this tab.

6.4.1 Open last workspace at start-up

Set this check box if you would like the High-performance Embedded Workshop to automatically open the last workspace you opened when it is launched.

6.4.2 Restore the files on opening workspace

When you close a workspace, the HEW stores, which files were open. When you open a workspace, the HEW can restore (i.e. open) the same files so that you can continue your session in exactly the same state as when you left it. If you would like the files associated with a workspace to be opened when you open a workspace then set this check box.

6.4.3 Display workspace information dialog on opening workspace

When many workspaces are being used, it is sometimes difficult to remember exactly what was contained within each workspace. To help resolve this, the High-performance Embedded Workshop allows you to enter a textual description of each workspace.

➡ To enter a workspace description:

1. Select the workspace icon from the “Projects” tab of the “Workspace” window.
2. Click the right mouse button to invoke the pop-up menu and then select the “Properties” option. The dialog shown in figure 6.10 will be displayed.
3. Enter the description into the “Information” field.
4. Check the “Show workspace information on workspace open” check box if you want a workspace properties dialog to be launched on opening a workspace. This check box has the same role as the “Display workspace information dialog on opening workspace” on the “Workspace” tab of the “Options” dialog.
5. Click “OK” to save the description on the “Information” dialog. Click the “Cancel” button not to save the description.

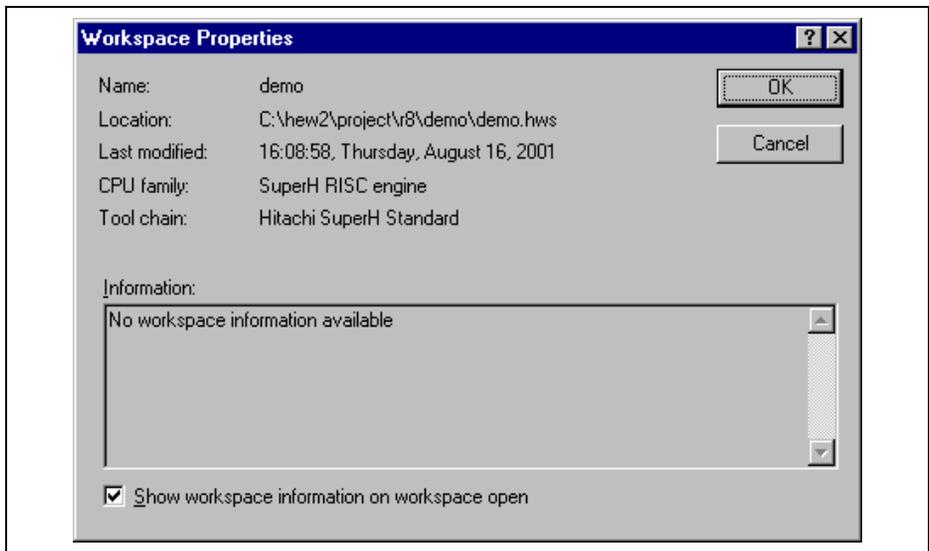


Figure 6.10: Workspace Properties Dialog

When a workspace is opened, the High-performance Embedded Workshop can display this information so that it is possible to determine whether the workspace is the desired workspace. To

display this information on opening a workspace, set the “Display workspace information dialog on opening workspace” check box.

6.4.4 Save workspace before executing any tools

To force the High-performance Embedded Workshop into saving the current workspace before executing any build phases (i.e. build, build all or build file operations) or version control commands set the “Save workspace before executing any phases” check box.

6.4.5 Prompt before saving workspace

In addition to the above check box, set this to prompt before saving.

6.4.6 Default directory for new workspaces

When a new workspace is created the High-performance Embedded Workshop invokes the “New Workspace” dialog. One of the fields on this dialog is the directory in which the new workspace will be created. By default, this is the root directory. However, if you would like to set this default directory to another location (e.g. “C:\Workspaces”) then enter the desired directory into the field or browse to it graphically via the “Browse...” button.

6.4.7 Prompt before saving session

Checking this option will force the High-performance Embedded Workshop into displaying a prompt before the session is saved to disk.

6.5 Using an External Editor

The High-performance Embedded Workshop allows you to use an external editor. Once an external editor has been specified, it will be launched when the following actions are performed:

- Double clicking on a file in the “Projects” tab of the “Workspace” window.
 - Double clicking on an entry in the “Navigation” tab of the “Workspace” window.
 - Double clicking on an error/warning in the “Build” tab of the “Output” window.
 - Double clicking on an entry in the “Find in Files” tab of the “Output” window.
 - Selecting the [**O**pen <file>] option from the “Workspace” windows pop-up menu.
 - Clicking the “Launch Editor” toolbar button.
- ⇒ To specify an external editor:
1. Select [**T**ools->**O**ptions...]. The “Options” dialog will be displayed. Select the “Editor” tab (figure 6.11).

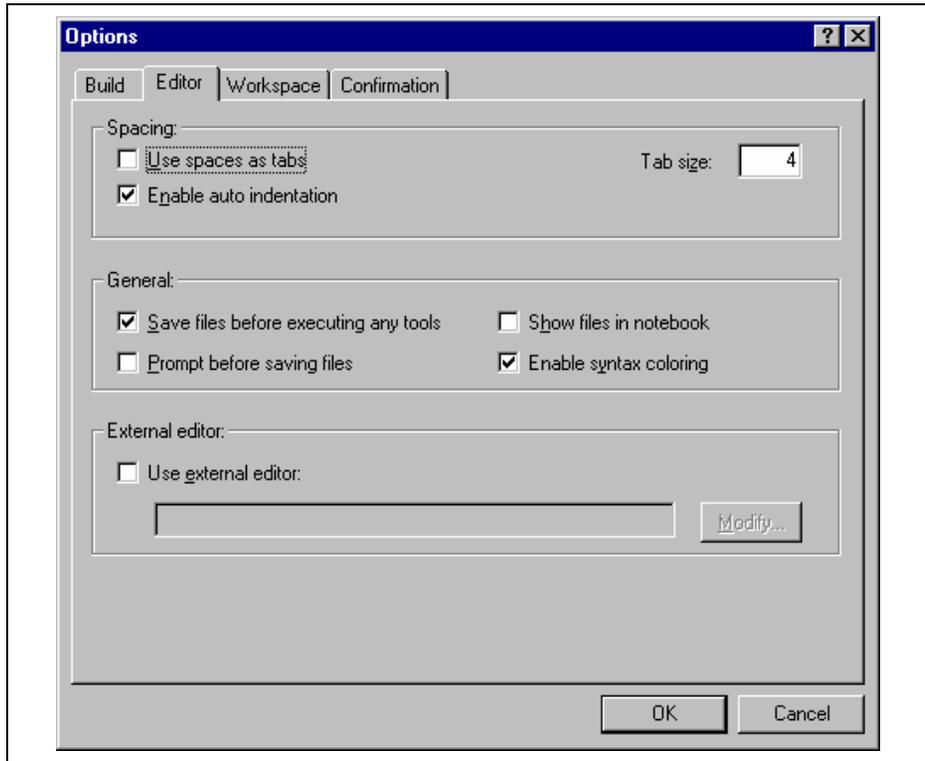


Figure 6.11: Options Dialog Editor Tab

2. Check the “Use external editor” check box.

The “External Editor” dialog will be displayed (figure 6.12).

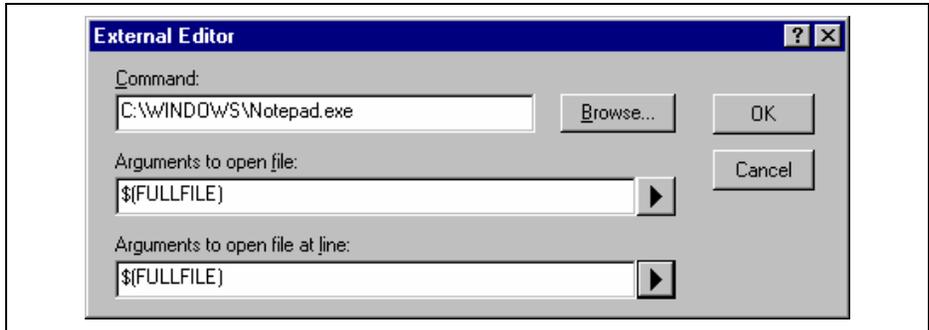


Figure 6.12: External Editor Dialog

3. Enter the path of the executable (without any arguments) into the “Command” field.
4. Enter the arguments required to open a file into the “Arguments to open file” field. Use the \$(FULLFILE) placeholder to represent the path of the file to be opened.
5. Enter the arguments required to open a file at a specific line into the “Arguments to open file at line” field. Use the \$(FULLFILE) placeholder to represent the path of the file to be opened and the \$(LINE) placeholder to represent the line number at which the cursor should be initially positioned.
6. Click “OK” to define the editor.

Note: When using an external editor be aware of the following issues:

- Each time you invoke the external editor, in whichever way, a separate instance of the editor will be launched.
- You must save your own files before you perform a build file, build or build all operation.

6.6 Customizing File Save

The High-performance Embedded Workshop allows you to customize file save on the “Editor” tab of the “Options” dialog (figure 6.11). To open the tab, select [**Tools->Options...**] and click the “Editor” tab.

The following sections explain the options related to file save.

6.6.1 Save files before executing any tools

To force the High-performance Embedded Workshop into saving edited files before executing any build phases (i.e. build, build all or build file operations) or version control commands, set the “Save files before executing any tools” check box.

6.6.2 Prompt before saving files

In addition to the above check box, set this to prompt before saving.

6.7 Using an External Debugger

The High-performance Embedded Workshop can launch an external debugger tool. If you want to use another debugger then you must add it to the “Tools” menu.

The “Debugger” tab of the “Customize” dialog (figure 6.13) is where the HDI-related information is configured. You may wish to use an older version of the debugger if certain targets are not currently supported in the new environment. Invoke it by selecting [**Tools->Customize...**] and then selecting the “Debugger” tab.

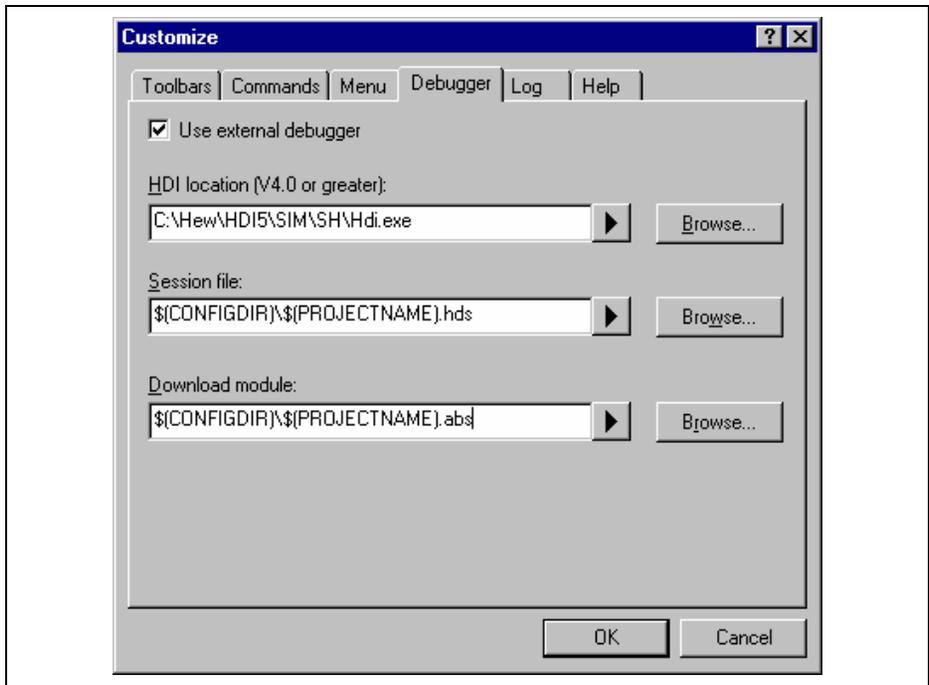


Figure 6.13: Customize Dialog Debugger Tab

To use an external debugger, check the “Use external debugger” checkbox and specify the items described below. There are three items of information, which need to be specified. Firstly, the location of the HDI executable must be specified. This must be version 4.0 or greater otherwise the behavior is not guaranteed. The second item of data is the session file. This tells HDI which session to load when it is launched. Finally, the location of the download module is required. This

allows the HEW to automatically switch to HDI when the download module changes after a build. Click the “Launch External Debugger” toolbar button to invoke HDI with the specified session file.

After a build, if the download module has been updated, the HEW will switch back to HDI to enable immediate debugging. Whilst using HDI, double clicking in any source window will switch back to the HEW with the source file open at the line which was double clicked.

6.8 Using Custom Placeholders

Throughout the High-performance Embedded Workshop the user can use a number of pre-defined placeholders for directory definitions. For example the user can use the “\$(PROJDIR)” variable to signify the current HEW project directory. This makes it much easier to relocate projects and keep all of the paths correct.

The High-performance Embedded Workshop also has the ability to define custom placeholders. This means you can enter your own custom placeholder definition and decide upon its directory value. Once defined this placeholder becomes available throughout the rest of the HEW system.

The placeholders can be defined on an application wide level so the placeholders are available to all workspaces and projects that use the HEW. The other method of defining the placeholders is using the workspace wide custom placeholders this means the placeholders can only be used in the current workspace. This list is only available when you have a workspace open.

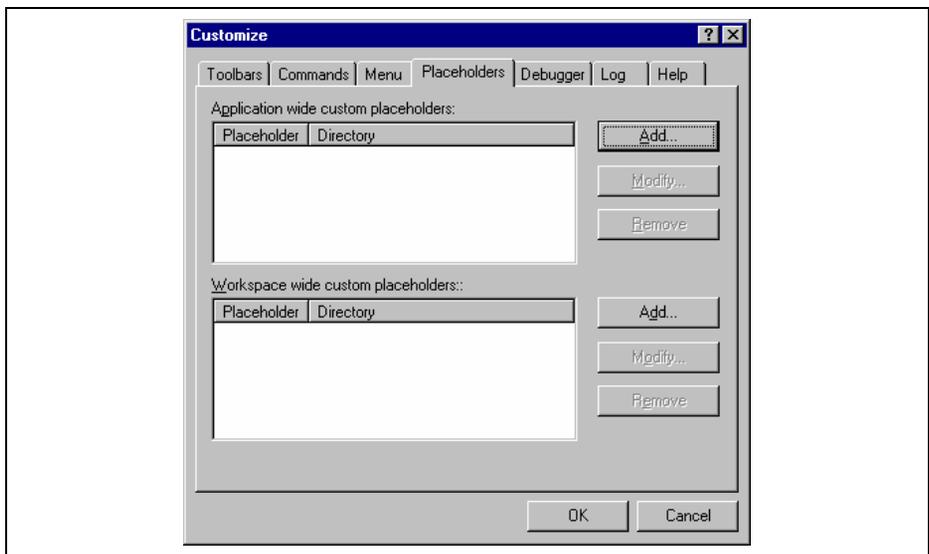


Figure 6.14: Customize Dialog Placeholder Tab

➡ To add a custom placeholder:

1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed. Select the “Placeholders” tab (figure 6.14).
2. Choose whether you need to use an “Application wide custom placeholder” or “Workspace wide custom placeholder”. Click “Add” on the adjacent button to the list you require.
3. The dialog, add “New Custom Placeholder” dialog is displayed. (figure 6.15)
4. In the fields provided choose a suitable name for the placeholder and a description of what the placeholder means.
5. Then choose a directory, which relates to this placeholder. It is possible to use placeholders that are already defined in this field such as \$(PROJDIR).

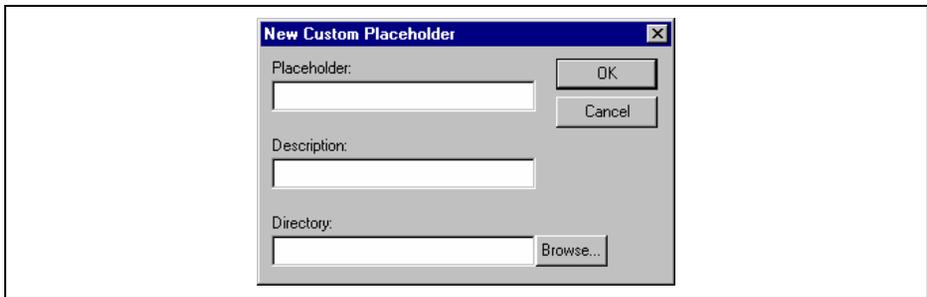


Figure 6.15: New Custom Placeholder Dialog

6.9 Using Confirmation Dialog Boxes

The HEW allows users to specify many items for confirmation even in a single operation. These items are usually set by default. However, it may be annoying to handle the confirmation dialog boxes if you take the same operation several times. Use the [Confirmation] page of the [Options] dialog box to control those confirmation dialog boxes.

- ☞ To view the content of [Display confirmation dialogs for]:
 1. Select [**Tools->Options...**]. Then select the “Confirmation” tab (figure 6.16).
 2. Items to be confirmed are listed in the dialog box.
 3. To toggle a checkmark for an item, select the checkbox on the left of the item’s name.

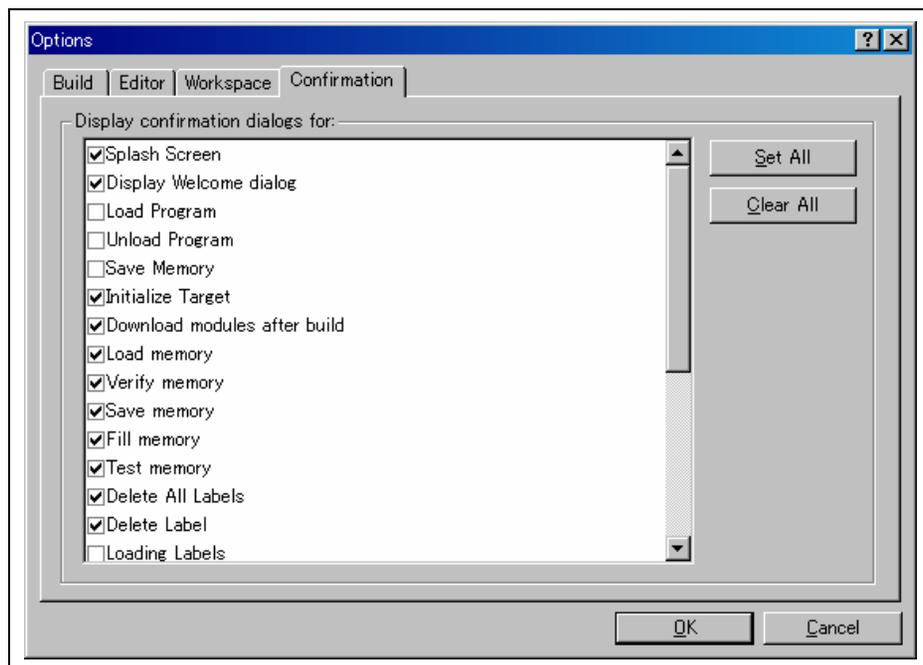


Figure 6.16: Confirmation Page for Optional Dialog Boxes

7. Version Control

The High-performance Embedded Workshop provides facilities for connecting to a version control tool. Some of the reasons why version control tools are used with a project are:

- To maintain the integrity of a project.
- To store each stage of a project.
- To enable different users to co-develop a project by controlling revisions to its source files.

Figure 7.1 illustrates a typical project where a version control system is in use. This shows three users who all use the same-shared network drive to exchange source code. The version control system provides access and updates to the source files.

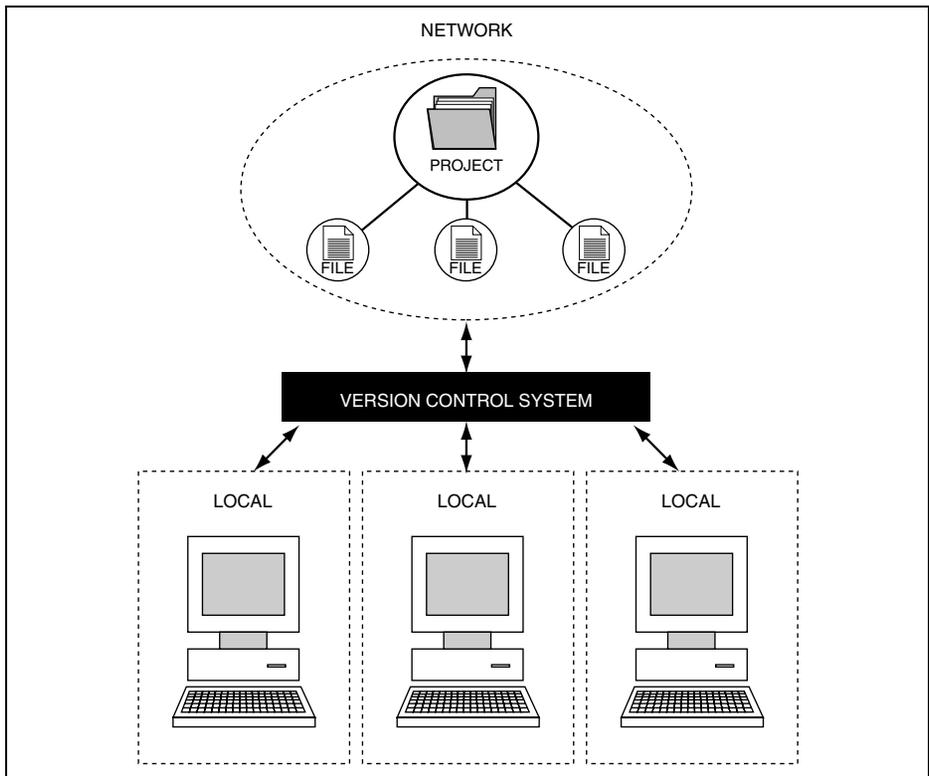


Figure 7.1: Version Control

7.1 Selecting a Version Control System

Initially, the version control sub-menu will appear as shown in figure 7.2. At this time only the [Version Control -> Select...] option is available because a version control system is not yet active for the current workspace.

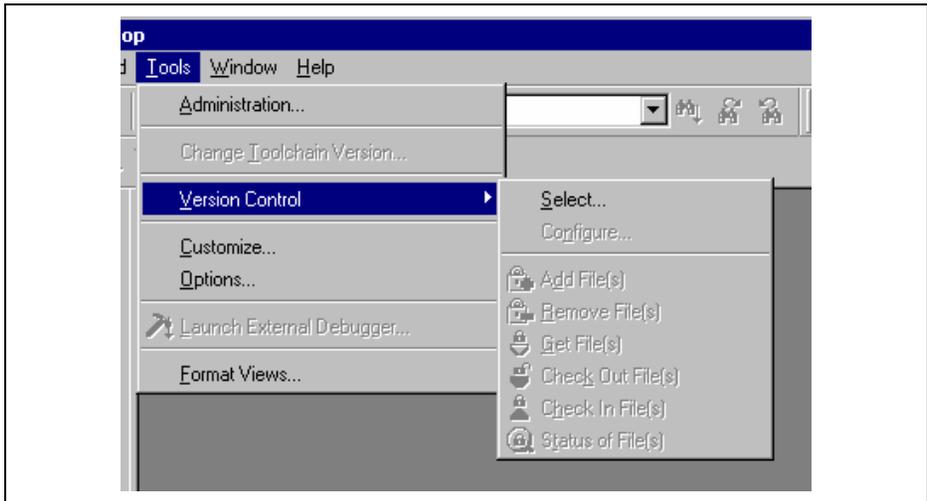


Figure 7.2: Version Control Sub-Menu

- ☞ To select a version control system:
1. Select [Version Control->Select...]. The dialog shown in figure 7.3 will be displayed. This dialog lists all of the supported version control systems.
 2. Select the desired version control system from the “Version control systems” list and click the “Select” button. The “Current version control system” is changed to reflect the new selection.
 3. Click the “OK” button to confirm the selection.

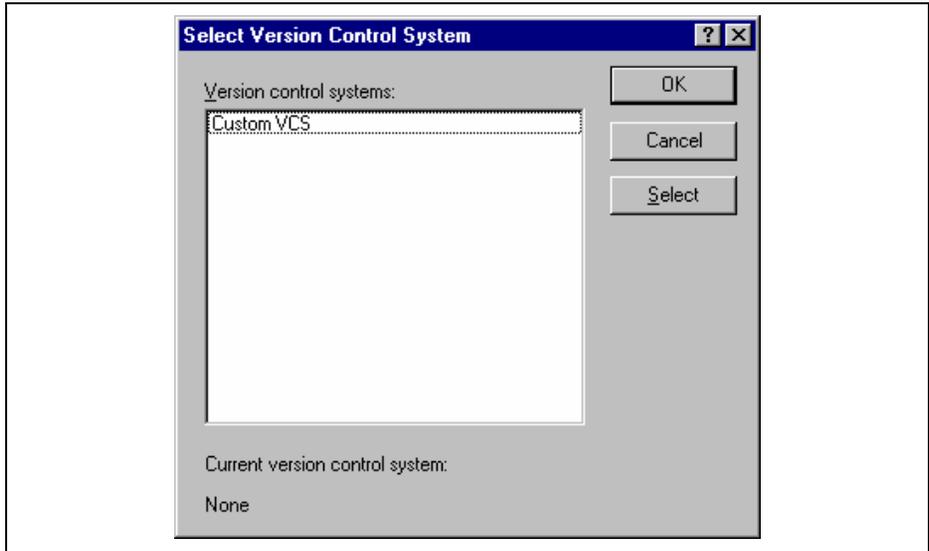


Figure 7.3: Select Version Control System Dialog

Note: Only those version control systems which have been installed with the HEW will appear in the “Select Version Control System” dialog (figure 7.3).

Once a version control tool is selected you will notice that the [**Version Control->Configure...**] option has now become available.

The next chapter discusses the usage of the custom version control system.

8. Using the Custom Version Control System

The custom version control system is a configurable addition to the High-performance Embedded Workshop, which allows you to connect to a version control system already installed on your machine. To clarify further, the High-performance Embedded Workshop does not provide a version control tool itself, only a means by which you can integrate the version control system, which you use into your workspaces and projects.

8.1 Defining Version Control Menu Options

The custom version control system allows you to invoke a version control command either by selecting an option from the **[Tools->Version Control]** sub-menu or by clicking a version control toolbar button. When either of these actions are performed, the associated commands are executed and the output is displayed in the “Version Control” tab of the “Output” window.

➡ To execute a version control menu option or toolbar button:

1. Select whichever items you would like to apply the version control command to from the “Workspace” window. This may include a workspace, project(s), folder(s) and file(s).
When the command is selected, all of the files will be extracted from the selected items and passed, in turn, to the version control command. For example, if you select the workspace icon then all of the files in all of the projects will be passed, in turn, to the version control command. This will include any system files. For example if you select the project item then
2. Select the required menu option from the **[Tools->Version Control]** sub-menu or click the desired version control toolbar button.

The custom version control support allows you the most flexibility in specifying how a version control system is to be used. To configure it, select **[Version Control->Configure...]**. The “Version Control Setup” dialog will be displayed (figure 8.1).

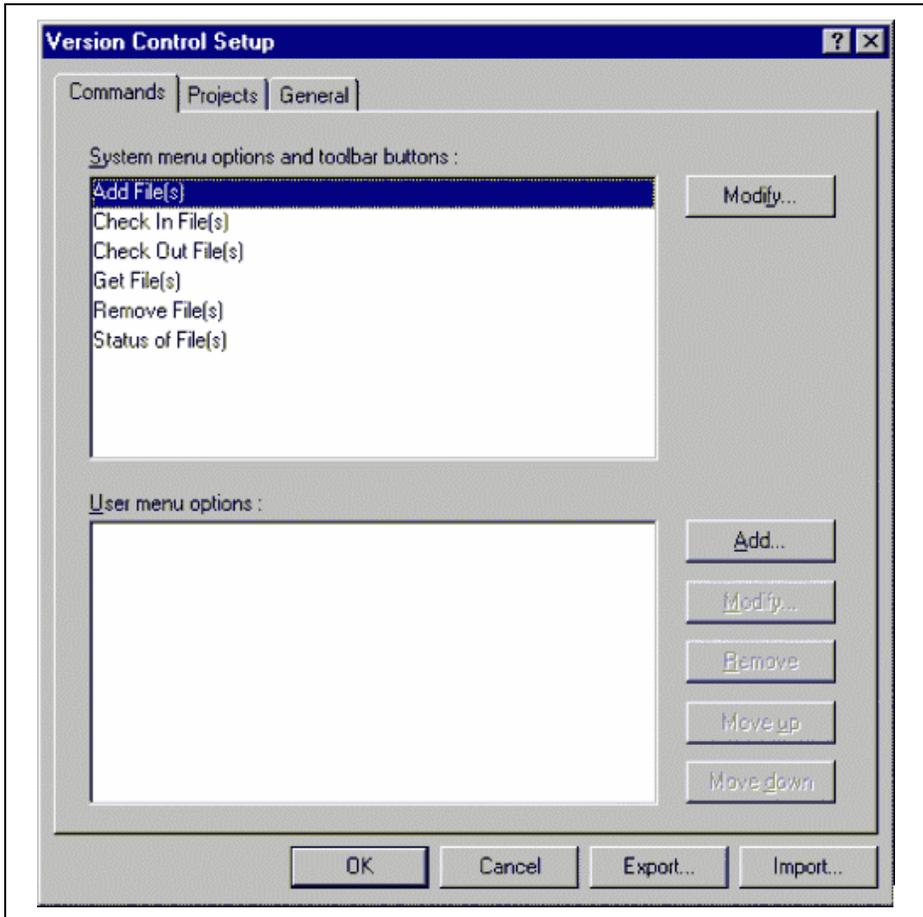


Figure 8.1: Version Control Setup Dialog Commands Tab

The “Commands” tab contains two lists of menu options. The first list, “System menu options and toolbar buttons”, represents those menu options which always appear on the version control sub-menu. These menu options also have an associated toolbar button on the version control toolbar. The second list, “User menu options”, represents those additional user defined options which are added to the bottom of the version control sub-menu. Figure 8.2 shows the structure of the version control sub-menu.

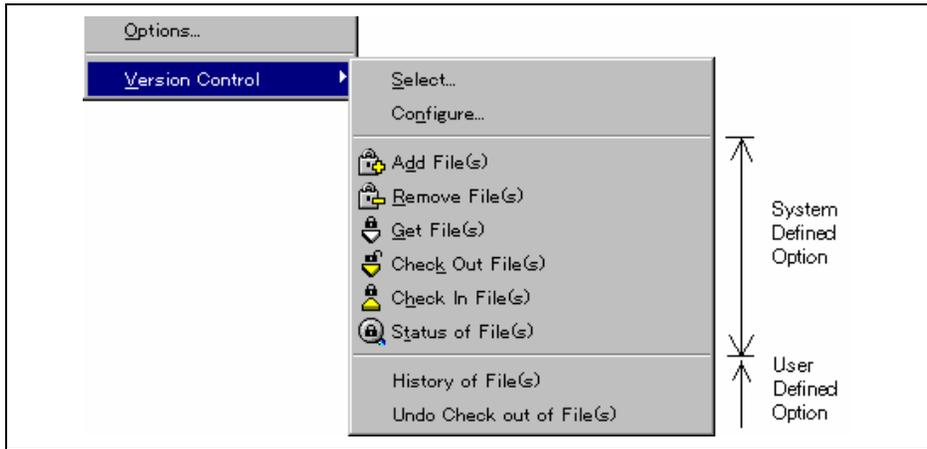


Figure 8.2: Version Control Sub-Menu

8.1.1 System menu options and toolbar buttons

In order to invoke commands from the toolbar or the system defined options of the **[Tools->Version Control]** sub-menu, you must first define the associated commands that should be executed when they are activated. The names of the options and their intended action are listed in table 8.1.

Table 8.1: System Menu Option

Option	Description
Add File(s)	Add selected file(s) to version control system.
Remove File(s)	Remove selected file(s) from version control system.
Get File(s)	Get a read only local copy of the selected file(s) from version control system.
Check In File(s)	Put back, i.e. update, the selected file(s) in version control system with the local copy.
Check Out File(s)	Get a writable local copy of the selected file(s) from version control system.
Status of File(s)	View the status of the selected file(s).

☞ To modify a system menu / toolbar option:

1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed.
2. Select the option to be modified from the “System menu options and toolbar buttons” list and then click the “Modify...” button. The dialog shown in figure 8.3 will be displayed. This figure shows a dialog when “Add File(s)” has been selected for example.
3. Commands are added via the “Add...” button. See the section, “*Defining Version Control Commands*”, later in this chapter for further information.
4. Close the “Define Command for “<command>”” dialog by clicking “OK”.
5. Close the “Version Control Setup” dialog by clicking “OK”.

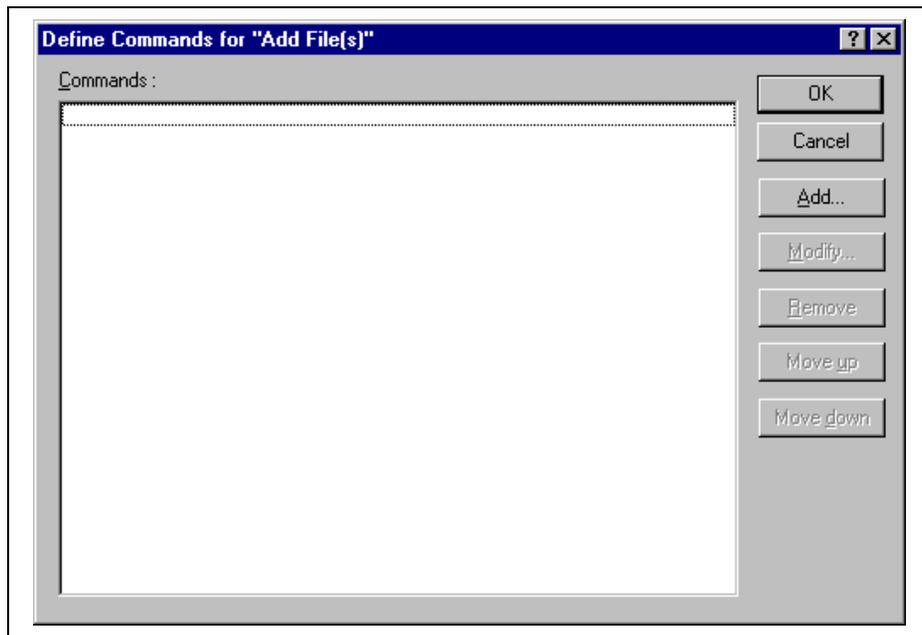


Figure 8.3: Modify System Menu Option (Example)

8.1.2 User menu options

You can create as many user defined menu options as you like, name them how you want and define their order in the menu. User defined menu options do not appear on the version control toolbar.

- ☞ To create a new version control menu option:
 1. Select [**V**ersion **C**ontrol->**C**onfigure...]. The dialog shown in figure 8.1 will be displayed.
 2. Click the “Add..” button. The dialog shown in figure 8.4 will be displayed.
 3. Enter the name of the menu option into the “Option” field.
 4. Commands are added to the menu option via the “Add...” button. See the section, “*Defining Version Control Commands*”, later in this chapter for further information.
 5. Close the “Add Menu Option” dialog by clicking “OK”.
 6. Close the “Version Control Setup” dialog by clicking “OK”.

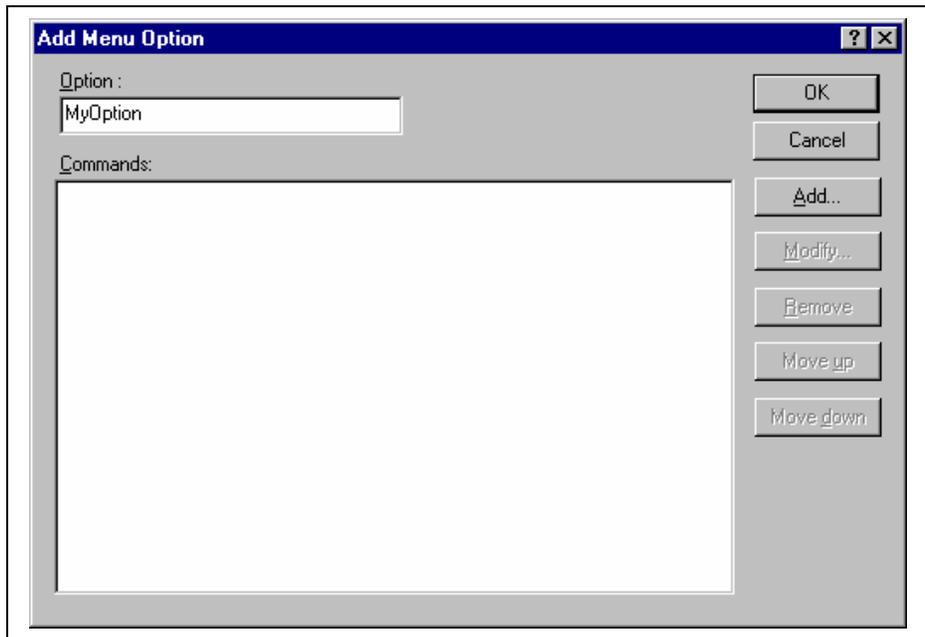


Figure 8.4: Add Menu Option Dialog

- To remove an existing version control menu option:
 1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed.
 2. Select the menu option to be removed from the “User menu options” list and then click the “Remove” button.
 3. Close the “Version Control Setup” dialog by clicking “OK”.
- To modify an existing version control menu option:
 1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed.
 2. Select the menu option to be modified from the “User menu options” list and then click the “Modify...” button beside the list. The dialog shown in figure 8.4 will be displayed. (The title of the dialog is “Modify Menu Option”.)
 3. Modify the commands as necessary and then click “OK”.
 4. Close the “Version Control Setup” dialog by clicking “OK”.
- To change the ordering of version control menu options:
 1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed.
 2. Select the menu option to be moved and then click the “Move up” and “Move down” buttons as necessary.
 3. Close the “Version Control Setup” dialog by clicking “OK”.

8.2 Defining Version Control Commands

Commands are defined when the “Add...” or “Modify...” buttons are clicked on the dialogs shown in figure 8.3 and figure 7.4. In either case, the dialog shown in figure 8.5 is invoked.

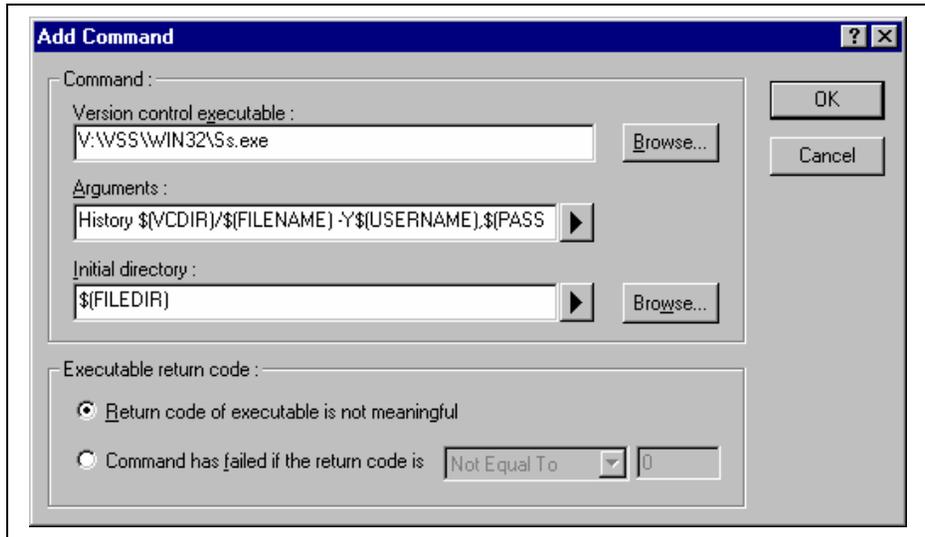


Figure 8.5: Add/Modify Command Dialog

- To define a command:
 1. Enter the full path of the command into the “Version control executable” field or browse to it graphically by clicking the “Browse...” button.
 2. Enter the arguments for the command into the “Arguments” field.
 3. Enter the initial directory in which you would like to run the executable from into the “Initial directory” field or browse to it graphically by clicking the “Browse...” button. In most cases this should be set to the “\$(FILEDIR)” placeholder, i.e. execute the command from the same directory as the file.
 4. Set the “Executable return code” options as described in the following section.
 5. Click “OK” to define the new command.

8.2.1 Executable return code

If the return code of the command(s) can be used to indicate a failure then you should select the “Command has failed if the return code is” option and set the two fields to the right as required.

If the “Command has failed if the return code is” option is selected then the HEW will check the return code of each command to determine whether a failure occurred. If so, no further commands will be executed and any other processes which would follow the commands (e.g. build) will not be executed.

If the “Return code of tool is not meaningful” option is selected then the HEW will not check the return code of each. Consequently, all commands will execute regardless.

8.3 Specifying Arguments

It is obvious that arguments must be specified correctly, otherwise the version control tool executed will not function as intended. However, it is also important, when using custom version control support, to specify the arguments in a *flexible* way as a single version control command can be applied to more than one file. To facilitate this, the “Arguments” field has a placeholder button (refer to appendix C, “Placeholders”, for an in depth discussion of placeholders) which, when clicked on, invokes a pop-up menu of available placeholders (figure 8.6). An explanation of each placeholder and how their values are derived can be found in table 8.2, Arguments Field Placeholders.

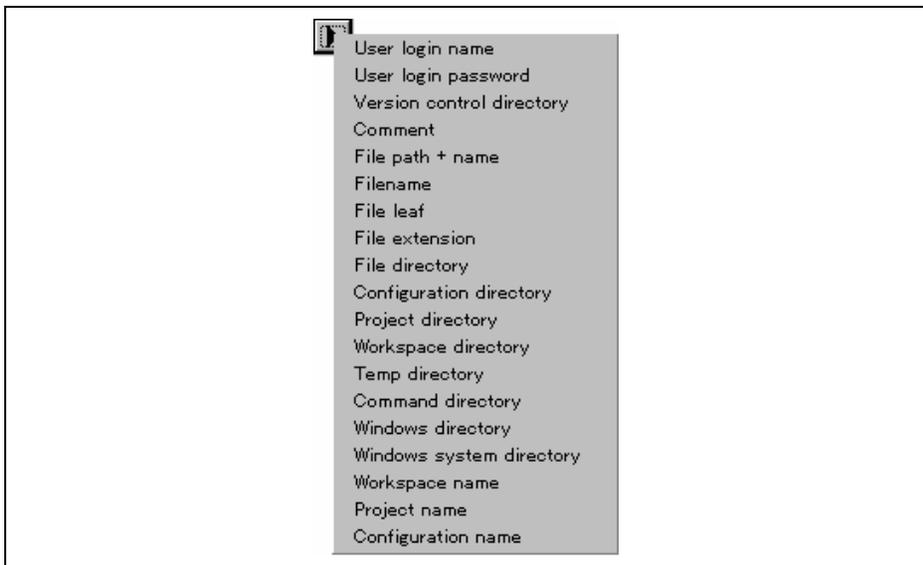


Figure 8.6: Arguments Field Placeholder Pop-up Menu

Table 8.2: Arguments Field Placeholders

Placeholder	Value and How its Determined
User login name	Current user login (“General” tab)
User login password	Current user password (“General” tab)
Version control directory	“Virtual” version control mapping (“Projects” tab)
Comment	Comment specified before command execution
File path + name	Full path and name of file involved in operation
Filename	Filename (including extension) of file involved
File leaf	Filename (excluding extension) of file involved
File extension	Extension of file involved in operation
File directory	Directory of file involved in operation
Configuration directory	Current configuration directory
Project directory	Current project directory
Workspace directory	Current workspace directory
Temp directory	Temporary directory
Command directory	Version control executable directory
Windows directory	Directory where Windows® is installed
Windows system directory	Directory where Windows® system files exist
Workspace name	Current workspace name
Project name	Current project name
Configuration name	Current configuration name

8.3.1 Specifying File Locations

When referring to a file’s location, be sure to use a placeholder, otherwise the command will only relate to a hardwired file. For example, let’s imagine that a version control executable has been selected which uses a –GET command to obtain a read only copy of a file. The “Arguments” field could be specified as:

```
-GET "c:\vc\files\project\main.c"
```

However, when executed, this command can only ever get the file MAIN.C. To resolve this problem, HEW uses a system of placeholders and directory mappings. The latter tell the HEW which “working” directories (i.e. where source files are being worked on) map to which “controlled” directories (i.e. where the source files are stored in the version control system). Mappings between these two directory systems can be specified via the “Projects” tab of the “Version Control Setup” dialog (figure 8.7).

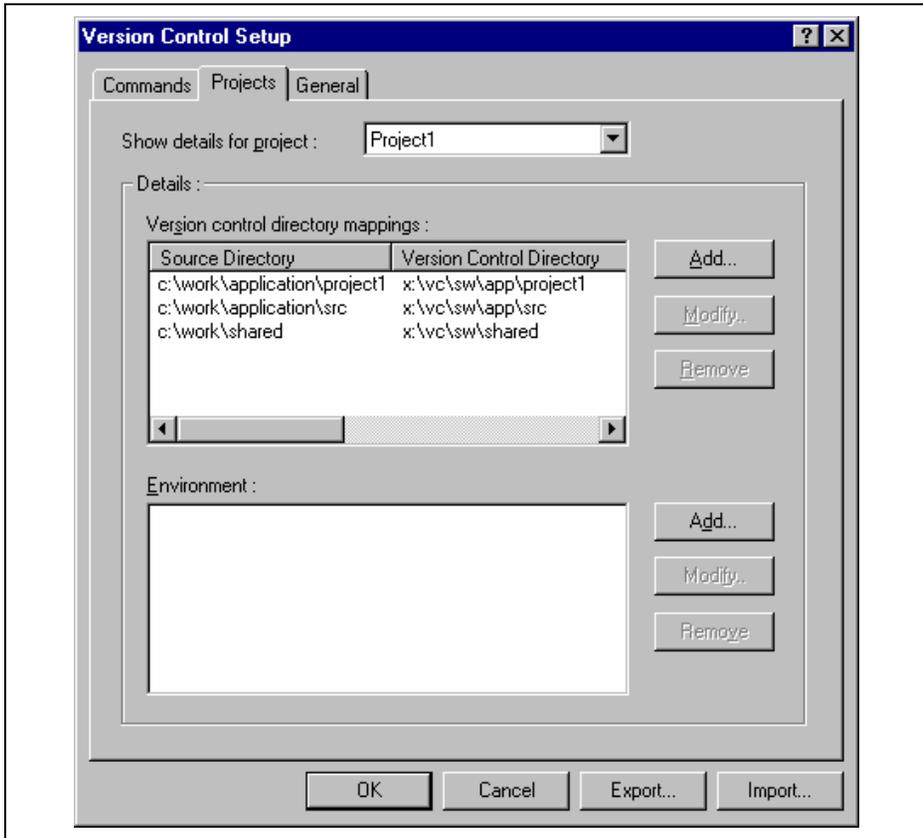


Figure 8.7: Version Control Setup Dialog Projects Tab

- To define a new mapping:
 1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed. Select the “Projects” tab, and the dialog shown in figure 8.7 will be displayed.
 2. Click the “Add...” button that is next to the “Version control directory mappings” list. The dialog shown in figure 8.8 will be displayed.
 3. Enter the source (i.e. “working”) directory into the “Source directory” field or browse to it graphically by clicking the “Browse...” button.
 4. Enter the version control directory (i.e. “controlled”) directory into the “Version control directory”.

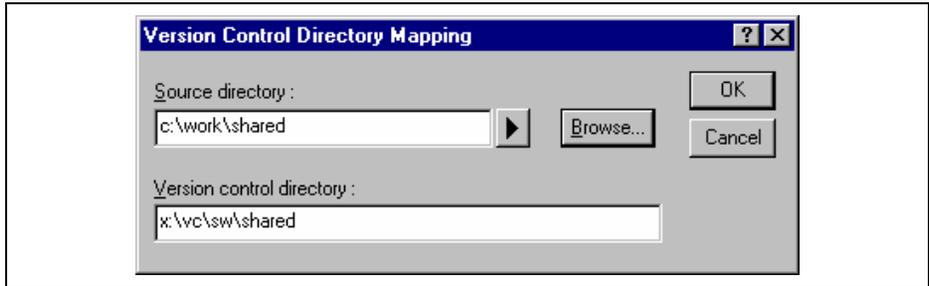


Figure 8.8: Version Control Directory Mapping Dialog

- To modify an existing mapping:
 1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed. Select the “Projects” tab, the dialog shown in figure 8.7 will be displayed.
 2. Select the mapping to be modified from the “Version control directory mappings” list and then click the “Modify...” button. The dialog shown in figure 8.8 will be displayed.
 3. Make the necessary changes to the two directories and then click “OK” to confirm the edits.
- To remove an existing mapping:
 1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed. Select the “Projects” tab, the dialog shown in figure 8.7 will be displayed.
 2. Select the mapping to be removed from the “Version control directory mappings” list and then click the “Remove” button.

Once the mappings have been defined you can use the “Version control directory” placeholder, \$(VCDIR), to represent the directory in which the project file is stored. Consider the scenario shown in figure 8.9. Here are three directories, which are mapped from a shared version control drive (X:\) to a local drive where the development is being done (C:\).

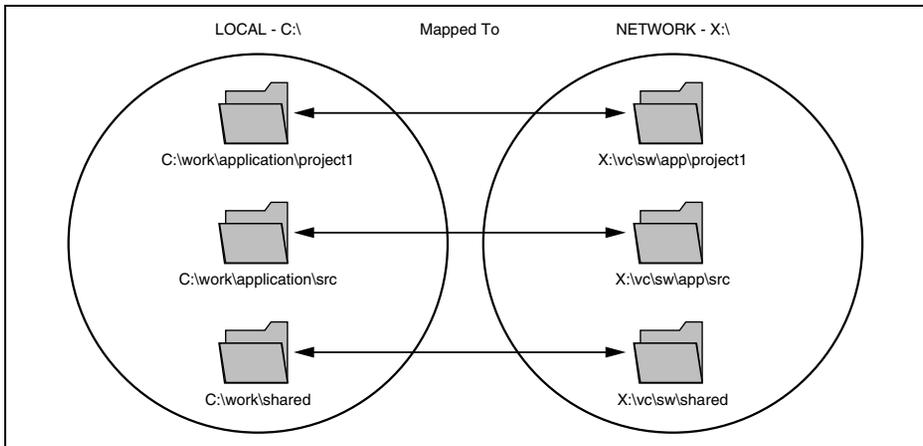


Figure 8.9: Example Mappings

Now let’s imagine that a version control executable has been selected which uses a –GET command to obtain a read only copy of a file. In order to get all of the files in a project we need to use the following command:

```
-GET "$ (VCDIR) \ $ (FILENAME) "
```

When the HEW executes the command for a given project file, it will replace \$(VCDIR) for the equivalent version control directory in the file mapping.

For example, suppose FILE1.C is located at:

```
c:\work\application\project1\file1.c
```

If the get command is applied to FILE1.C then:

- (1) X:\vc\sw\app\project1 is substituted for \$(VCDIR) as this is the version control directory mapping for c:\work\application\project1.
- (2) FILE1.C is substituted for \$(FILENAME).

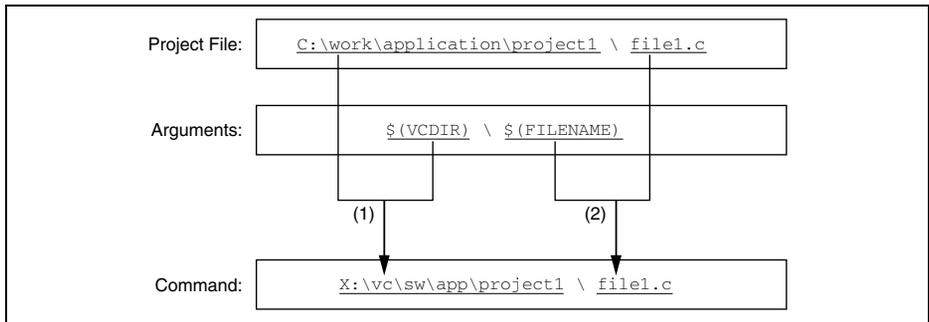


Figure 8.10: Example of Substitution

8.3.2 Specifying Environment

Select the “Projects” tab of the “Version Control Setup” dialog to view the current settings (figure 8.7).

To add a new environment variable click the “Add...” button beside the “Environment” list (the dialog shown in Figure will be invoked). Enter the variable name into the “Variable” field, the variable’s value into the “Value” field and then click “OK” to add the new variable to the “Environment” list.

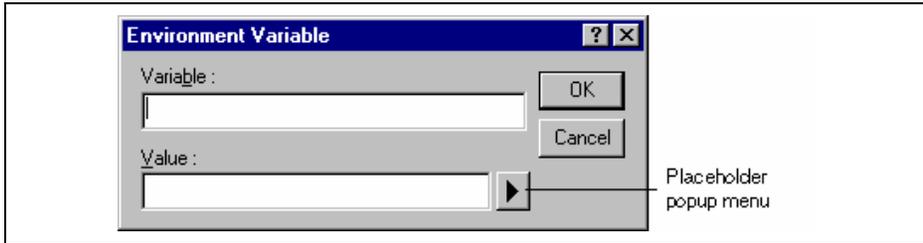


Figure 8.11: Environment Variable Dialog

To modify an environment variable, select the variable that you want to modify from the “Environment” list and then click the “Modify...” button beside it. Make the required changes to the “Variable” and “Value” fields and then click “OK” to add the modified variable back to the list. To remove an environment variable, select the variable that you want to remove from the “Environment” list and then click the “Remove” button beside it.

8.3.3 Specifying Comments

If a command contains the placeholder “\$(COMMENT)” then the HEW will request that you enter the comment when the command is executed (via the dialog as shown in figure 8.12).

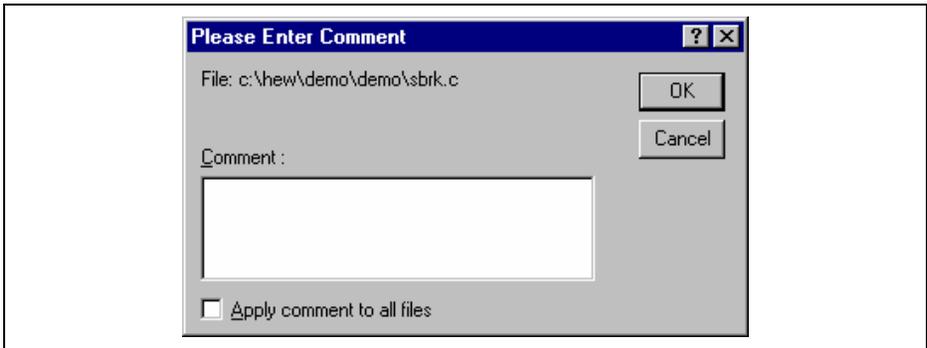


Figure 8.12: Please Enter Comment Dialog

You may specify a comment for each file or, if you would like to specify the same comment for all files, check the “Apply comment to all files” check box before clicking “OK”.

8.3.4 Specifying a User Name and Password

Most version control tools will require you to pass a user name and password on the command line in order to keep files secure and to keep a record of which files were changed by which users. The custom version control support provides two placeholders “User login name”, \$(USERNAME), and “User login password”, \$(PASSWORD). When the command is executed, these placeholders will be replaced with the current settings in the “General” tab of the “Version Control Setup” dialog (figure 8.13).

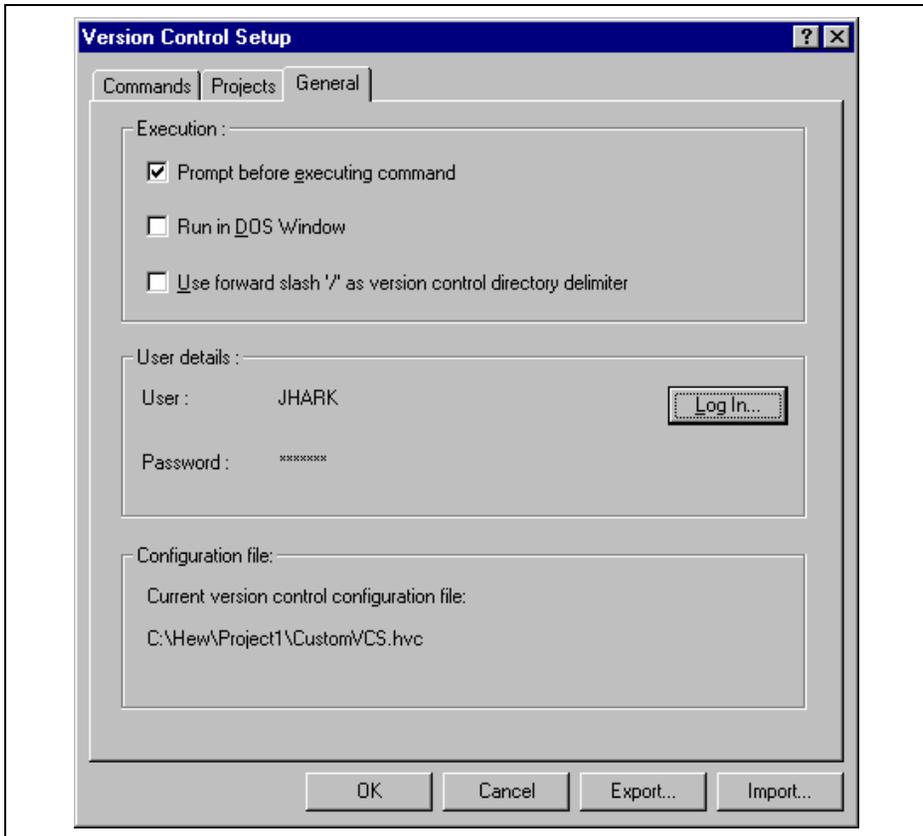


Figure 8.13: Version Control Setup Dialog General Tab

In order to give the \$(USERNAME) and \$(PASSWORD) fields a value you will first need to login. If you have not logged in before a command is executed which uses either of these placeholders then you will be prompted to do so before the command can be executed.

- To login (i.e. specify a user name and password):
 1. Click the “Log in...” button. The dialog shown in figure 8.14 will be displayed.
 2. Enter your user name into the “User name” field.
 3. Enter your password into the “Password” field.
 4. Re-type your password again into the “Confirm password by retyping it below” field.
 5. Click “OK” to set the new user name and password. If there is any inconsistency between the two versions of the password which you entered then you will be requested to type your password again.

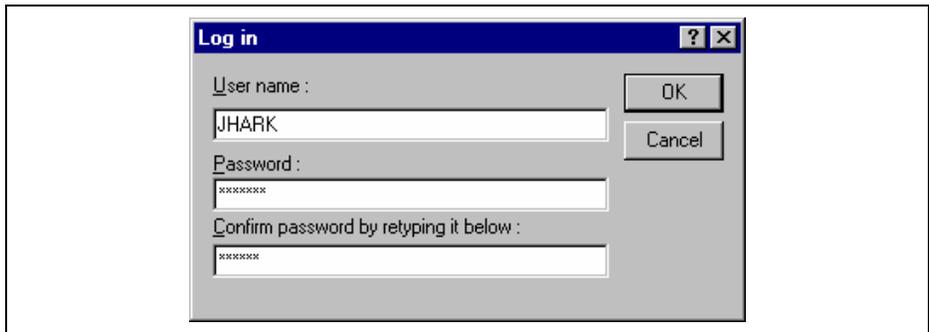


Figure 8.14: Log in Dialog

8.4 Controlling Execution

The “General” tab of the “Version Control Setup” dialog (figure 8.13) allows you to control the way in which the version control tool is executed. It also shows the full path to the current version control configuration file.

8.4.1 Prompt before executing command

If this check box is set then, before any version control commands are executed, a dialog is displayed (figure 8.15) which lists all of the files involved in the operation. Files may be deselected by clearing the associated check box. Clicking “OK” will apply the command to each of the selected files. Clicking “Cancel” will abort the operation.

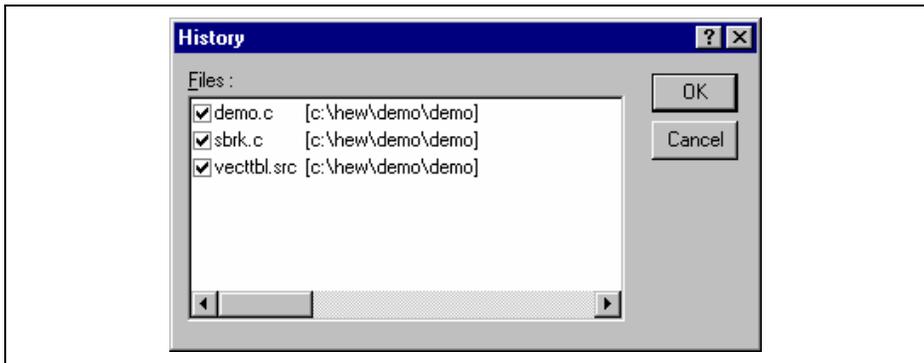


Figure 8.15: Command Prompt Dialog (Example)

8.4.2 Run in DOS Window

By default, the output of the version control commands is redirected to the “Version Control” tab of the “Output” window. If you would rather run each command in a separate DOS window then set this check box.

8.4.3 Use forward slash ‘/’ as version control directory delimiter

By default, when the HEW substitutes the placeholder \$(VCDIR) it uses the backward slash character ‘\’ to divide directories. However, if the version control system you are using uses a forward slash character (e.g. Visual SourceSafe) to divide directories then set the “Use forward slash ‘/’ as version control directory delimiter”.

8.5 Importing and exporting a Set-up

Each workspace can have a different version control set-up. The HEW allows you to store the version control settings independently so that you can import them into other workspaces. This greatly reduces the amount of time it takes to configure the same version control settings across several workspaces.

➡ To export a version control set-up:

1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed.
2. Click the “Export...” button. A standard file save dialog will be displayed. Browse to the directory in which you would like to save the configuration.
3. Enter the name of the file and then click “OK”.

- ➡ To import a version control set-up:
1. Select [**Version Control->Configure...**]. The dialog shown in figure 8.1 will be displayed.
 2. Click the “Import...” button. A standard file open dialog will be displayed. Browse to the *.HVC file which you would like to import.
 3. Select the file and then click “OK”.

9. Using Visual SourceSafe

The High-performance Embedded Workshop provides specific support for the Visual SourceSafe version control system. At the time of writing, the HEW can only attach to versions 5 and 6 of Visual SourceSafe.

The Visual SourceSafe version control system associates a project in your workspace with a project inside a Visual SourceSafe database. It allows you to quickly invoke the standard commands either by selecting an option from the [**Tools->Version Control**] sub-menu or by clicking a version control toolbar button.

9.1 Attaching Visual SourceSafe to a Workspace

The following sections describe how you can associate Visual SourceSafe with your current workspace.

9.1.1 Selecting Visual SourceSafe

First, you need to select Visual SourceSafe as the version control system.

☞ To use Visual SourceSafe 5.0 or 6.0:

1. Select [**Tools->Version Control->Select...**]. The “Select Version Control System” dialog will be displayed (figure 7.3) which lists all of the supported version control systems.
2. Select the “Visual SourceSafe 5.0/6.0” entry from the Version Control Systems list and click the “Select” button.
3. Click “OK” to confirm the selection. The SourceSafe Login dialog is displayed (figure 9.1).
4. Enter your Visual SourceSafe login into “Username” and password into “Password”.
5. Enter into Database path the full path to the Visual SourceSafe database (i.e. SRCSAFE.INI) into which you would like to add this project.
6. Click “OK”. The “Create SourceSafe Project” dialog is invoked (figure 9.2).
7. The “Project name” field displays the name of the project (i.e. folder) to be created in the database. If necessary you can change this name to another.
8. The tree underneath the “Project name” field shows the structure of the database specified in step 6. Select the folder into which you would like to create the folder specified in the “Project name” field.
9. Click “OK”.
10. HEW will require you to repeat steps 7-9 for as many projects as are present in the current workspace.

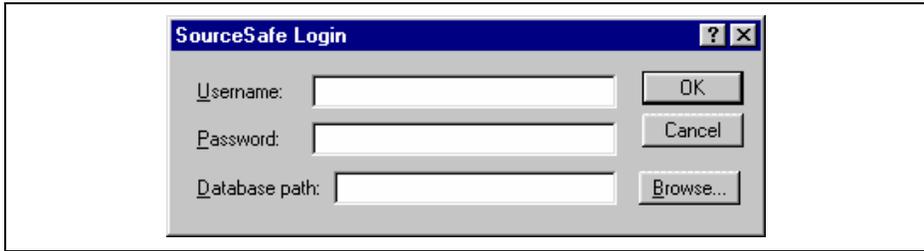


Figure 9.1: SourceSafe Login Dialog



Figure 9.2: Create SourceSafe Project

The HEW has now created the necessary projects within Visual SourceSafe and sets-up the version control toolbar and menu for immediate access. However, although the Visual SourceSafe projects themselves have been created, no files have been added to them.

9.1.2 Adding files to Visual SourceSafe

The previous section has only established the mappings between the project directory on your hard disk (i.e. the working directory) and the project directory in Visual SourceSafe (i.e. the controlled directory). Although the project directory (and any subdirectories) on your hard disk may contain many source files whereas the directly its mapped to in Visual SourceSafe will be initially empty.

Firstly, you must select Visual SourceSafe as the version control system.

- To add a file or files to Visual SourceSafe:

1. Select the file(s), which you would like to add to Visual SourceSafe. You may also select a file folder, project folder, a workspace folder or combination thereof. When selecting the project or workspace folder then the system files will be added to the selected file list. For example, selecting the project folder will also add the project file to the file list. If the project file is then checked out and the version is newer than when it was last loaded you will be asked whether you want to reload the project.
4. Click the Add Files toolbar button () or select the [**Tools->Version Control->Add Files**] menu option.

When you add files to Visual SourceSafe the local versions in your working directory will become read only. To check that the add files operation was carried out as you expected, or to quickly review the status of all of the files in a project:

1. Select the project folder whose files you want to check.
2. Click the Status of Files toolbar button () or select the [**Tools->Version Control->Status of Files**] menu option.
3. The status of each file will be displayed in the “Version Control” tab of the “Output” window. The information shown includes whether the file is added to the project, if the file is checked out and, if it is checked out, who did so.

9.2 Visual SourceSafe commands

The following 8 operations are available:

- Add a file to version control
- Remove a file from version control
- Get a read only copy of a file or files
- Check out a read/write copy of a file or files (i.e. for editing)
- Check in a previously checked out file or files (i.e. update Visual SourceSafe with the edits made)
- Undo a previously check out operation on a file or files (i.e. cancel any edits made)*
- View the status of a file
- View the history of a file*

*These commands can only be accessed via the **[Tools->Version Control]** sub-menu whereas all of the other commands can be accessed from both the toolbar and menu.

9.2.1 Removing a File from Version Control

Although files appear in your HEW project (in the “Projects” tab of the “Workspace” window, Visual SourceSafe is not necessarily controlling them.

- ☞ To remove a file or files from Visual SourceSafe:
 1. Select the file(s), which you would like to remove from Visual SourceSafe. You may also select a file folder, project folder, a workspace folder or combination thereof.
 2. Click the Remove Files toolbar button () or select the **[Tools->Version Control->Remove Files]** menu option.

9.2.2 Getting a Read Only Copy of a File from Version Control

Visual SourceSafe protects your source files and ensures that only one user can have a writable copy of a controlled file at any one time. However, it is possible for any user to obtain a read only copy of any file.

- ☞ To get a read only copy of a file or files from Visual SourceSafe:
 1. Select the file(s), which you would like to get from Visual SourceSafe. You may also select a file folder, project folder, a workspace folder or combination thereof.
 2. Click the Get Files toolbar button () or select the **[Tools->Version Control->Get Files]** menu option.

9.2.3 Checking Out a Writable Copy of a File from Version Control

Visual SourceSafe protects your source files and ensures that only one user can have a writable copy of a controlled file at any one time. The check out operation takes a writable copy of the file from Visual SourceSafe and places it on your local drive. This can only be done if another user does not already check out the file or files in question.

- To check out a writable copy of a file or files from Visual SourceSafe:
 1. Select the file(s), which you would like to check out from Visual SourceSafe. You may also select a file folder, project folder, a workspace folder or combination thereof.
 2. Click the Check Out Files toolbar button () or select the **[Tools->Version Control->Check Out Files]** menu option.

9.2.4 Checking In a Writable Copy of a File into Version Control

Visual SourceSafe protects your source files and ensures that only one user can have a writable copy of a controlled file at any one time. The check out operation takes a writable copy of the file from Visual SourceSafe and places it on your local drive. Once a file is checked out it is edited and then checked back in so that the edits can be made available to other users.

- To check in edits made to a file or files in Visual SourceSafe:
 1. Select the file(s) upon which you would like to check back into Visual SourceSafe. You may also select a file folder, project folder, a workspace folder or combination thereof.
 2. Click the Check In Files toolbar button () or select the **[Tools->Version Control->Check In]** menu option.

9.2.5 Undoing a Check Out Operation

Visual SourceSafe protects your source files and ensures that only one user can have a writable copy of a controlled file at any one time. The check out operation takes a writable copy of the file from Visual SourceSafe and places it on your local drive. Once a file is checked out it is edited and then checked back in so that the edits can be made available to other users. However, if the check out operation was carried out by mistake, or perhaps is no longer required, then the operation can be undone.

- To undo a check out of a file or files from Visual SourceSafe:
 1. Select the file(s) upon which you would like to undo a previous check out operation. You may also select a file folder, project folder, a workspace folder or combination thereof.
 2. Select the **[Tools->Version Control->Undo Check Out]** menu option.

9.2.6 Viewing the Status of a File

Although files appear in your HEW project (in the Projects tab of the Workspace window), Visual SourceSafe is not necessarily controlling them. Of those files, which are being controlled by Visual SourceSafe, some will be checked in and others will be checked out (i.e. being edited by a user). The status command displays the current status of a file or file(s).

- ☞ To view the status of a file or files in Visual SourceSafe:
 1. Select the file(s) whose status you would like to view. You may also select a file folder, project folder, a workspace folder or combination thereof.
 2. Click the Status of Files toolbar button () or select the [Tools->Version Control->Status of Files] menu option.

9.2.7 Viewing the History of a File

Visual SourceSafe controls the edits to the files in its projects and allows you to view the complete history of these edits right back to the time that the file was first added to the project.

- ☞ To view the history of a file or files in Visual SourceSafe:
 1. Select the file(s) whose history you would like to view. You may also select a file folder, project folder, a workspace folder or combination thereof.
 2. Select the [Tools->Version Control->Show History] menu option.

9.3 Visual SourceSafe Integration Options

You can control the way in which the history and status commands are displayed by selecting **[Tools->Version Control->Configure...]**.

To display the results of a history command in a dialog box then check the “Display dialog box for history” check box or clear it if you would rather display the output in the “Version Control” tab of the “Output” window. To display the results of a status command in a dialog box then check the “Display dialog box for file status” check box or clear it if you would rather display the output in the “Version Control” tab of the “Output” window.

Debugger Part

Section 1 Overview

The High-performance Embedded Workshop (HEW) is a graphical user interface intended to ease the development and debugging of applications written in C/C++ programming language and assembly language for 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.

The E10A emulator (hereafter referred to as the emulator) is a software and hardware development support tool for application systems using the Renesas original microcomputer.

The PCMCIA card emulator or PCI card emulator (hereafter referred to as the card emulator), which is the main unit of the emulator, is connected, through the H-UDI port*, to the user system. The user system can be debugged under the conditions similar to the actual application conditions. The emulator enables debugging anywhere indoors or out. The host computer for controlling the emulator must be an IBM PC compatible machine with a PCMCIA type II or PCI slot.

Figures 1.1 and 1.2 show the system configuration using the emulator.

Note: The H-UDI is an interface compatible with the Joint Test Action Group (JTAG) specifications.

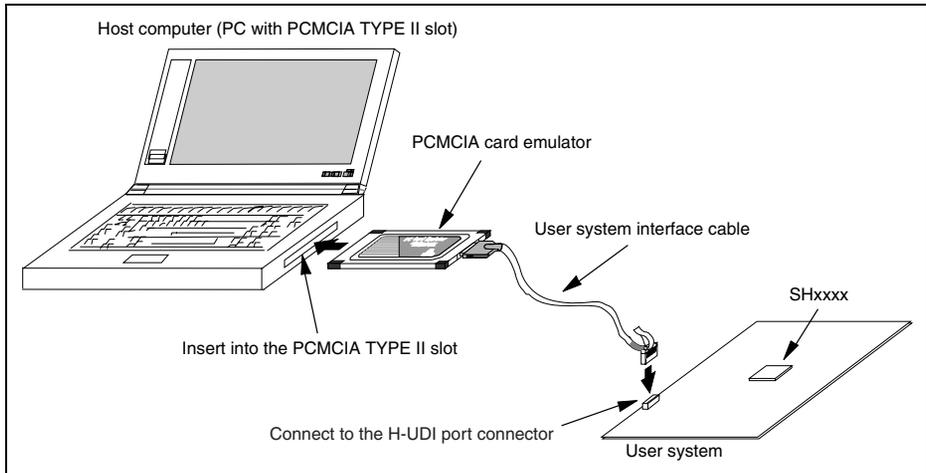


Figure 1.1 System Configuration with the Emulator (PCMCIA Card Emulator Used)

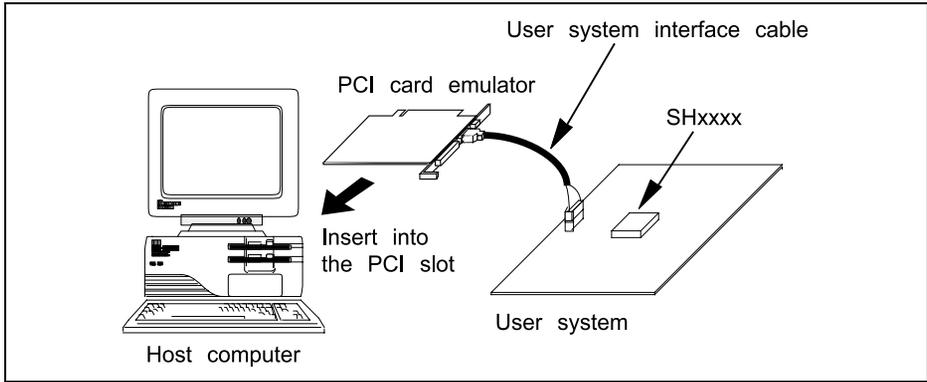


Figure 1.2 System Configuration with the Emulator (PCI Card Emulator Used)

The emulator provides the following features:

- Excellent cost-performance card emulator
Compactness and low price are obtained through the use of a PCMCIA or PCI interface.
- Realtime emulation
Realtime emulation of the user system is enabled at the maximum operating frequency of the CPU.
- Excellent operability
Using the HEW2 (High-performance Embedded Workshop 2) on the Microsoft® Windows® 98, Microsoft® Windows® Me, Microsoft® Windows® 2000, Microsoft® Windows® XP, and Microsoft® Windows NT® operating systems enables user program debugging using a pointing device such as a mouse. The HEW2 enables high-speed downloading of load module files.
- Various debugging functions
Various break and trace functions enable efficient debugging. Breakpoints and break conditions can be set by the specific window, trace information can be displayed on a window, and command-line functions can be used.
- Debugging of the user system in the final development stage
The user system can be debugged under conditions similar to the actual application conditions.
- Compact debugging environment
When the card emulator specific to the PCMCIA interface is used, a laptop computer can be used as a host computer, creating a debugging environment in any place.
- AUD trace function*
The AUD trace function enables realtime trace.

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

1.1 Warnings

CAUTION

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

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

1.2 Environmental Conditions

CAUTION

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

Table 1.1 Environmental Conditions

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

Table 1.2 lists the acceptable operating environments.

Table 1.2 Operating Environments

Item	Description
Host computer	Built-in Pentium® III or higher-performance CPU (600 MHz or higher recommended); IBM PC or compatible machine with the PCMCIA TYPE II slot or the PCI slot.
OS	Windows® 98, Windows® Me, Windows® 2000, Windows® XP, or Windows NT®
Minimum memory capacity	128 Mbytes or more (double of the load module size recommended)
Hard-disk capacity	Installation disk capacity: 50 Mbytes or more. (Prepare an area at least double the memory capacity (four-times or more recommended) as the swap area.)
Pointing device such as mouse	Connectable to the host computer; compatible with Windows® 98, Windows® Me, Windows® 2000, Windows® XP, and Windows NT®.
Power voltage	5.0 ± 0.25 V
Current consumption	HSxxxKCM01H: 110 mA (max) HSxxxKCM02H: 230 mA (max) HSxxxKCI01H: 340 mA (max) HSxxxKCI02H: 600 mA (max)
CD-ROM drive	Required to install the HEW for the emulator or refer to the emulator user's manual.

1.3 Components

Check all the components unpacking. For details on the E10A emulator components, refer to section 1.1 in the additional document, Specific Guide for the SHxxxx E10A Emulator. If the components are not complete, contact our E-mail address for user registration or refer to the web site.

Section 2 E10A Emulator Functions

This section describes the emulator functions. They differ according to the device supported by the emulator. For the usage of each function, refer to section 6, Tutorial.

2.1 Overview

Table 2.1 gives a functional overview of the emulator.

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

Table 2.1 Emulator Functions

No.	Item	Function
1	User program execution functions	<ul style="list-style-type: none"> • Executes a program with the operating frequency within a range guaranteed by devices. • Reset emulation • Step functions: <ul style="list-style-type: none"> Single step (one step: one instruction) Source-level step (one step: one-line source) Step over (a break did not occur in a subroutine) Step out (executed until a program is returned to the original call function in a subroutine where a PC is being executed)
2	Reset function	<ul style="list-style-type: none"> • Issues a power-on reset from the HEW to the device during break.
3	Trace functions	<ul style="list-style-type: none"> • Trace function incorporated in the device • AUD trace: <ul style="list-style-type: none"> Branch trace or memory access trace
4	Break functions	<ul style="list-style-type: none"> • Hardware break condition (conditions and the number of conditions differ according to the device) • PC break condition (255 points) • Forced break function

Table 2.1 Emulator Functions (cont)

No.	Item	Function
5	Performance measurement functions	<ul style="list-style-type: none"> • Uses a counter in the device to measure the number of cycles that passes during point-to-point execution. • Measures the number of cycles that pass in executing individual functions and lists them at the end of execution from a 'Go' command.
6	Memory access functions	<ul style="list-style-type: none"> • Downloading to RAM • Downloading to flash memory • Single-line assembly • Reverse assembly (disassembly) • Reading of memory • Writing to memory • Automatic updating of a display of selected variables during user program execution • FILL • Search • Move • Copy
7	General/control register access function	Reads or writes the general/control register.
8	Internal I/O register access function	Reads or writes the internal I/O register.
9	Source-level debugging function	Various source-level debugging functions.
10	Command line function	<p>Supports command input.</p> <p>Batch processing is enabled when a file is created by arranging commands in input order.</p>
11	Help function	Describes the usage of each function or command syntax input from the command line window.

The specific functions of the E10A emulator are described in the next section.

2.2 Trace Functions

The E10A emulator has two trace functions.

2.2.1 Internal Trace Function

The branch source and branch destination addresses, mnemonics, operands, and source lines are displayed. Since this function uses the trace buffer built into the device, a realtime trace can be acquired.

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

2.2.2 AUD Trace Function

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

(1) Trace acquisition event

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

(a) Branch generation information

The branch source and branch destination addresses are acquired.

(b) Memory access information within the specified range

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

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

This function is called the window trace function.

(c) Software trace

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

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

- Notes:
1. This function can be supported with SHC/C++ compiler V6.0 or later for SH4 series, and with SHC/C++ compiler V7.0 or later for SH3 series.
 2. The types of events acquired by a trace differ according to the product. For the specification of each product, refer to the online help.

(2) Trace acquisition mode

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

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

Table 2.2 AUD Trace Acquisition Mode

Type	Mode	Description
Continuous trace occurs	Realtime trace	When the next branch occurs while the trace information is being output, all the information may not be output. The user program can be executed in realtime, but some trace information may be lost.
	Non realtime trace	When the next branch occurs while the trace information is being output, the CPU stops operations until the information is output. The user program is not executed in realtime.
Trace buffer full	Trace continue	This function overwrites the oldest trace information to store the latest trace information.
	Trace stop	After the trace buffer becomes full, the trace information is no longer acquired. The user program is continuously executed.

(3) Trace display contents

After the program execution has stopped, the following trace results are displayed in the [Trace] window.

- PTR: Pointer to a location in the trace buffer (+0 for the last executed instruction)
- IP: Indicates the number of steps that have been passed after the latest information has been traced. The branch information is counted as one in the branch source and destination.
- Type: Displays the type of trace acquisition information.
- Address: Displays the trace acquisition address.
- Data: Displays the trace acquisition data. For information without data, displays '*****'.
- Instruction, Source, Label: Displays the mnemonic, source, or label information of the trace acquisition address. When the [Source] column is double-clicked, the cursor moves to the corresponding position in the editor window.

The Type, Address, and Data columns have different meanings according to the selected AUD trace types.

Table 2.3 [Trace Window] Display Contents

Trace Type	TYPE Column	ADDR Column	DATA Column
Branch trace	BRANCH	Branch source address	No display
	DESTINATION	Branch destination address	No display
Window trace ¹	MEMORY	Memory access address	Memory access data
Software trace ¹	S_TRACE	Trace(x) function execution address	Variable x data
Data lost ^{1,2}	LOST	No display	No display
CPU wait generation ^{1,2}	CPU-WAIT	No display	No display

Notes: 1. Not displayed in the internal trace.

2. Not output [Lost] or [CPU-WAIT] depending on the device to be debugged. Note that it is not clear that the trace data has not been output in time or the CPU has generated a wait to output the trace data.

(4) Useful functions of the trace window

The trace window provides the following useful functions.

- Searches for the specified data.
- Extracts the specified data.
- Filters and displays again the specified data.
- Supplements the information from the branch destination address to the next branch source address.

For the usage of those functions, refer to section 5.14, Viewing the Trace Information.

2.3 Break Function

The E10A emulator has the following three break functions.

(1) Hardware break function

Uses a break controller incorporated in the device.

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

This function can be also set from the EVENT column in the editor window. For the setting, refer to section 5.2, Viewing a Program.

(2) PC break function (BREAKPOINT)

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

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

(3) Forced break function

Forcibly breaks the user program.

2.4 Performance Measurement Function

The emulator has three types of performance measurement functions.

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

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

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

This function is hereafter called the performance measurement function or PA1.

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

2.4.2 Profiling Function

The profiling function is used to measure the performance of each function.

A function having bad performance can be easily found if the statistics of the time for each function are maintained.

- Notes:
1. Use of the profiling and performance measurement functions at the same time is not possible. The [Can not use this function] error message dialog box will be displayed if simultaneous use is attempted.
 2. Items to be measured differ according to the product and some products do not support this function. For the specification of each product, refer to the online help.

2.5 Memory Access Functions

The emulator has the following memory access functions.

(1) Memory read/write function

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

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

(2) User program downloading function

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

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

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

(3) Memory data uploading function

The specified amount of memory from the specified address can be saved in an S-format file.

(4) Memory data downloading function

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

(5) Displaying the variable contents

The variable contents specified in the user program are displayed. For the usage of the function for displaying the variable contents, refer to section 5.12, Looking at Variables.

(6) Other memory operation functions

Other functions are as follows:

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

For details, refer to the online help.

Notes: 1. Memory access during user program execution:

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

The stopping time of the user program is as follows:

Environment:

Host computer: 800 MHz (Pentium® III)

SH7710: 66.67 MHz (CPU clock) (mode 1. FRQCR = H'1003, 16.67-MHz input clock)

JTAG clock: 3.75 MHz

When a one-byte memory is read in the command-line window, the stopping time will be about 32 ms.

2. Memory access during user program break:

The program can also be downloaded for the flash memory area by the emulator.

Other memory write operations are enabled for the RAM area. Therefore, an operation such as memory write or BREAKPOINT should be set only for the RAM area and the internal flash memory. When the memory area can be read by the MMU, do not perform memory write, BREAKPOINT setting, or downloading.

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

2.6 Stack Trace Function

The emulator uses the stack's information to display the name of the calling function for a function at which the program counter is currently pointing. This function can be used only when the load module that has the Dwarf2-type debugging information is loaded. For the usage of this function, refer to section 6.20, Stack Trace Function.

2.7 Online Help

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

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

Section 3 Preparation before Use

3.1 Emulator Preparation

Unpack the emulator and prepare it for use as follows:



WARNING

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

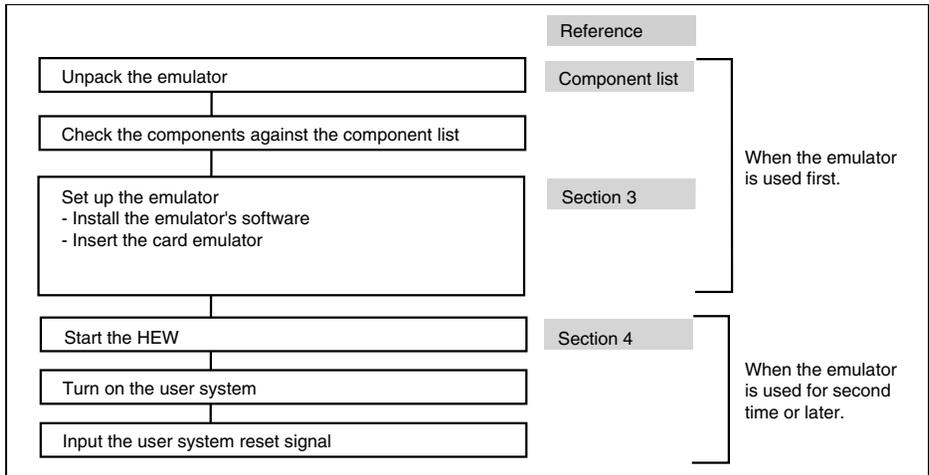


Figure 3.1 Emulator Preparation Flow Chart

3.2 Installing Emulator's Software

Execute Setup.exe from the root directory of the CD-R.

Follow the cues given by the installation wizard to install the software.

Since hardware settings are also made during installation, the installation procedure differs according to the operating system or interface (PCI or PCMCIA) being used. Follow the installation steps carefully according to the environment you are using.

3.2.1 Installing under Windows® 98 or Windows® Me Operating System

(1) When the emulator is a PCI card:

1. Install the HEW2 (when the component type has to be selected during installation, be sure to select [PCI Card Driver]).
2. Shut the operating system down and turn off the power to the host computer.
3. Insert the PCI-card emulator in a slot on the host computer. Refer to section 3.3, Connecting the Card Emulator to the Host Computer.
4. Restart the host computer. The hardware is now recognized and the driver is automatically installed.*

(2) When the emulator is a PCMCIA card:

1. Install the HEW2 (when the component type has to be selected during installation, be sure to select [PC Card Driver (PCMCIA)]).
2. Insert the PCMCIA-card emulator in the host computer's slot. Refer to section 3.3, Connecting the Card Emulator to the Host Computer.
3. The hardware is now recognized and the driver is automatically installed.*

Note: When [Add New Hardware Wizard] is displayed, select the [Search for the best driver for your device. (Recommended)] radio button and then the [Specify a location] check box to select the path to be searched for drivers. The location must be specified according to the emulator type, as indicated below:

When using the PCI-card emulator: <Drive>\DRIVERS\PCI\95

When using the PCMCIA-card emulator: <Drive>\DRIVERS\PCMCIA\95
(<Drive> is the CD-ROM drive name.)

3.2.2 Installing under Windows NT® 4.0 Operating System

(1) When the emulator is a PCI card:

1. Shut the operating system down and turn off the power to the host computer.
2. Insert the PCI-card emulator in a slot on the host computer. Refer to section 3.3, Connecting the Card Emulator to the Host Computer.
3. Start the host computer and log-on with an administrator-level user name.
4. Install the HEW2. (For a component, be sure to select [PCI Card Driver]. There is a check box for selecting the type name of the product under the [PCI Card Driver] component. Select the appropriate type name. If the correct name is not selected, the correct driver will not be installed, and the emulator will not operate.)
5. Restart the host computer.

(2) When the emulator is a PCMCIA card:

1. Shut the operating system down and turn off the power to the host computer.
2. Insert the PCMCIA-card emulator in the host computer's slot. Refer to section 3.3, Connecting the Card Emulator to the Host Computer.
3. Start the host computer and log-on with an administrator-level user name.
4. During HEW2 installation, the setting value should be checked beforehand because inquiries are made about the resource used by the PCMCIA-card emulator. Start the [Start] menu -> [Programs] -> [Administrative Tools (Common)] -> [Windows NT Diagnostics], check the status of the IRQ, I/O port, and memory from the resource panel, and determine the setting values that do not conflict with other devices. (The following resources are used: IRQ: one channel, I/O port: H'F bytes, and memory: H'4000 bytes.)
5. Install the HEW2. (For a component, be sure to select [PC Card Driver (PCMCIA)]. There is a check box for selecting the type name of each product under the [PC Card Driver (PCMCIA)] component. Select the appropriate type name. If the correct name is not selected, the correct driver will not be installed and the emulator will not operate.)
6. Restart the host computer.

- Notes:
1. For the SH7729, SH7729R, and SH7622 E10A emulators, there is a check box for selecting the MODEL name that appears on the component list. Select the correct type name.
 2. The driver that has been selected in the [Drivers] component starts after the host computer is initiated. If the host computer is initiated with the card disconnected or with the incorrect driver installed, the driver cannot initiate and the service control manager informs the system of an error. This, however, is not a problem.

3.2.3 Installing under Windows® 2000 or Windows® XP Operating System

- (1) When the emulator is a PCI card:
 1. Log-on with an administrator-level user name.
 2. Install the HEW2. (When a component is selected, be sure to select [PCI Card Driver].)
 3. Shut the operating system down and turn off the power to the host computer.
 4. Insert the PCI-card emulator in a slot on the host computer. Refer to section 3.3, Connecting the Card Emulator to the Host Computer.
 5. Restart the host computer and log-on with an administrator-level user name. The hardware is now recognized and the driver is automatically installed.*
- (2) When the emulator is a PCMCIA card:
 1. Log-on with an administrator-level user name.
 2. Install the HEW2. (When a component is selected, be sure to select [PC Card Driver (PCMCIA)].)
 3. Insert the PCMCIA-card emulator in the host computer's slot. Refer to section 3.3, Connecting the Card Emulator to the Host Computer.
 4. The hardware is now recognized and the driver is automatically installed.*

Note: When [Found New Hardware Wizard] is displayed, select the [Search for a suitable driver for my device (recommended).] radio button and then the [Specify a location] check box to select the path to be searched for drivers. The location must be specified according to the emulator type, as indicated below:

When using the PCI-card emulator: <Drive>\DRIVERS\PCI\2000

When using the PCMCIA-card emulator: <Drive>\DRIVERS\PCMCIA\2000

(<Drive> is the CD-ROM drive name.)

3.3 Connecting the Card Emulator to the Host Computer

Insert the card emulator, according to its type, in a PCMCIA TYPE II slot or PCI slot on the host computer (figures 3.2 and 3.3).

Note: When using Windows[®]98, Windows[®]Me, Windows[®]2000, Windows[®]XP, or Windows NT[®], be sure to install the emulator software before putting the card emulator in place.

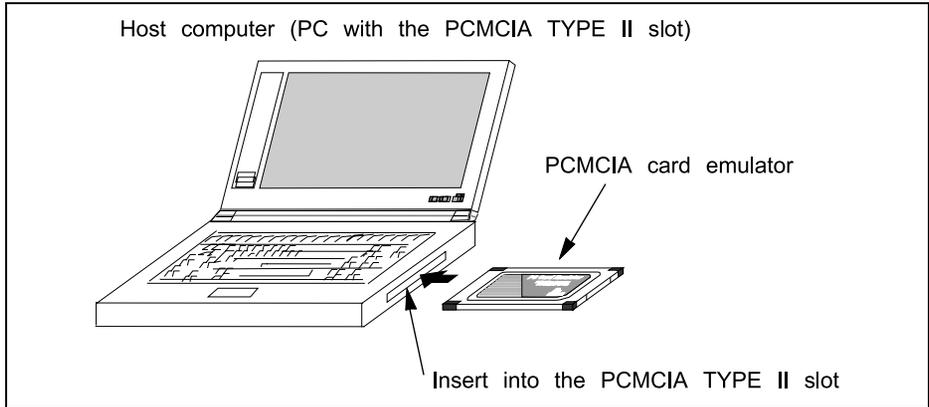


Figure 3.2 Inserting the PCMCIA Card Emulator in the Host Computer's Slot

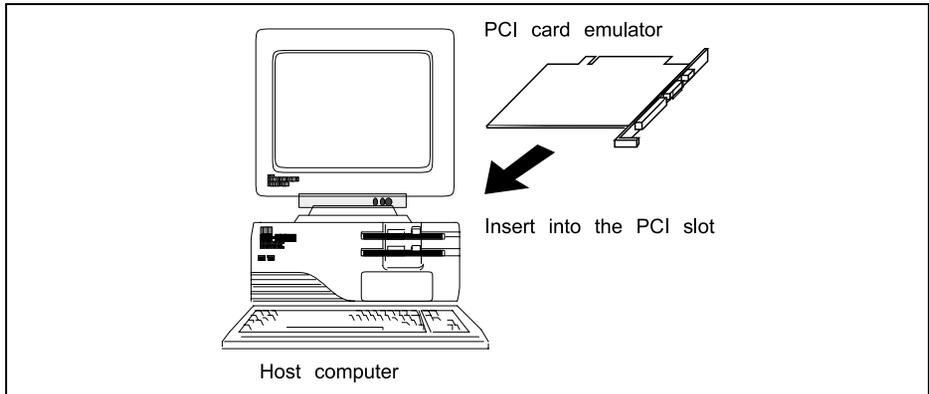


Figure 3.3 Inserting the PCI Card Emulator in the Host Computer's Slot

Use the procedure, described in section 3.4, to connect the emulator to the user system with the user system interface cable, or to disconnect them when moving the emulator or the user system.

WARNING

**When inserting the PCI-card emulator, note the following.
Failure to do so will damage the host computer.**

- 1. Turn off the host computer.**
- 2. Insert the emulator into the PCI slot in parallel.**
- 3. Screw in the emulator after checking the connector and cable positions.**

3.4 Connecting the Card Emulator to the User System

(1) The H-UDI port connector must be installed to the user system. Table 3.1 shows the recommended H-UDI port connector for the emulator.

Table 3.1 Recommended H-UDI Port Connector

Connector	Type Number	Manufacturer	Specifications
14-pin connector	2514-6002	Minnesota Mining & Manufacturing Ltd.	14-pin straight type
36-pin connector	DX10M-36S	Hirose Electric Co., Ltd.	Screw type
	DX10M-36SE,		Lock-pin type
	DX10GM-36SE		

Note: When the 14-pin connector is used, do not install any components within 3 mm of the H-UDI port connector.

When the 36-pin connector is used, do not connect other signal lines to the H-UDI port connector.

(2) Note that the TDO signal of the user system interface cable connector must be connected to the TDI pin of the H-UDI port connector and the TDI signal of the user system interface cable connector must be connected to the TDO pin of the H-UDI port connector. The pin arrangement of the H-UDI port connector is shown in section 1.4 in the additional document, Specific Guide for the SHxxxx E10A Emulator.

- (3) Figure 3.4 shows how to connect the user system interface cable to the user system when the 14-pin straight type connector is used. Connect the ground line of the cable to the user system ground. The end of the ground line has a hole having a diameter of 3 mm, and therefore, when the ground line is screwed to the user system, the screw diameter must be 3 mm.

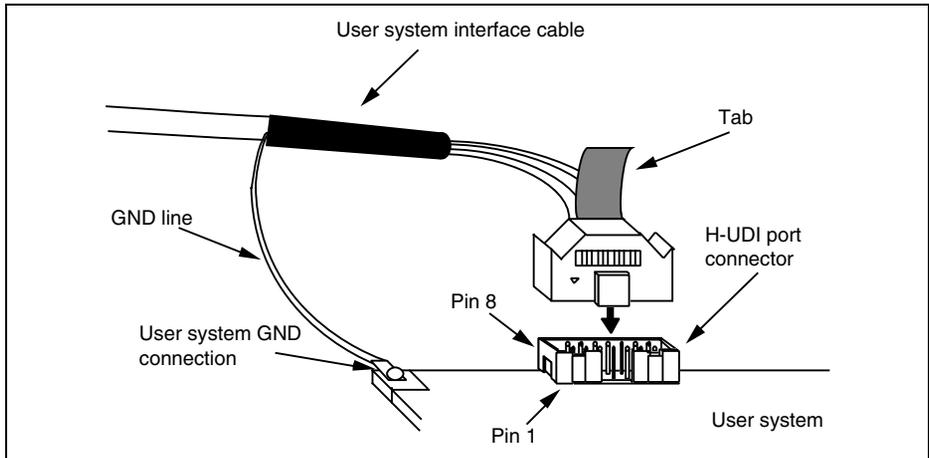


Figure 3.4 Connecting the User System Interface Cable to the User System when the 14-pin Straight Type Connector is Used

- Notes:
1. To connect the signals output from the H-UDI port connector, refer to the device pin alignment.
 2. To remove the user system interface cable from the user system, pull the tab on the connector upward.
 3. The range of frequencies that the H-UDI operates at is different according to the devices used. For details, refer to section 2.2.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK), in the Specific Guide for the SHxxxx E10A Emulator.
 4. Connect the H-UDI signals from the H-UDI port connector directly to the device.
 5. When developing user systems, do not connect the TDI and TDO signals of the device to the boundary scan loop, or separate them by using a switch (figure 3.5).

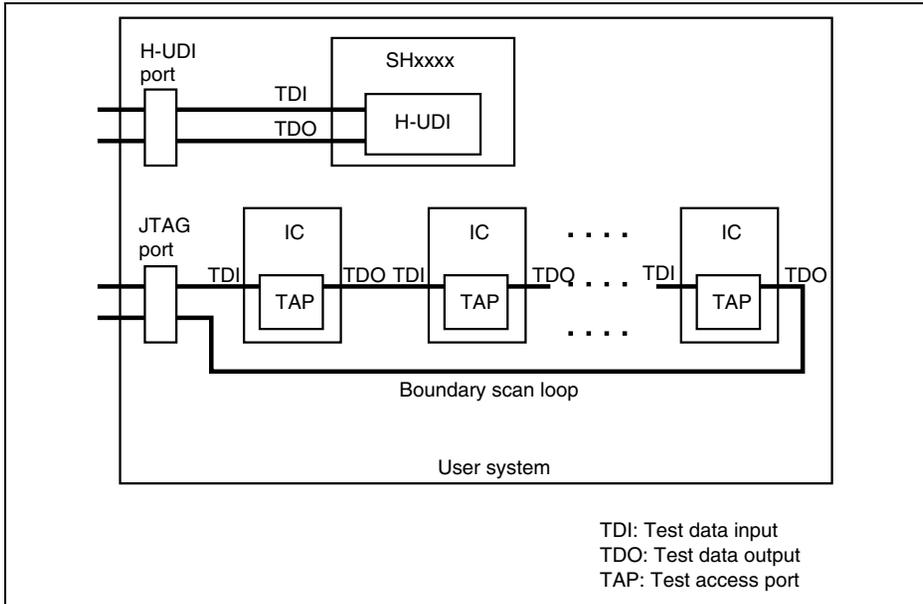


Figure 3.5 User System Example

3.5 System Check

When the software is executed, use the procedure below to check that the emulator is connected correctly. Here, use the workspace for a tutorial provided on the product.

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

1. Connect the emulator to the host computer.
2. Connect the user system interface cable to the connector of the emulator.
3. Connect the user system interface cable to the connector in the user system.
4. Turn on the host computer.
5. Select [Renesas High-performance Embedded Workshop 2] -> [High-performance Embedded Workshop 2] from the [Programs] in the [Start] menu.

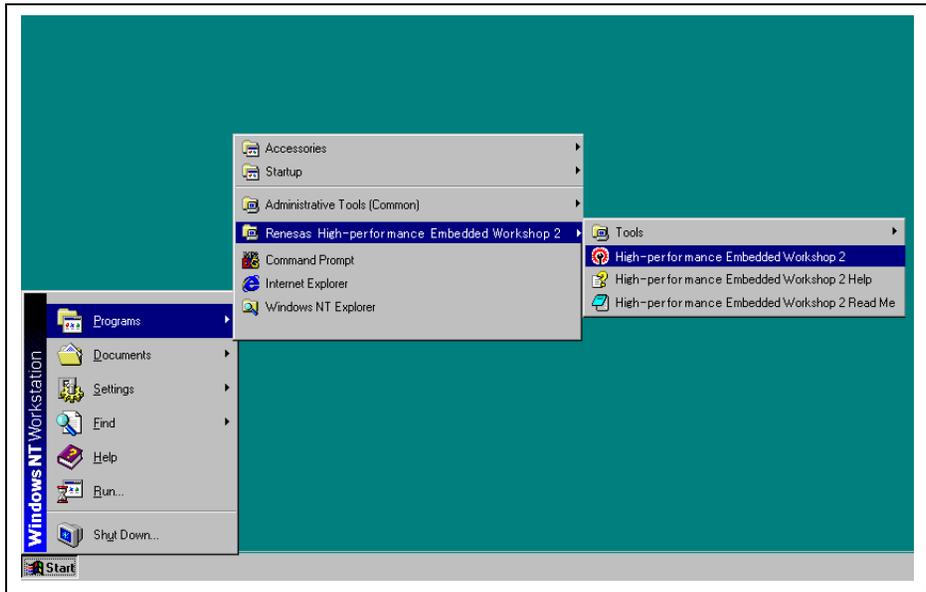


Figure 3.6 [Start] Menu

Note: The [Renesas High-performance Embedded Workshop 2] -> [Tools] is not displayed depending on the user's environment.

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

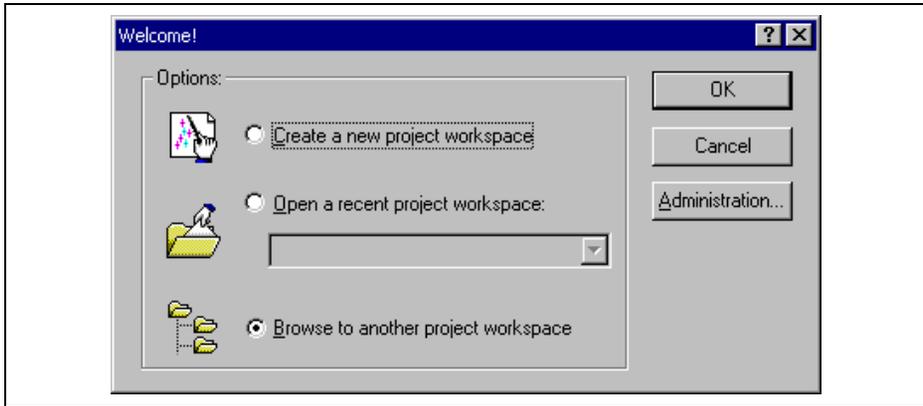


Figure 3.7 [Welcome!] Dialog Box

[Create a new project workspace] radio button: Creates a new workspace.

[Open a recent project workspace] radio button: Uses the current workspace and displays the history of the opened workspace.

[Browse to another project workspace] radio button: Uses the current workspace; this radio button is used when the history of the opened workspace does not remain

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

When the [Open workspace] dialog box is opened, specify the following directory:
<HEW2 installation directory>\Tools\Renesas\DebugComp\Platform\E10A\xxxx\Tutorial

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

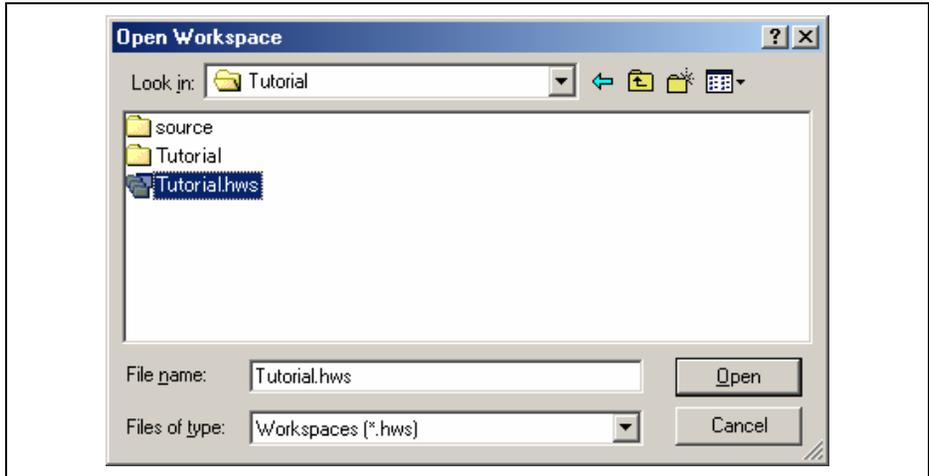


Figure 3.8 [Open Workspace] Dialog Box

7. The [Select Emulator mode] dialog box is displayed depending on the MCU used.



Figure 3.9 [Select Emulator mode] Dialog Box

Select the MCU name in use from the [Device] drop-down list box. The following items are selected in the [Mode] group box.

- E10A-USB Emulator

The E10A-~~USB~~ emulator for the specified MCU is activated. Debugging the program is enabled.

— Writing Flash memory

The user program is programmed to the flash memory. Debugging the program is disabled.
To download the load module, register it in the workspace.

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

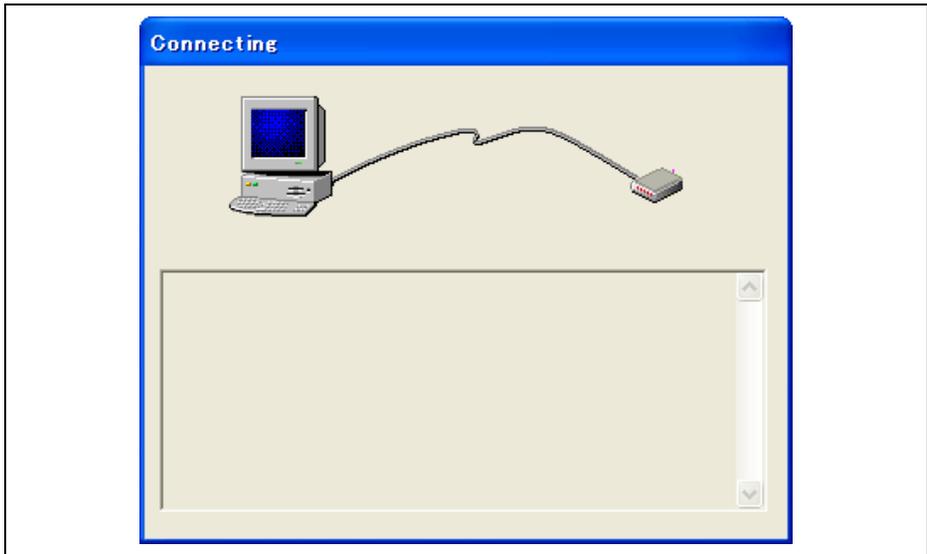


Figure 3.10 [Connecting] Dialog Box

9. The [E10A Driver Details] dialog box is displayed. With the [Driver] combo box, select the driver to connect the HEW2 with the emulator. [Interface] displays the interface name of the PC interface board to be connected, and [Channel] displays the interface to which the board is connected. Once the driver is selected in the [E10A Driver Details] dialog box, this dialog box is not displayed when the HEW2 is run next time. (This procedure will not be executed by target devices.)

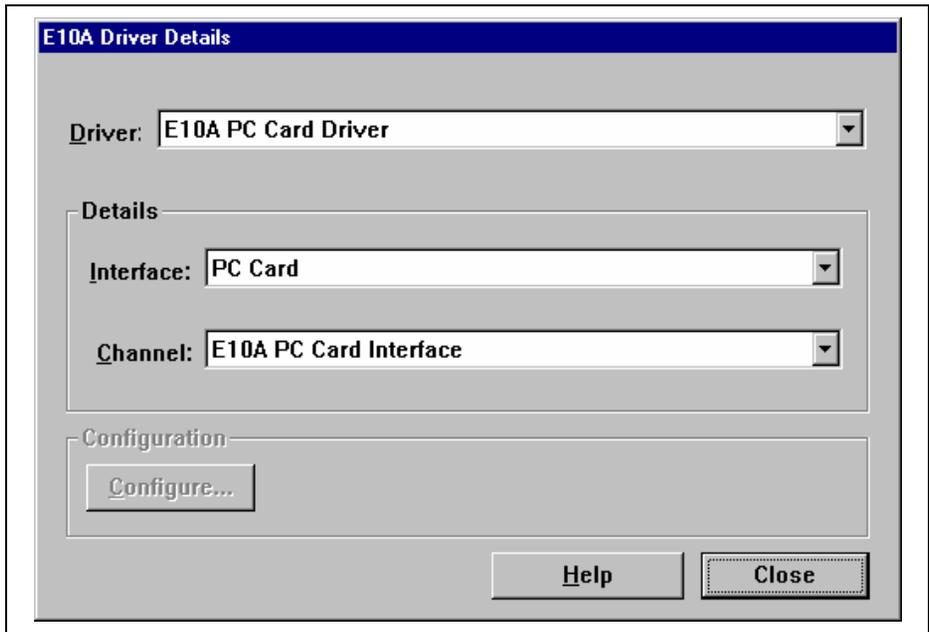


Figure 3.11 [E10A Driver Details] Dialog Box

- With the [Driver] combo box, select the driver to connect the HEW2 with the emulator.
- [Interface] displays the interface name of the card emulator to be connected, and [Channel] displays the interface to which the board is connected.
 - [Driver] combo box: Select [E10A PC Card Driver] to use the PCMCIA card emulator. Select [E10A PCI Card Driver] to use the PCI card emulator. For details, refer to table 2.4 in section 2.2.1 of the additional document.
 - [Interface] combo box: Select [PC Card] to use the PCMCIA card emulator. [PCI] is displayed to use the PCI card emulator. (If the driver is not installed, the [PC Card] or [PCI] is not displayed.)
- Click the [Close] button.

10. The dialog box is displayed as shown in figure 3.12.



Figure 3.12 Dialog Box of the RESET Signal Input Request Message

11. Power on the user system.

12. Input the reset signal from the user system, and click the [OK] button.

13. When using the MCU with flash memory, the dialog box shown in figure 3.13 is opened. Input the system clock value.

The system clock is not the input clock; it shows the CPU's operating frequency after the system clock has been multiplied.

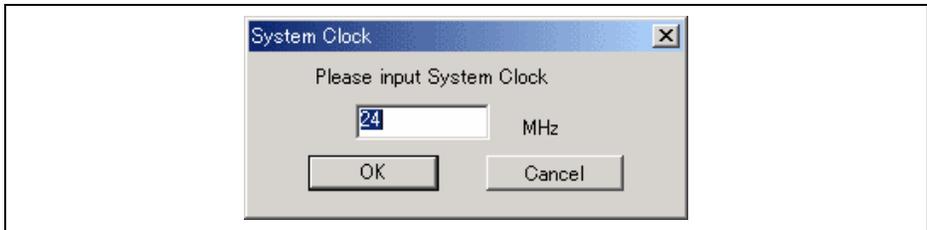


Figure 3.13 [System Clock] Dialog Box

14. After the following dialog box is displayed, input the ID code as a security code for the flash memory. However, H'FFFFFFFF is disabled as the ID code.
Input this ID code when [E10A Emulator] is selected and the [New ID code] check box is unselected on activating the emulator. If the ID code is not matched or the [New ID code] check box is selected, the flash memory contents are erased.

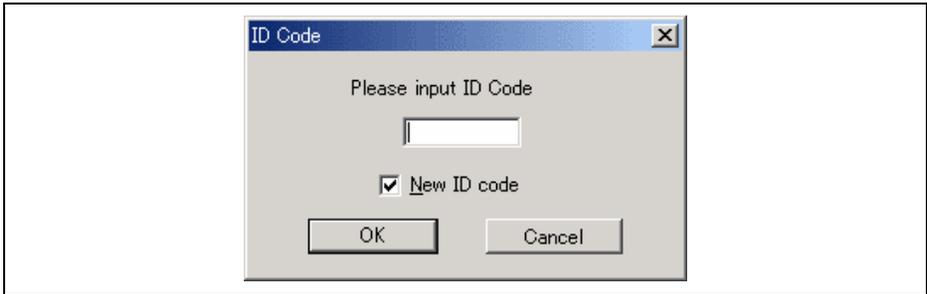


Figure 3.14 [ID Code] Dialog Box

15. When "Connected" is displayed in the [Output] window of the HEW, the emulator initiation is completed.

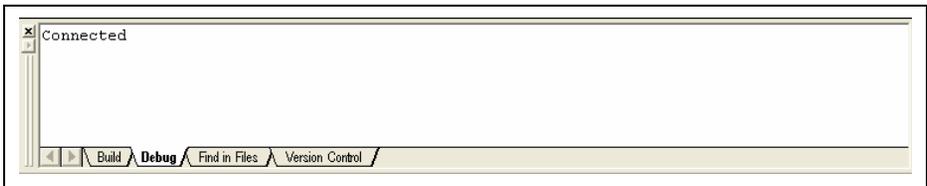


Figure 3.15 [Output] Window

Note: When the user program has already been downloaded to the flash memory, source-level debugging cannot be executed because there is no debugging information on the user program after the emulator has been activated. Be sure to load the debugging information file. For details, refer to section 4.3.1, Setting at Emulator Activation, in the Debugger Part.

Notes: 1. If the emulator is not initiated, the following dialog boxes shown in figures 3.16 through 3.24 will be displayed.

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

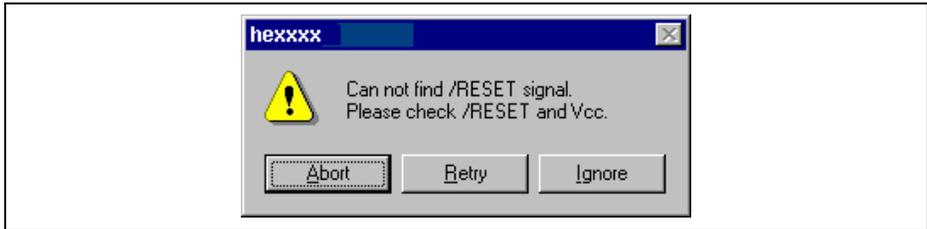


Figure 3.16 [Can not find /RESET signal] Dialog Box

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

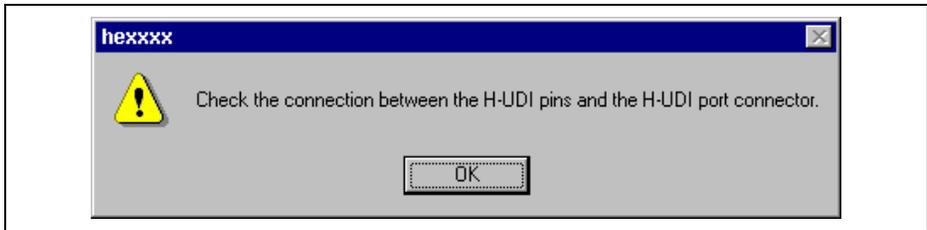


Figure 3.17 [Check the connection] Dialog Box

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



Figure 3.18 [COMMUNICATION TIMEOUT ERROR] Dialog Box



Figure 3.19 [INVALID ASERAM FIRMWARE!] Dialog Box

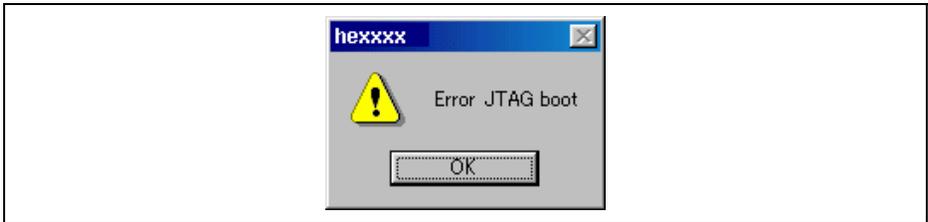


Figure 3.20 [Error JTAG boot] Dialog Box

- (d) The following dialog box is displayed when the flash memory cannot be erased. Exchange the MCU since the flash memory has been reprogrammed more times than the limitation.



Figure 3.21 [Flash memory erase error!] Dialog Box

- (e) The following dialog box is displayed when the flash memory cannot be reprogrammed. An incorrect system clock value has been input or the flash memory has been reprogrammed more times than the limitation.



Figure 3.22 [Error sending Flash memory write program] Dialog Box

- (f) The following dialog box is displayed when an incorrect ID code has been input. For security, the flash memory is completely erased.



Figure 3.23 [ID code error!] Dialog Box

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



Figure 3.24 [Boot Failed!] Dialog Box

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

3.6 Uninstalling the Emulator's Software

Follow this procedure to remove the installed emulator's software from the user's host computer. As the installed product is known by the HEW, uninstall the product on the HEW screen.

It is also possible to uninstall the emulator's software by using [Add/Remove Programs] in the control panel. In this case, however, note that all the tools (including the compiler) in the HEW will be removed.

1. Activate the HEW.
2. Click the [Administration...] button in the [Welcome!] dialog box.

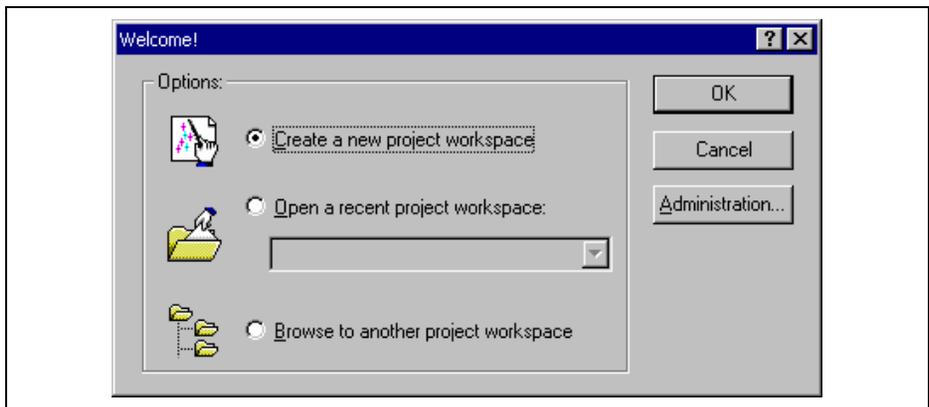


Figure 3.25 [Welcome!] Dialog Box

3. The [Tools Administration] dialog box is opened.

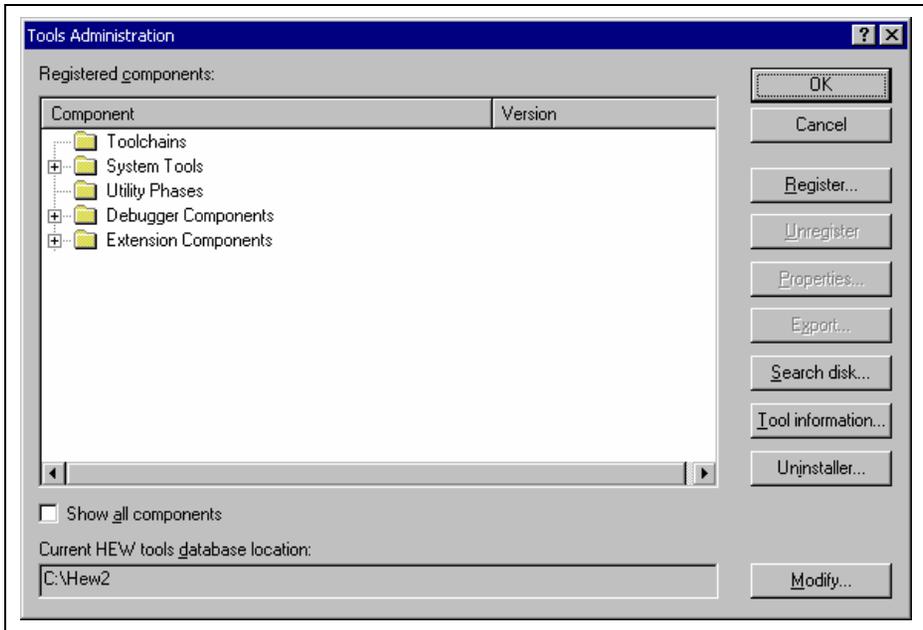


Figure 3.26 [Tools Administration] Dialog Box

- Click the [+] mark at the left of [Debugger Components] in the [Registered components] list box to list the installed components. Then, highlight the product name to be uninstalled.

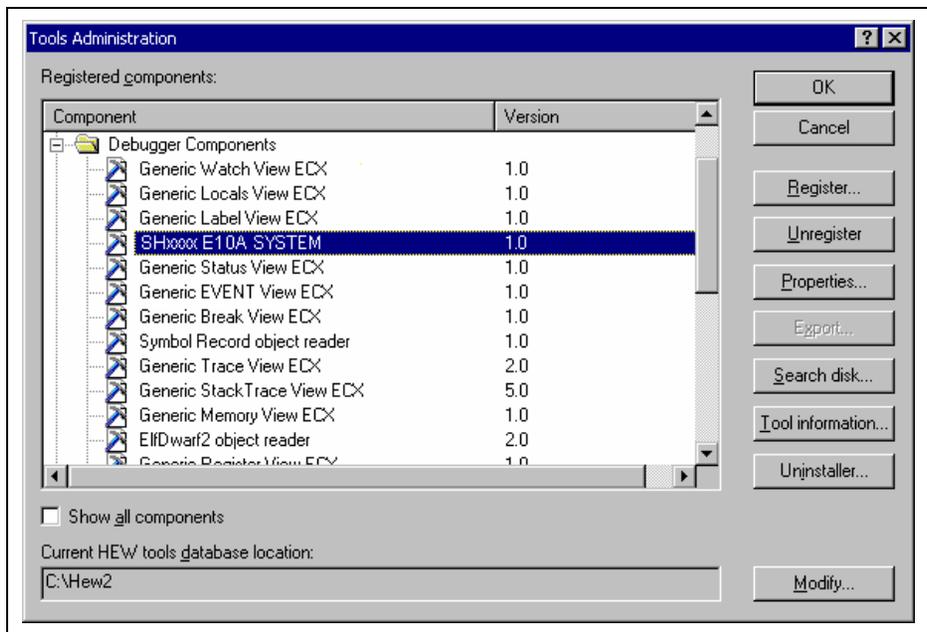


Figure 3.27 Highlighting the Product to be Uninstalled

- Click the [Unregister] button. After the following message box is displayed, click the [Yes] button.



Figure 3.28 [Unregistering this tool] Message Box

This is the end of canceling the HEW registration. Then, remove the file for the emulator from the host computer.

6. Click the [Uninstaller...] button in the [Tools Administration] dialog box to open the [Uninstall HEW Tool] dialog box.

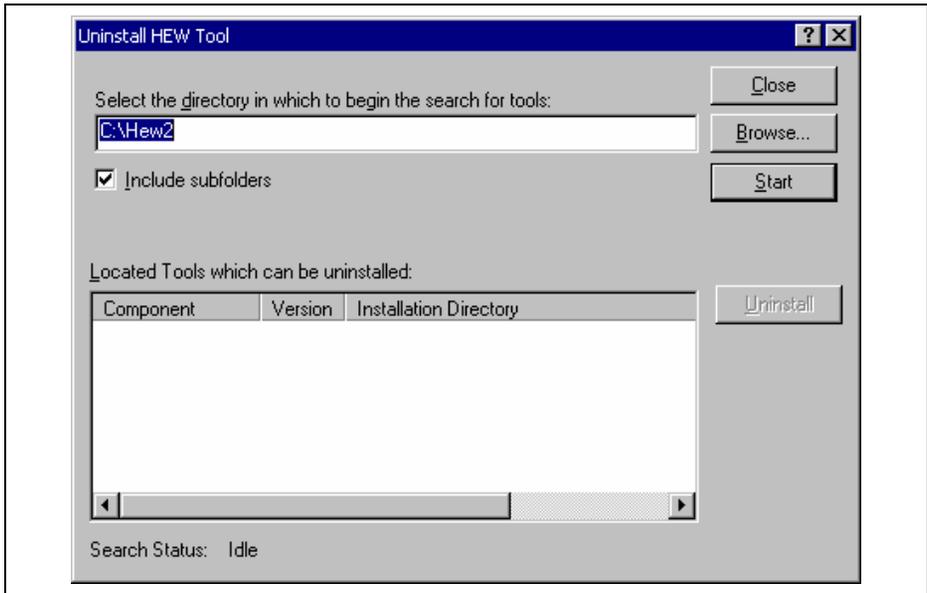


Figure 3.29 [Uninstall HEW Tool] Dialog Box

7. Click the [Start] button to list the installed components.

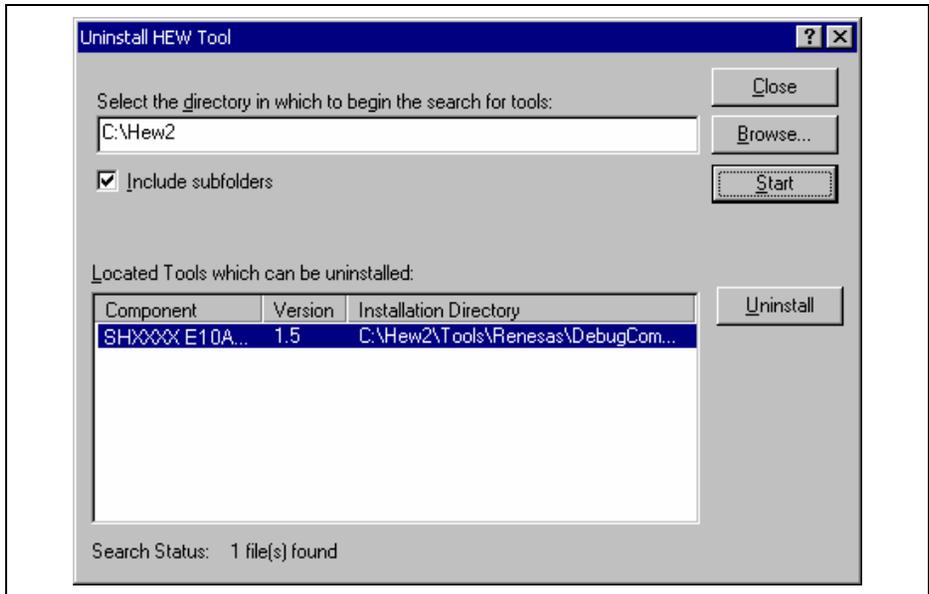


Figure 3.30 Highlighting the Product to be Uninstalled

Highlight the product name to be uninstalled and click the [Uninstall] button. This is the end of uninstallation.

CAUTION

A shared file may be detected while the program is being removed. If another product may be using the shared file, do not remove the file. When Microsoft® Windows NT®4.0 operating system is used, the removal of the registry information on the driver may be asked. If other product may use the target driver, do not remove the registry information. If another product does not start up after the removal process, re-install that product.

Section 4 Preparations for Debugging

4.1 Workspaces, Projects, and Files

Just as a word processor allows you to create and modify documents, HEW 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. Figure 4.1 illustrates this graphically.

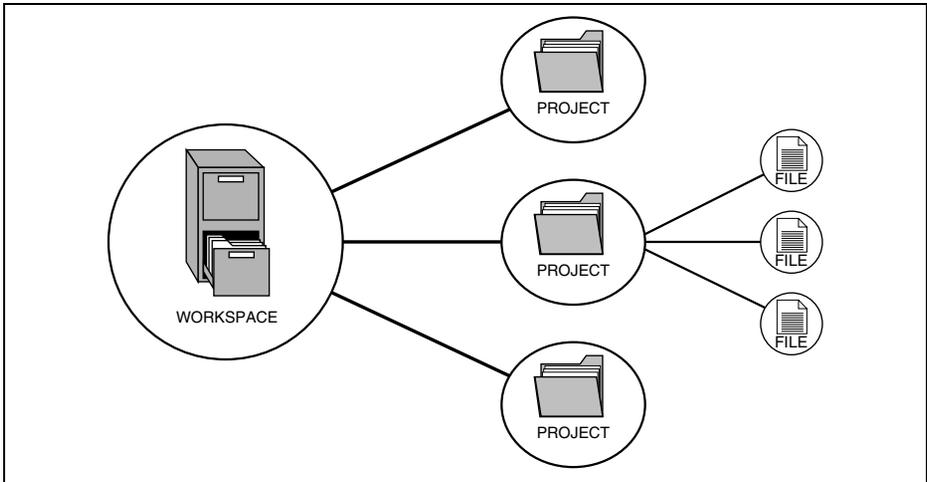


Figure 4.1 Workspaces, Projects, and Files

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 Method for Activating HEW

To activate the HEW, follow the procedure listed below.

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

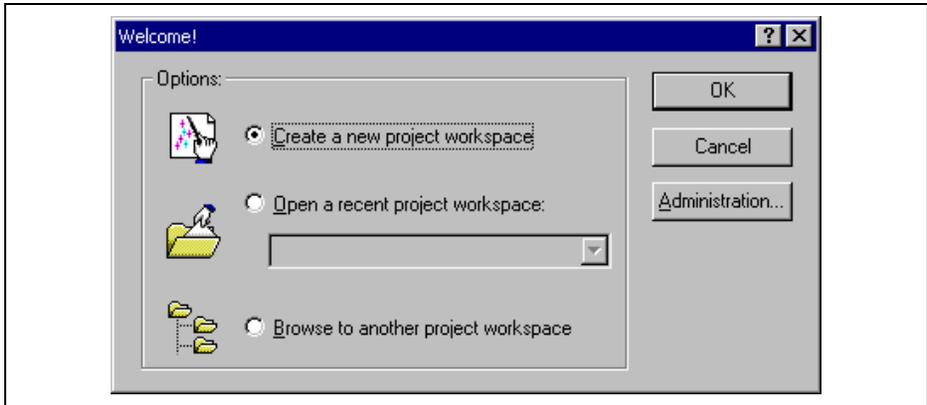


Figure 4.2 [Welcome!] Dialog Box

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

In this section, we describe the following three ways to start up the HEW:

- [Create a new project workspace] - a toolchain is not in use
- [Create a new project workspace] - a toolchain is in use
- [Browse to another project workspace]

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

4.2.1 Creating the New Workspace (Toolchain Not Used)

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

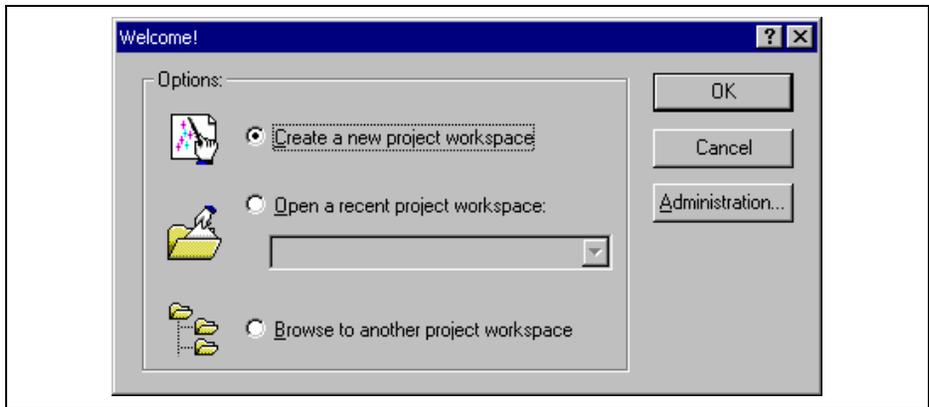


Figure 4.3 [Welcome!] Dialog Box

2. The Project Generator is started. This section omits the description on the setting for the toolchain.

If you have not purchased the toolchain, the following dialog box is displayed.

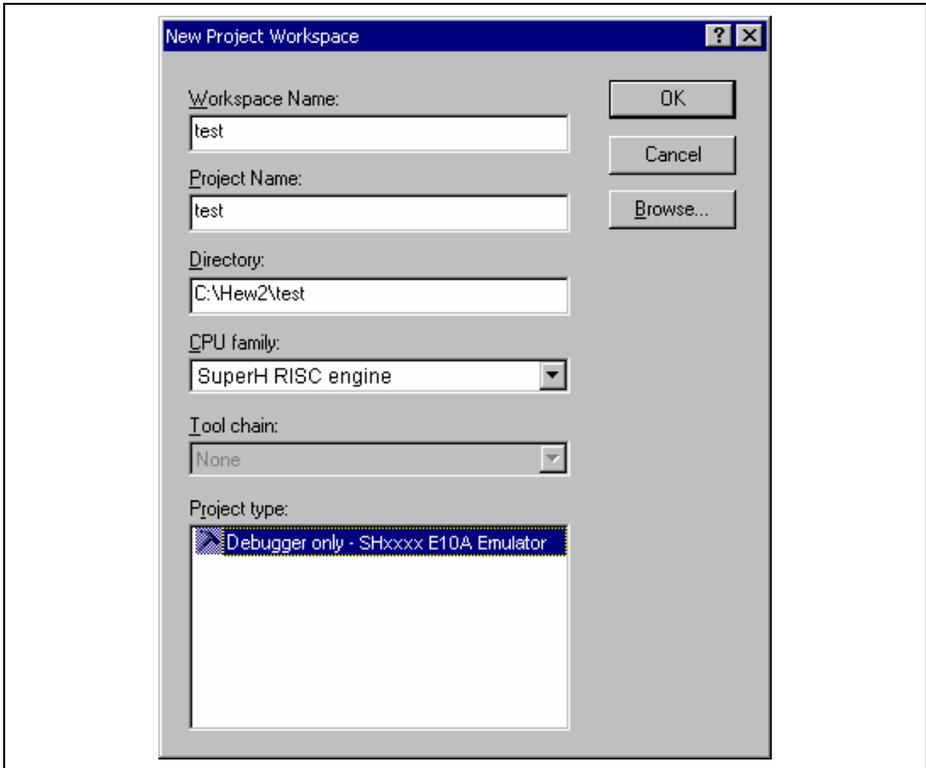


Figure 4.4 [New Project Workspace] Dialog Box

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

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

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

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

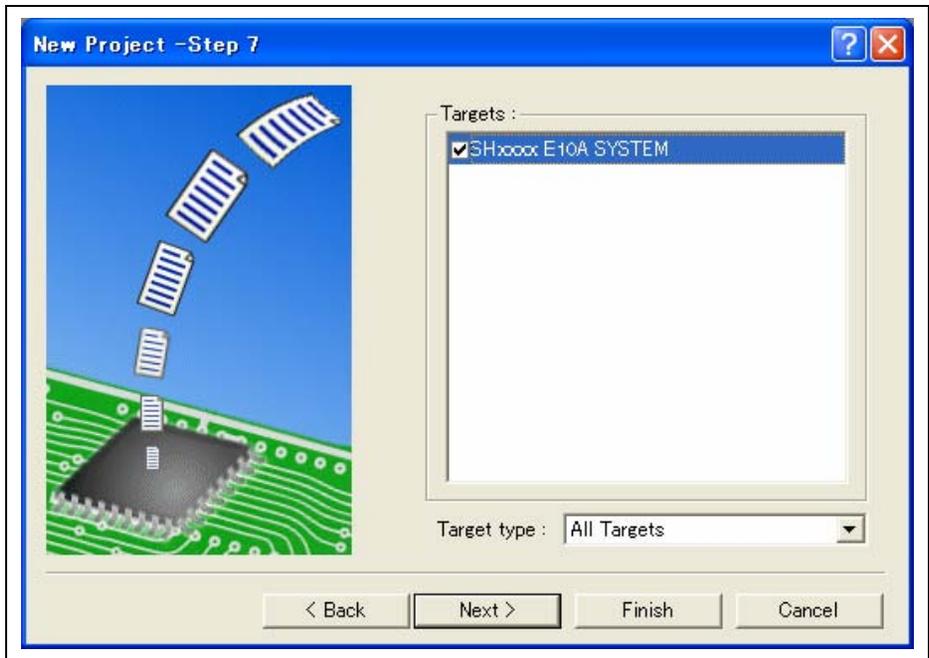


Figure 4.5 [New Project – Step 7] Dialog Box

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

4. Set the configuration file name. The configuration file saves the state of HEW except for the emulator.

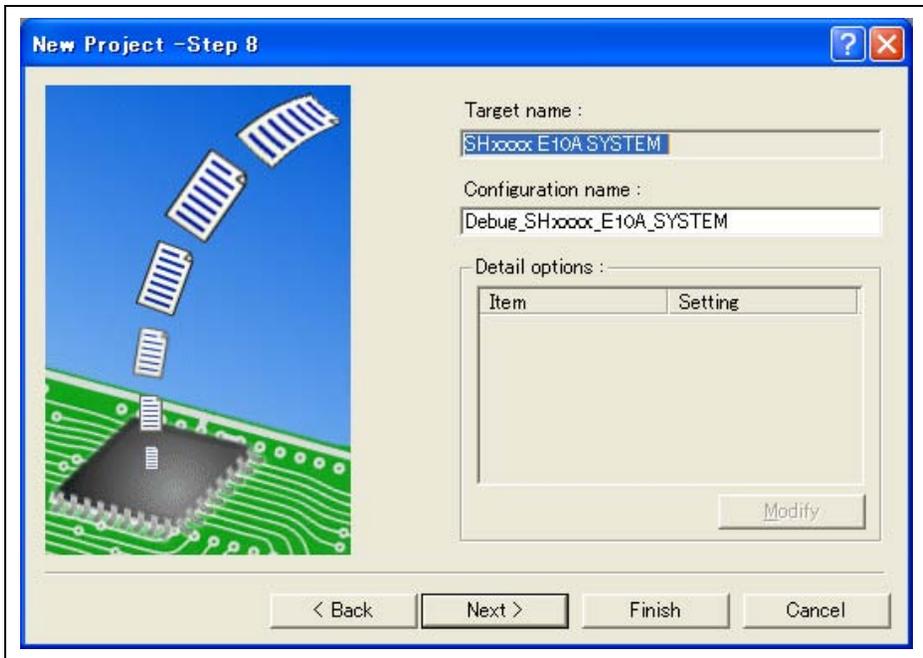


Figure 4.6 [New Project – Step 8] Dialog Box

This is the end of the emulator setting.

Click the [Finish] button to exit the Project Generator. The HEW is activated.

5. After the HEW has been activated, the emulator is automatically connected. For operation during connection, refer to section 3.5, System Check.

4.2.2 Creating the New Workspace (Toolchain Used)

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

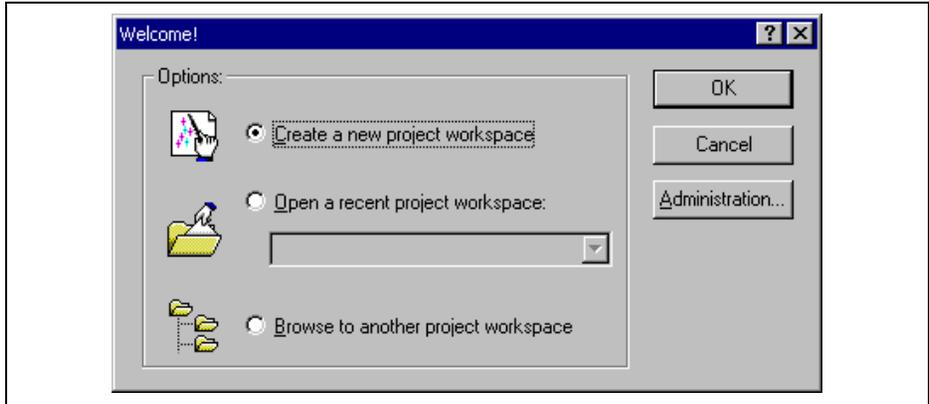


Figure 4.7 [Welcome!] Dialog Box

2. The Project Generator is started. For details, refer to the High-performance Embedded Workshop2 Tutorial, separately provided. This section omits the description on the setting for the toolchain.

If you have purchased the toolchain, the following dialog box is displayed.

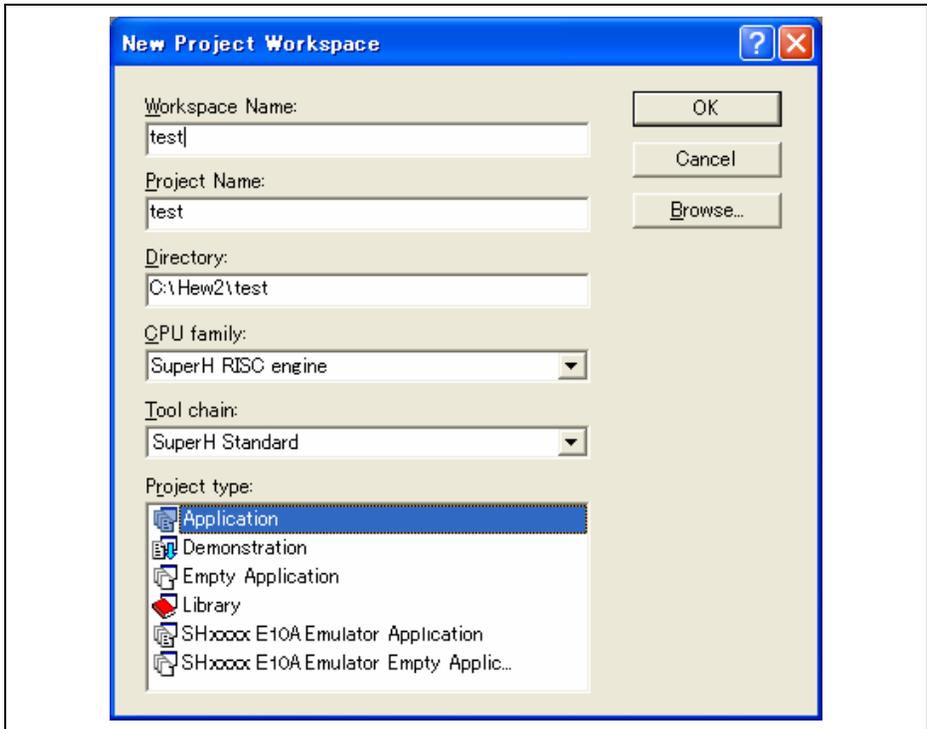


Figure 4.8 [New Project Workspace] Dialog Box

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

- Notes:
1. For the E10A emulator, the following project types are the same:
[Application] and [SHxxxx E10A Emulator Application]
[Assembly Application] and [SHxxxx E10A Emulator Assembly Application]
[Empty Application] and [SHxxxx E10A Emulator Empty Application]
 2. When [Demonstration] is selected in the E10A emulator, note the following:
The [Demonstration] is a program for the simulator. When using a program to be generated, delete the Printf statement.
 3. Make the required setting for the toolchain. When the setting has been completed, the following dialog box is displayed.

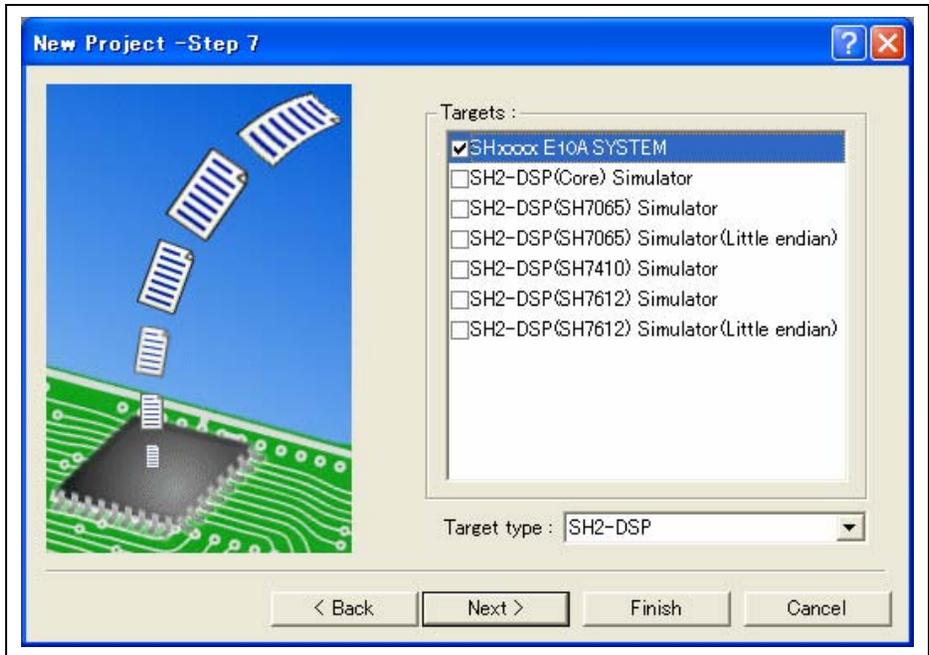


Figure 4.9 [New Project – Step 7] Dialog Box

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

- Set the configuration file name. The configuration file saves the state of HEW except for the emulator.

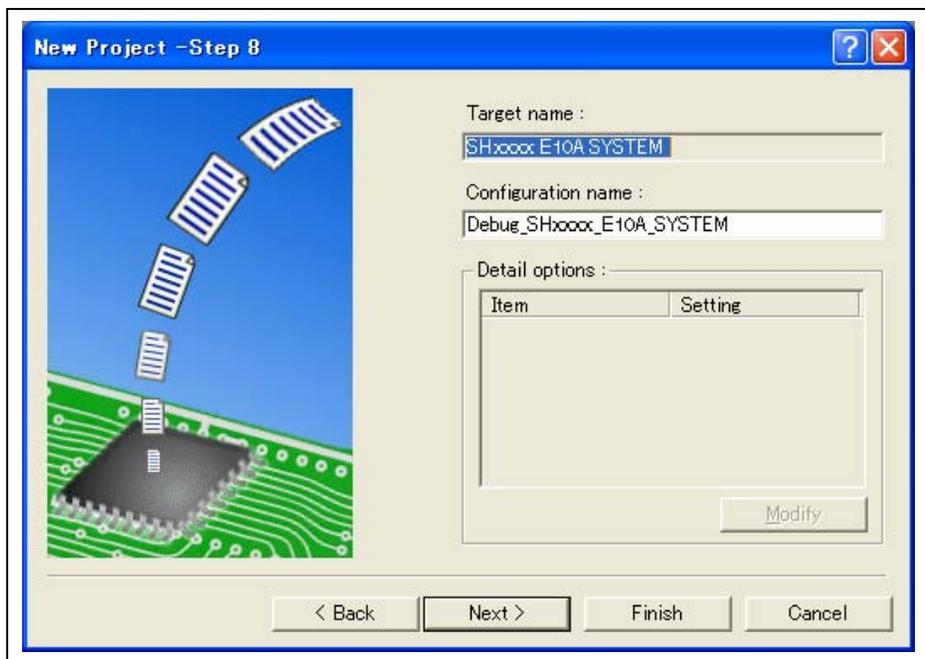


Figure 4.10 [New Project – Step 8] Dialog Box

This is the end of the emulator setting.

Exit the Project Generator according to the instructions on the screen. The HEW is activated.

- After the HEW has been activated, connect the emulator. However, it is not needed to connect the emulator immediately after the HEW has been activated.
To connect the emulator, use one of the methods (a) and (b) below. For operation during connection, refer to section 3.5, System Check.

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

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

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

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

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

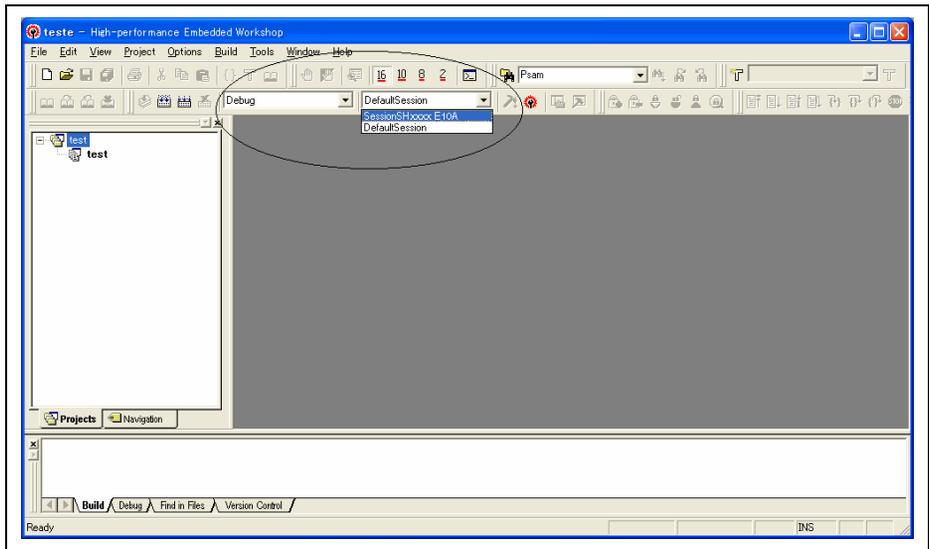


Figure 4.11 Selecting the Session File

In the list box that is circled in figure 4.11, select the session file name including the character string that has been set in the [Target name] text box in figure 4.10, [New Project – Step 8] dialog box. The setting for using the emulator has been registered in this session file.

After selected, the emulator is automatically connected.

4.2.3 Selecting an Existing Workspace

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

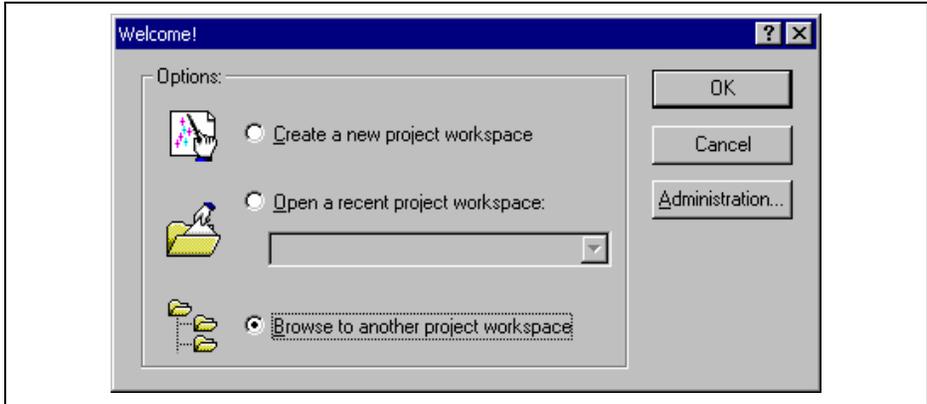


Figure 4.12 [Welcome!] Dialog Box

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

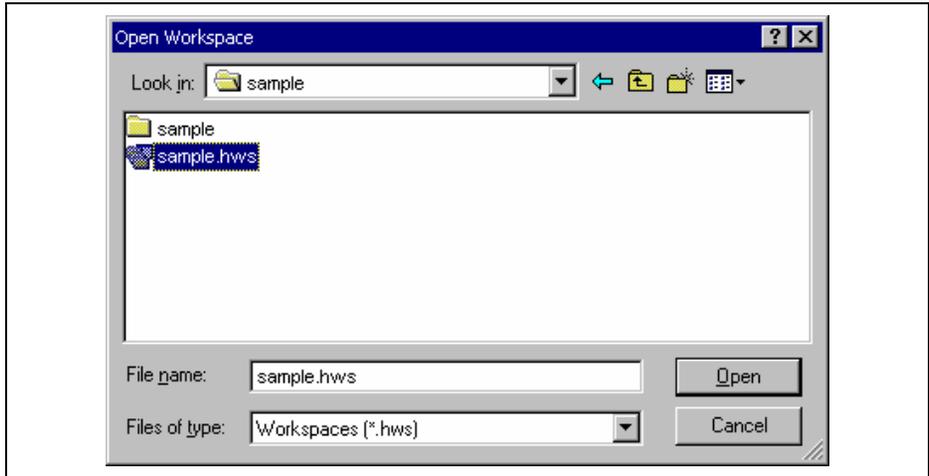


Figure 4.13 [Open Workspace] Dialog Box

3. This activates the HEW and recovers the state of the selected workspace at the time it was saved.

When the saved state information of the selected workspace includes connection to the emulator, the emulator will automatically be connected. To connect the emulator when the saved state information does not include connection to the emulator, refer to section 4.5, Connecting the Emulator.

4.3 Setting at Emulator Activation

4.3.1 Setting at Emulator Activation

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

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

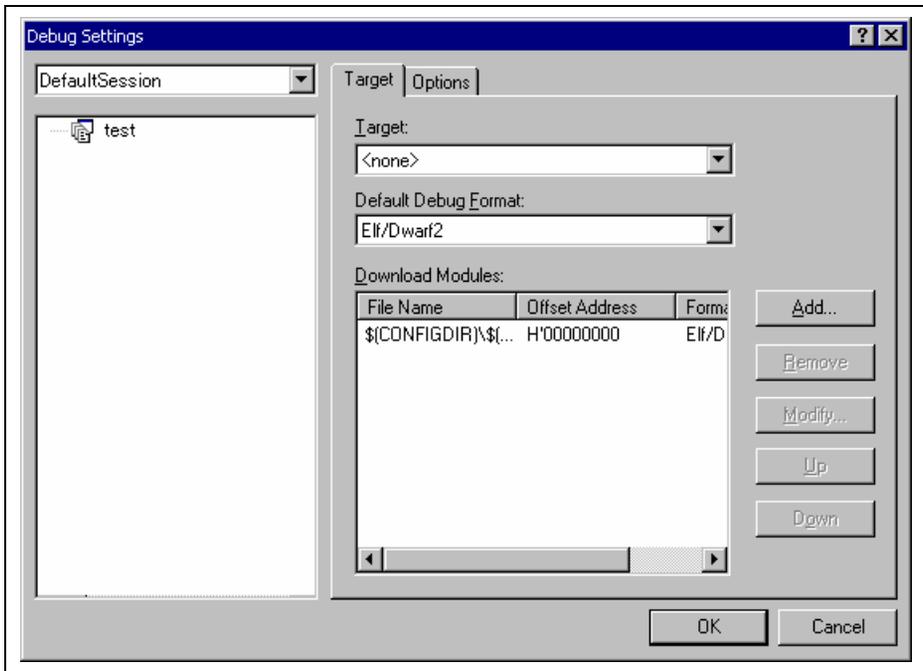


Figure 4.14 [Debug Settings] Dialog Box ([Target] Page)

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

Note: Here, no program has been downloaded. For downloading, refer to section 5.2, Downloading a Program, in the Debugger Part.

4. Click the [Options] tab.

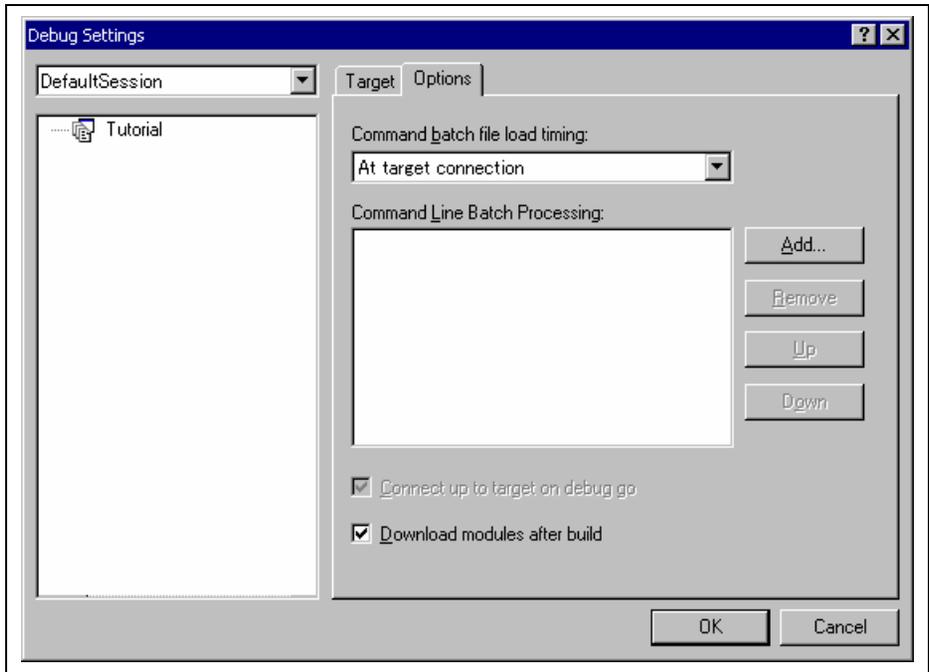


Figure 4.15 [Debug Settings] Dialog Box ([Options] Page)

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

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

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

4.3.2 Downloading a Program

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

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

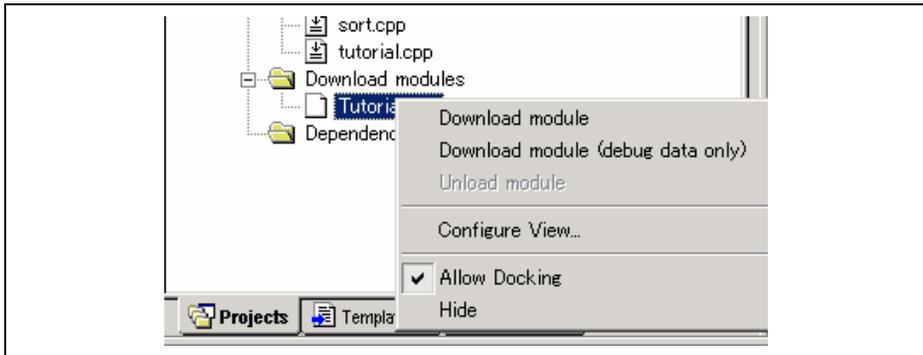


Figure 4.16 Download Menu of the [Workspace] Window ([Projects])

- Notes:
1. When load modules are downloaded, select [Debug] -> [Download] -> [All DownLoad Modules].
 2. The emulator downloads programs to the flash memory just before execution of the user program.

4.3.3 Setting the Writing Flash Memory Mode

The following describes the procedures when the emulator is used as the programming tool. The load module to be downloaded to the new workspace is registered and programmed.

- (a) Select the new project workspace.

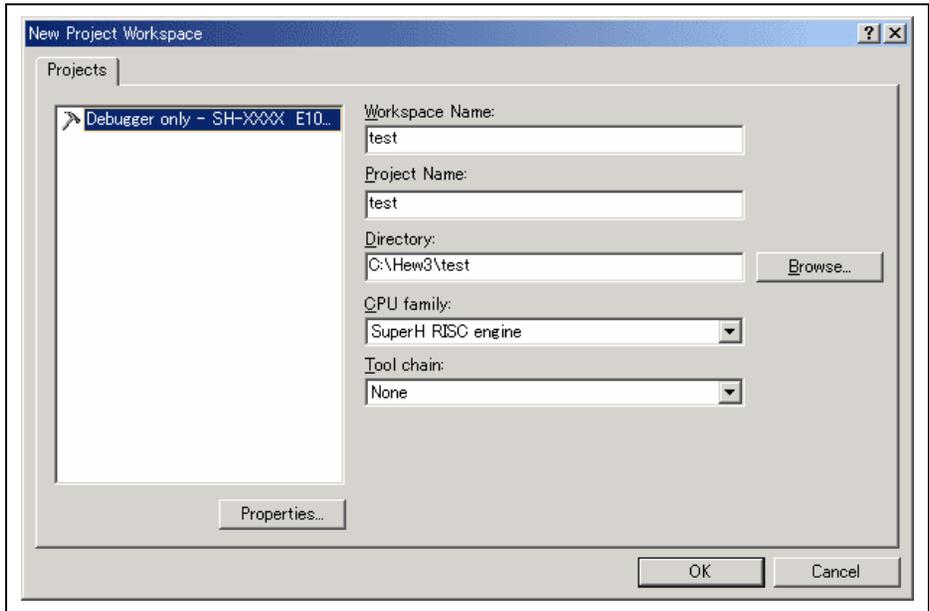


Figure 4.17 [New Project Workspace] Dialog Box

(b) Select the target MCU and click the [Next] button.

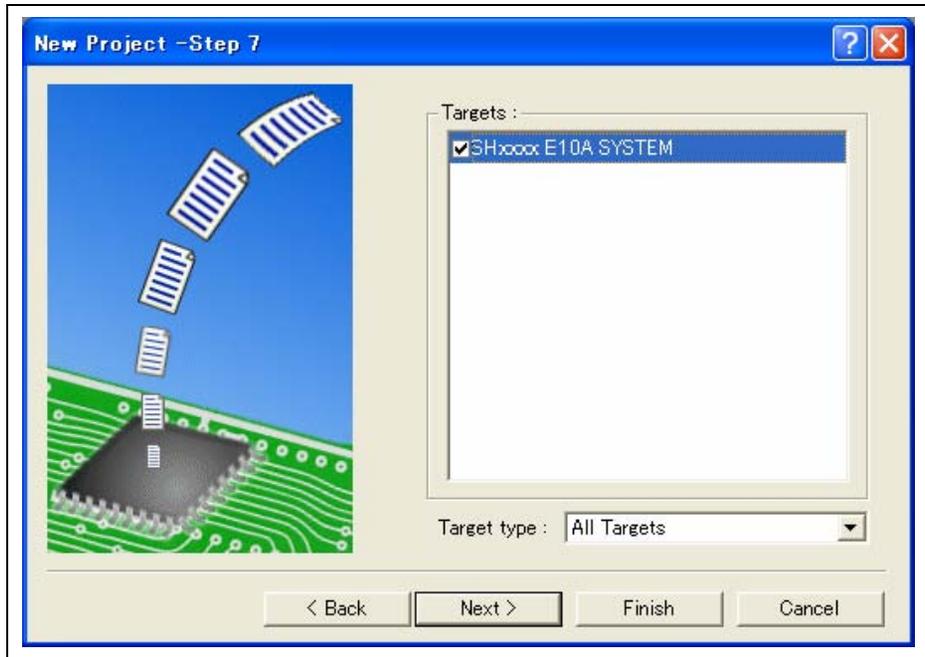


Figure 4.18 [New Project - Step 7] Dialog Box

(c) The [Select Emulator mode] dialog box is displayed.



Figure 4.19 [Select Emulator mode] Dialog Box

Select the [Writing Flash memory] mode.

(d) Turn on the target board and press the [OK] button.



Figure 4.20 Dialog Box of the RESET Signal Input Request Message

- (e) Input the system clock value. The system clock is not the input clock; it shows the CPU's operating frequency after the system clock has been multiplied.

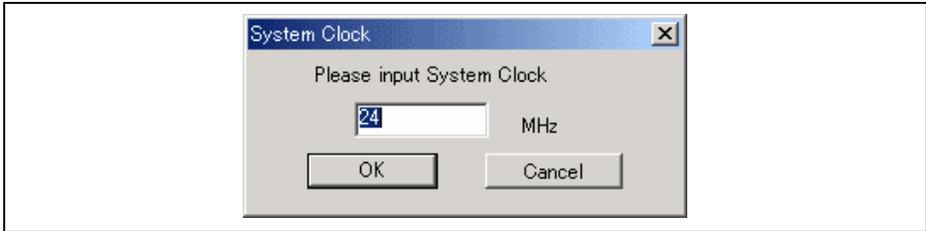


Figure 4.21 [System Clock] Dialog Box

- (f) Select [Debug Setting] from the [Option] menu.

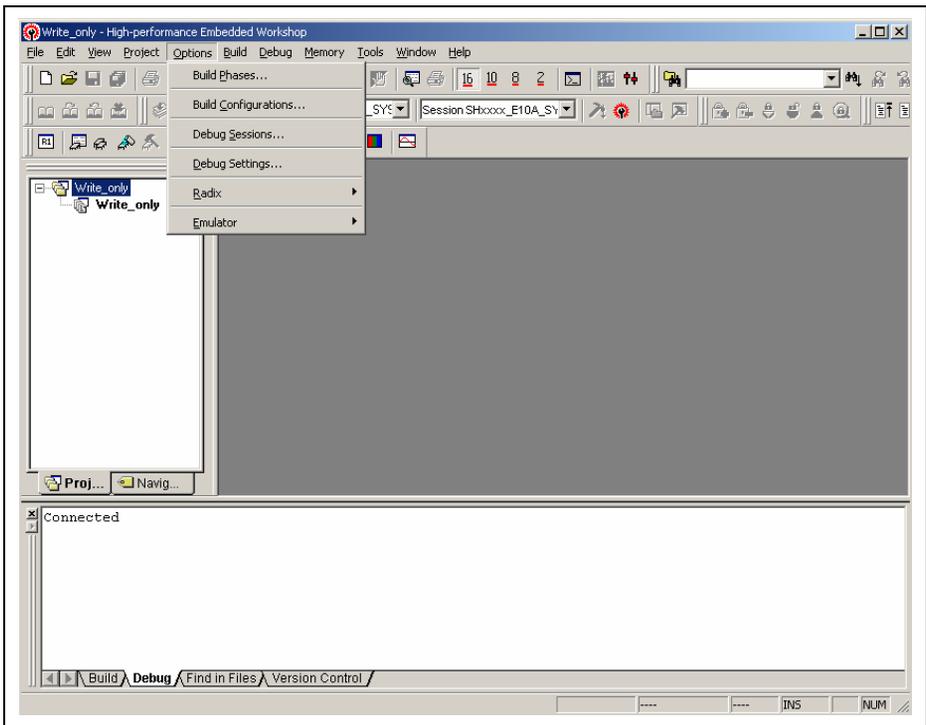


Figure 4.22 HEW Window

- (g) Select the target MCU and then the download module with the [Add...] button.

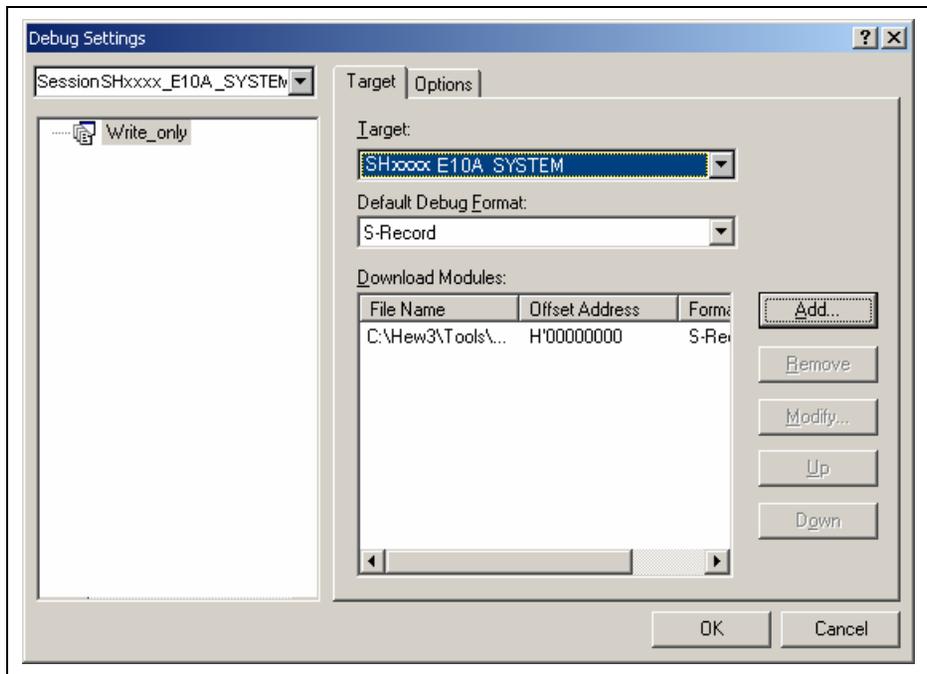


Figure 4.23 [Debug Setting] Dialog Box ([Target] Page)

(h) The download file is displayed on [Project Files].

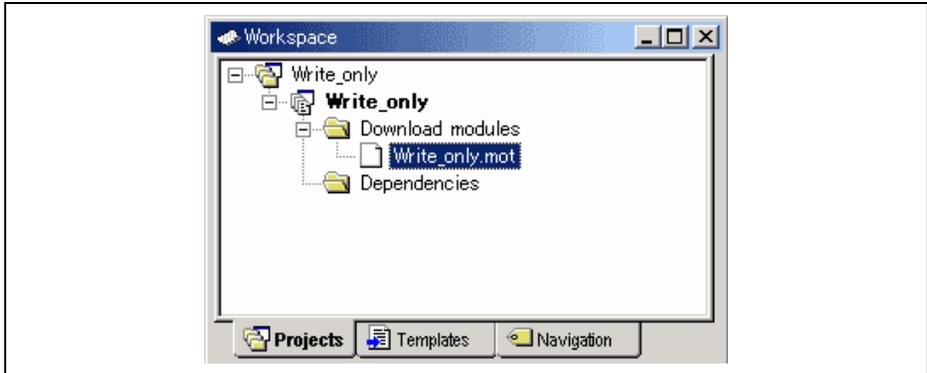


Figure 4.24 [Workspace] Window ([Project Files])

(i) Select and download the file with the right-hand mouse button.

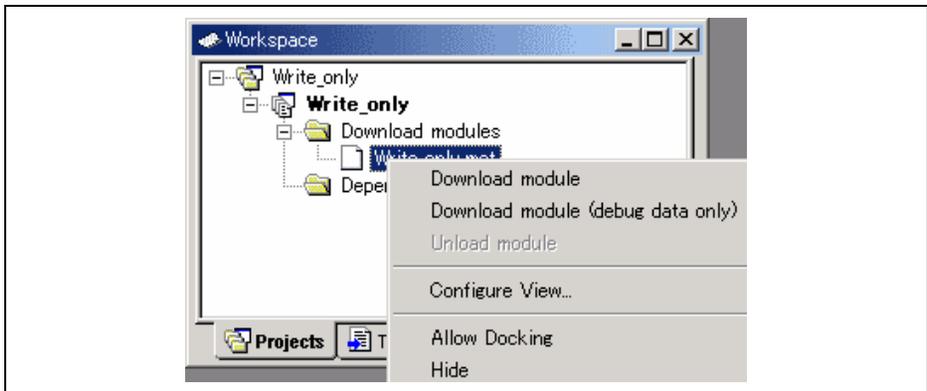


Figure 4.25 Download Menu of the [Workspace] Window ([Project Files])

- (j) The dialog box for sum checking is displayed and programming is completed. Sum data is a value that data in the internal ROM area has been added by byte. If no user program exists, the value is calculated by H'FF.

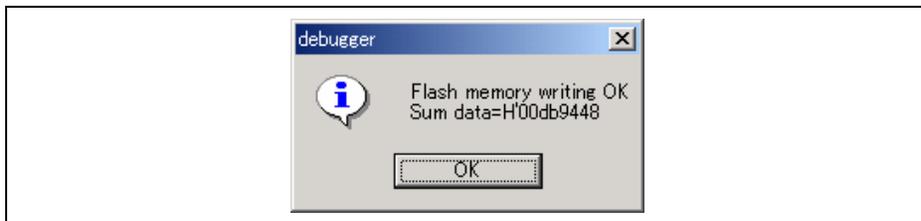


Figure 4.26 Message for Completion of Flash Memory Programming

- (k) When the following dialog box is displayed, close and restart or exit the workspace.



Figure 4.27 Message for Restarting or Exiting Writing Flash Memory Mode

4.4 Debugger Sessions

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

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

Each session's data should be stored in a separate file in the HEW project. Debugger sessions are described in detail below.

4.4.1 Selecting a Session

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

- From the toolbar
Select a session from the drop-down list box (figure 4.28) in the toolbar.



Figure 4.28 Toolbar Selection

- From the dialog box
 1. Select [Options -> Debug Sessions...]. This will open the [Debug Sessions] dialog box (figure 4.29).

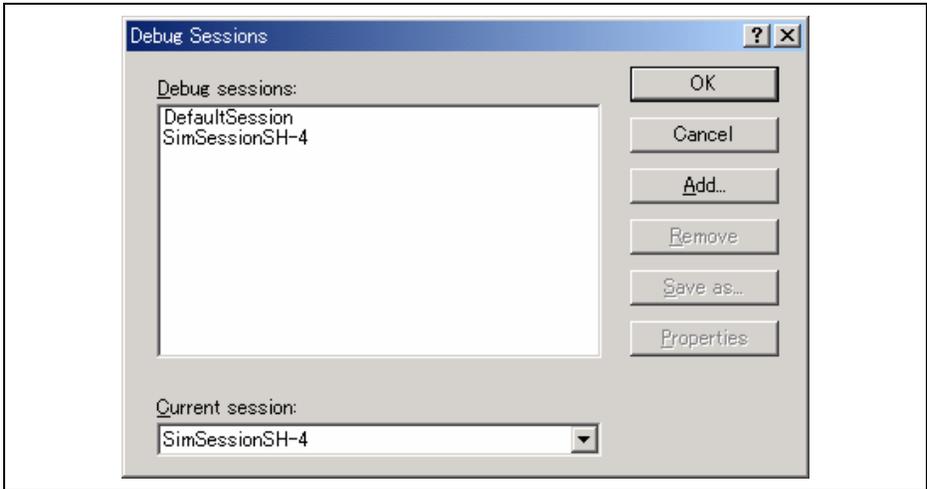


Figure 4.29 [Debug Sessions] Dialog Box

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

4.4.2 Adding and Deleting Sessions

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

- To add a new empty session
 1. Select [Options -> Debug Sessions...] to display the [Debug Sessions] dialog box (figure 4.29).
 2. Click the [Add...] button to display the [Add new session] dialog box (figure 4.30).
 3. Check the [Add new session] radio button.
 4. Enter a name for the session.
 5. Click the [OK] button to close the [Debug Sessions] dialog box.
 6. This creates a file with the same name as the entered session name. If the file name already exists, an error is displayed.

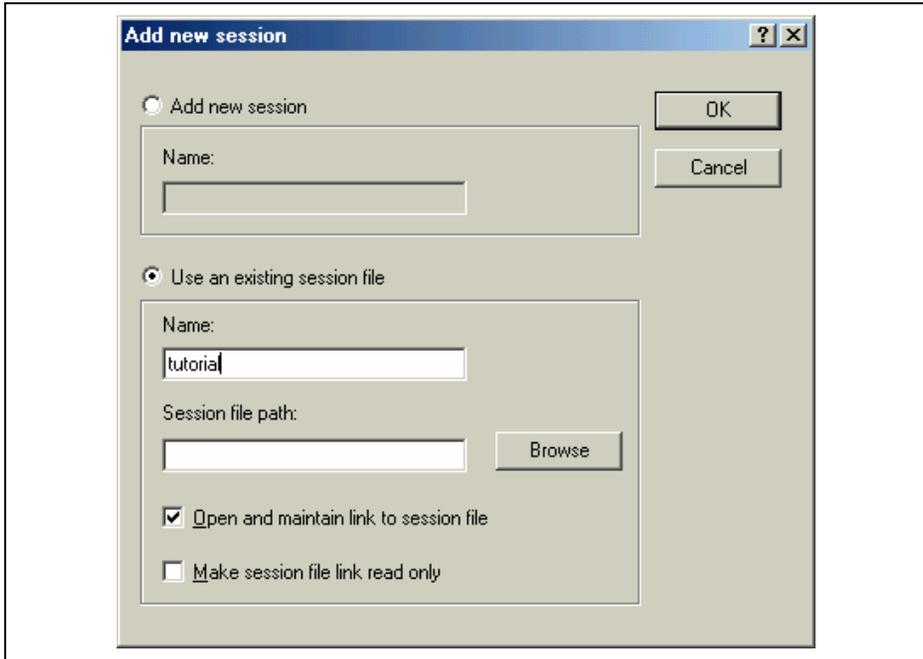


Figure 4.30 [Add new session] Dialog Box

- To import an existing session into a new session file
 1. Select [Options -> Debug Sessions...] to display the [Debug Sessions] dialog box (figure 4.29).
 2. Click the [Add...] button to display the [Add new session] dialog box (figure 4.30).
 3. Check the [Use an existing session file] radio button.
 4. Enter a name for the session.
 5. Enter the name of an existing session file that you would like to import into the existing project or click the [Browse] button to select the file location.

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

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

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

- To remove a session
 1. Select [Options -> Debug Sessions...] to display the [Debug Sessions] dialog box (figure 4.29).
 2. Select the session you would like to remove.
 3. Click the [Remove] button.

Note that the current session cannot be removed.
 4. Click the [OK] button to close the [Debug Sessions] dialog box.
- To view the session properties
 1. Select [Options -> Debug Sessions...] to display the [Debug Sessions] dialog box (figure 4.29).
 2. Select the session you would like to view the properties for.
 3. Click the [Properties] button to display the [Session Properties] dialog box (figure 4.31).

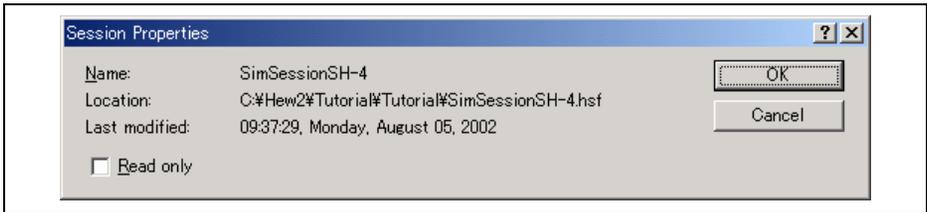


Figure 4.31 [Session Properties] Dialog Box

- To make a session read-only
 1. Select [Options -> Debug Sessions...] to display the [Debug Sessions] dialog box (figure 4.29).
 2. Select the session you would like to make read-only.
 3. Click the [Properties] button to display the [Session Properties] dialog box (figure 4.31).
 4. Check the [Read only] check box to make the link read-only. This is useful if you are sharing debugger-setting files and you do not want data to be modified accidentally.
 5. Click the [OK] button.
- To save a session with a different name
 1. Select [Options -> Debug Sessions...] to display the [Debug Sessions] dialog box (figure 4.29).
 2. Select the session you would like to save.
 3. Click the [Save as...] button to display the [Save Session] dialog box (figure 4.32).
 4. Specify the new file location.

5. If you want to export the session file to another location, leave the [Maintain link] check box unchecked. If you would like the HEW to use this location instead of the current session location, check the [Maintain link] check box.
6. Click the [Save] button.

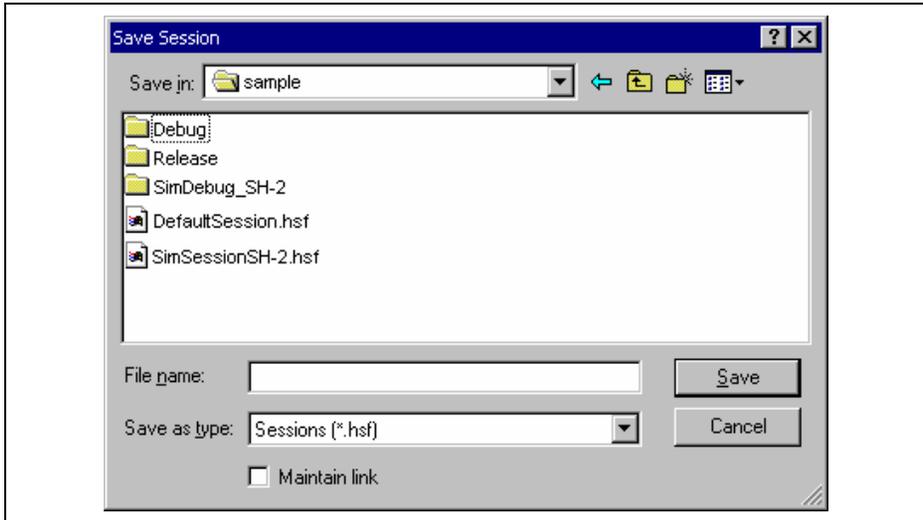


Figure 4.32 [Save Session] Dialog Box

4.4.3 Saving Session Information

- To save a session
Select [File -> Save Session].

4.5 Connecting the Emulator

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

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

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

When the dialog box is closed after setting the [Debug Settings] dialog box, the emulator will automatically be connected.

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

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

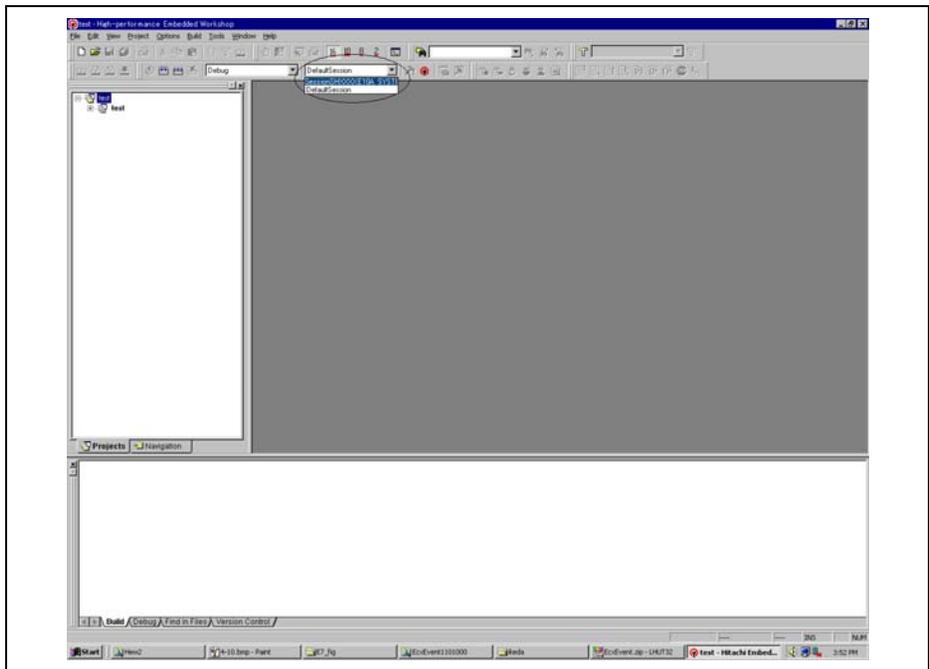


Figure 4.33 Selecting the Session File

In the list box that is circled in figure 4.33, select the session file name including the character string that has been set in the [Target name] text box in figure 4.10, [New Project – Step 8] dialog box. The setting for using the emulator has been registered in this session file.

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

4.6 Ending the Emulator

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

- Canceling the connection of the emulator being activated
- Exiting the HEW

(1) Canceling the connection of the emulator being activated

1. Select [Debug settings] from the [Options] menu to open the [Debug Settings] dialog box ([target] page) (see figure 4.14).
2. Select <None> or another product in the [Target] drop-down list box. When another product is selected, the connection with that product is started after canceling the connection of the emulator being activated.

(2) Exiting the HEW

1. Select [Exit] from the [File] menu.
2. A message box is displayed. If necessary, click the [Yes] button to save a session. After saving a session, the HEW exits. If not necessary, click the [No] button to exit the HEW.

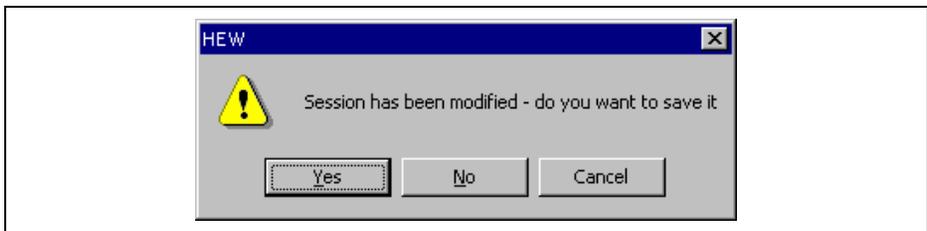


Figure 4.34 [Session has been modified] Message Box

Section 5 Debugging

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

5.1 Setting the Environment for Emulation

5.1.1 Opening the [Configuration] Dialog Box

Selecting [Options -> Emulator -> System...] or clicking the [Emulator System] toolbar button  opens the [Configuration] dialog box.

5.1.2 [General] Page

Sets the emulator operation conditions.

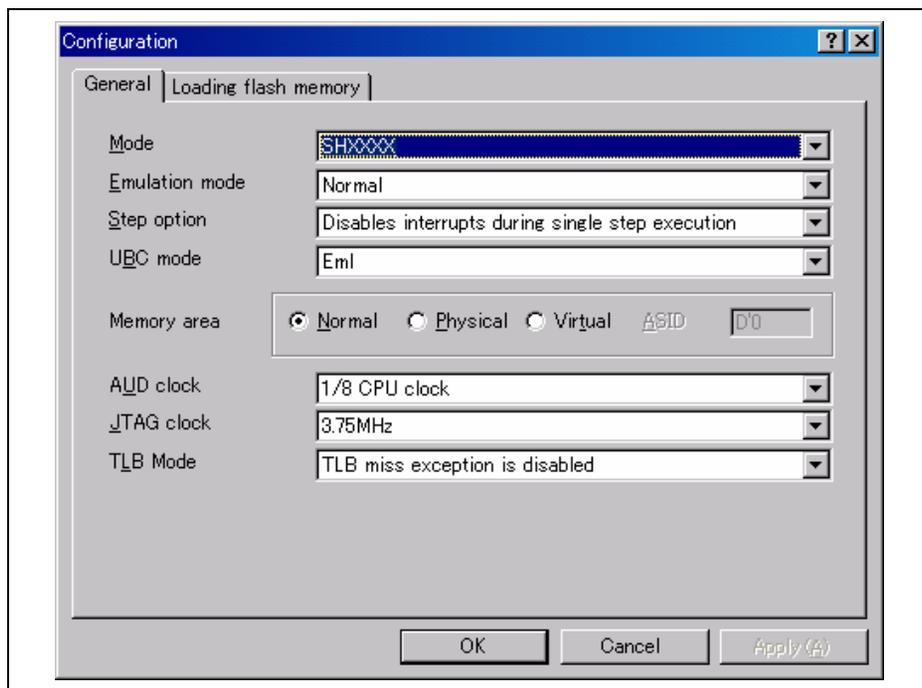


Figure 5.1 [Configuration] Dialog Box ([General] Page)

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

[Mode]	Displays the MCU name.
[Emulation mode]	Selects the emulation mode at user program execution. Select Normal to perform normal emulation. Select No break to disable PC breakpoint or break condition settings during emulation.
[Step option]	Sets the step interrupt option. Disable interrupts during single step execution: Disables interrupts ^{*1} during step execution. Enable interrupts during single step execution: Enables interrupts during step execution.
[UBC mode]	Sets the UBC mode. EML: The UBC is used as a Break Condition by the emulator. User: Releases the UBC to the users. ^{*2}
[AUD clock]	A clock used in acquiring AUD traces. If its frequency is set too low, complete data may not be acquired during realtime tracing. Set the frequency not to exceed the upper limit for the MCU's AUD clock. The AUD clock is only needed for using emulators that have an AUD trace function. For the upper limit for the AUD clock, refer to section 2.2.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK), in the Specific Guide for the SHxxxx E10A Emulator.
[JTAG clock]	A communication clock used except for acquiring AUD trace. If its frequency is set too low, the speed of downloading will be lowered. Set the frequency not to exceed the upper limit for the MCU's guaranteed TCK range. For the upper limit for TCK, refer to section 2.2.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK), in the Specific Guide for the SHxxxx E10A Emulator.

- [Flash memory synchronization] Selects whether or not the contents of the flash memory are acquired by the emulator when the user program is stopped or the position where the PC break is set is put back as the original code.
- When the flash memory is not reprogrammed by the user program, its contents need not be acquired by the emulator.
- If there is no problem with the state that the program in the flash memory has been replaced as the PC break code, the position where the PC break is set needs not be put back as the original code.
- Disable: Read or program is not performed on the flash memory except when the emulator is activated, the flash memory area is modified, and the settings of the PC break to the flash memory area are changed.
- PC to flash memory: When the user program is stopped, the specified PC break code is replaced as the original instruction. Select this option if there is a problem with the state that the program in the flash memory has been replaced as the PC break code.
- Flash memory to PC: When the user program is stopped, the contents of the flash memory are read by the emulator. Select this option if the flash memory is reprogrammed by the user program.
- PC to flash memory, Flash memory to PC:
When the user program is stopped, the contents of the flash memory are read by the emulator and the specified PC break code is replaced as the original instruction. Select this option if the flash memory is reprogrammed by the user program and there is a problem with the state that the program in the flash memory has been replaced as the PC break code.
- [AUD Port]: Using this option specifies whether or not the port that is multiplexed with the AUD pin is used as AUD.
- AUD not used: The port is not used as the AUD pin.
- AUD used: The port is used as the AUD pin.

- Notes: 1. Includes interrupts in a break.
2. When 'User' is selected, some functions are unavailable. For details, refer to the online help.

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

5.1.3 Downloading to the Flash Memory

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

For details, refer to section 6.22, Download Function to the Flash Memory Area, in the Debugger Part.

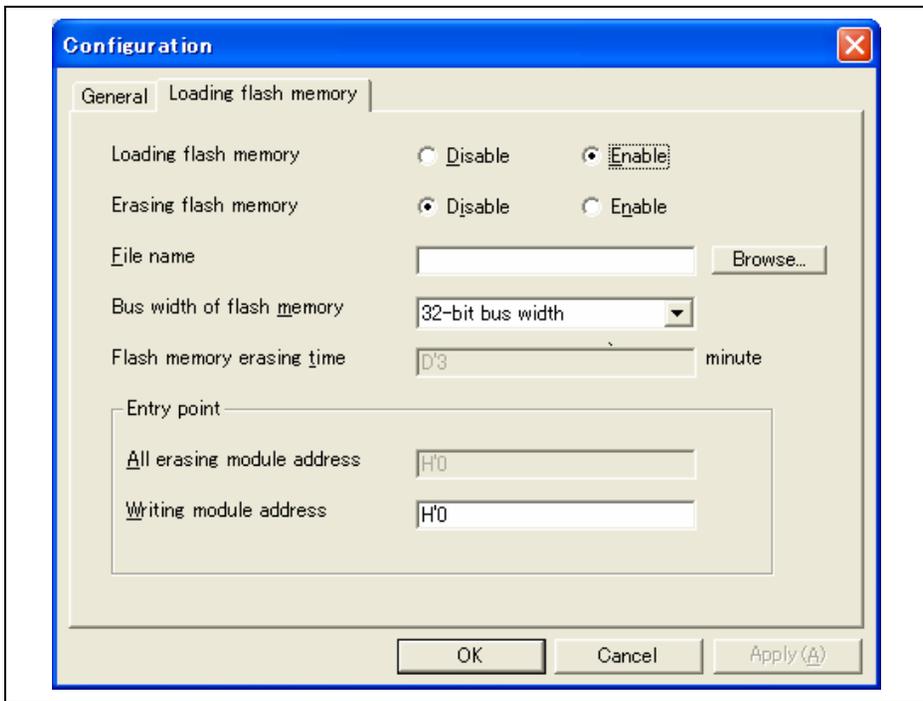


Figure 5.2 [Configuration] Dialog Box ([Loading flash memory] Page)

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

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

5.2 Downloading a Program

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

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

5.2.1 Downloading a Program

A load module to be debugged must be downloaded.

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

- Notes:**
1. Before downloading a program, it must be registered to the HEW as a load module. For registration, refer to section 4.3, Setting at Emulator Activation, in the Debugger Part.
 2. When a program is downloaded to the external RAM, the bus controller must be initially set in the area for downloading. Especially, check that the initialization of SDRAM or the setting of the bus width is appropriate for the target system.

5.2.2 Viewing the Source Code

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

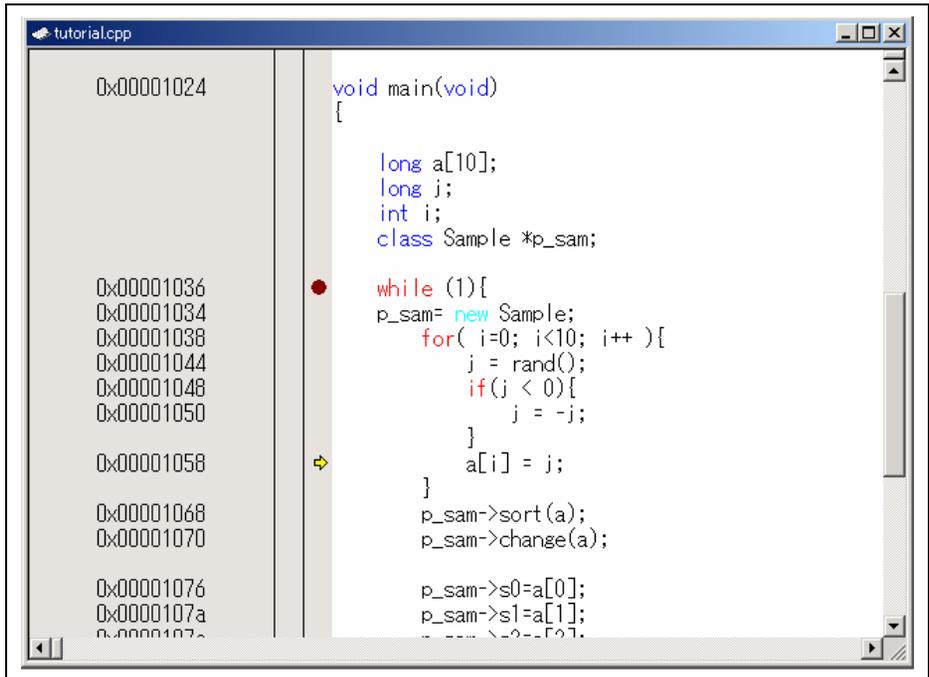


Figure 5.3 [Editor] Window

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

The first column (Source address column): Address information

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

The third column (Editor column): PC, bookmark, and breakpoint information

Source address column

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

Event column

The Event column displays the following item:

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

This is also set by using the popup menu.

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



Figure 5.4 Popup Menu

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

Editor column

Editor column displays the following items:

: A bookmark is set.

: A PC Break is set.

: PC location

 To switch off a column in all source files

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

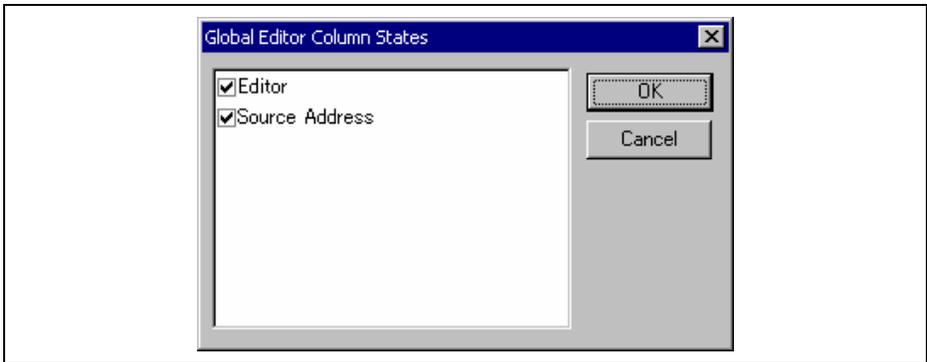


Figure 5.5 [Global Editor Column States] Dialog Box

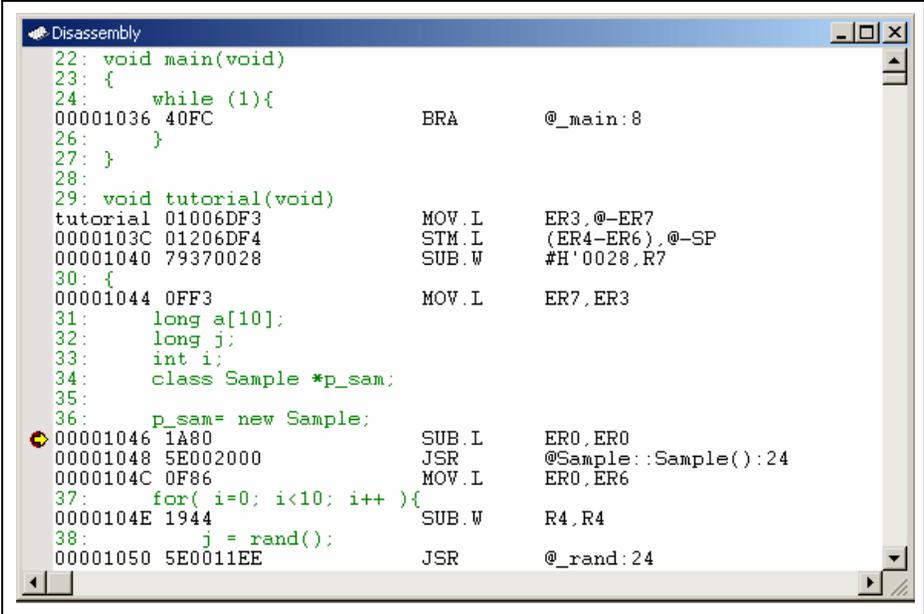
 To switch off a column in one source file

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

5.2.2 Viewing the Assembly-Language Code

Click the right-hand mouse button on the [Editor] window to open the popup menu and select [Go to Disassembly] to open the [Disassembly] window at the address that corresponds to the current source file.

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



```

Disassembly
22: void main(void)
23: {
24:     while (1){
00001036 40FC                BRA        @_main:8
26:     }
27: }
28:
29: void tutorial(void)
tutorial 01006DF3        MOV.L     ER3,@-ER7
0000103C 01206DF4        STM.L     (ER4-ER6),@-SP
00001040 79370028        SUB.W     #H'0028,R7
30: {
00001044 0FF3                MOV.L     ER7,ER3
31:     long a[10];
32:     long j;
33:     int i;
34:     class Sample *p_sam;
35:
36:     p_sam= new Sample;
00001046 1A80                SUB.L     ER0,ER0
00001048 5E002000        JSR       @Sample::Sample():24
0000104C 0F86                MOV.L     ER0,ER6
37:     for( i=0; i<10; i++ ){
0000104E 1944                SUB.W     R4,R4
38:         j = rand();
00001050 5E0011EE        JSR       @_rand:24

```

Figure 5.6 [Disassembly] Window

5.2.3 Modifying the Assembly-Language Code

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

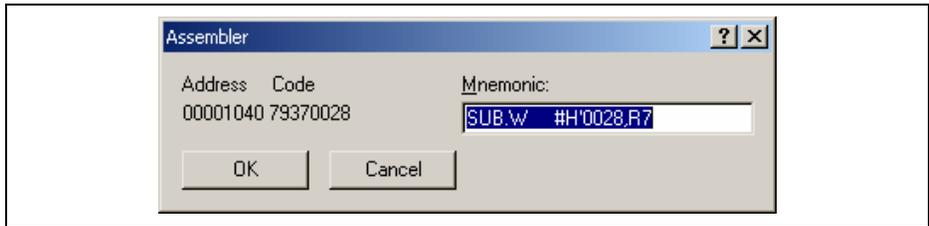


Figure 5.7 [Assembler] Dialog Box

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

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

5.2.4 Viewing a Specific Address

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

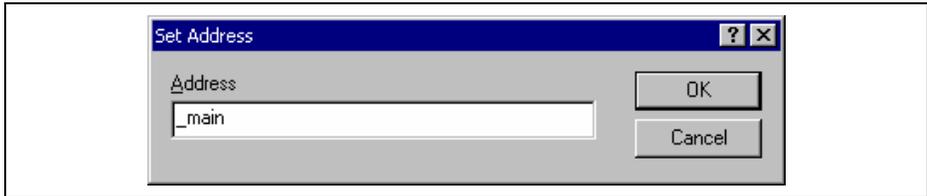


Figure 5.8 [Set Address] Dialog Box

Enter the address or label name in the edit box and either click on the [OK] button or press the [Enter] key. The [Disassembly] window will be updated to show the code at the new address. When an overloaded function or a class name is entered, the [Select Function] dialog box opens for you to select a function. For details, refer to section 5.11.3, Supporting Duplicate Labels.

5.2.5 Viewing the Current Program Counter Address

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

5.3 Debugging with the Command Line Interface

Use the [Command Line] window to enter text-based commands instead of window menus and commands.

5.3.1 Opening the [Command Line] Window

Choose [View -> Command Line] or click the [Command Line] toolbar button  to open the [Command Line] window.

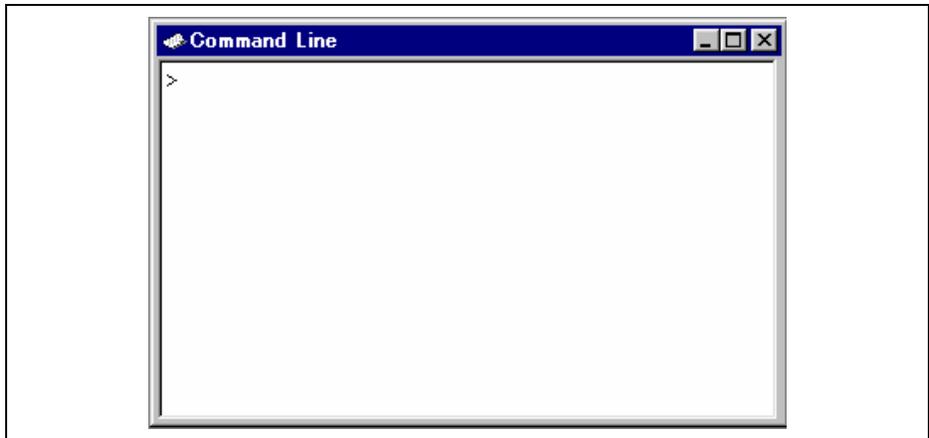


Figure 5.9 [Command Line] Window

This window allows the user to control the debugging platform by sending text-based commands. A series of predefined command lines can be called from a file and the output can be recorded in a file. The command can be executed by pressing the [Enter] key after the command is input at the prompt (>) on the last line. For information on the available commands, refer to appendix G, Command-Line Functions, and the on-line help.

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

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

5.3.2 Specifying a Command File

It is useful to use a command file when a series of predefined command lines need to be executed. Create a command file by a text editor and write necessary command lines. The default extension for command-file names is `.hdc`.

Choose [Set Batch File...] from the popup menu to open the [Set Batch File] dialog box, in which the name of a command file (*.hdc) can be specified. Clicking the [OK] button displays the specified command file name as the window title. Clicking the [Cancel] button closes the dialog box without changing the setting.

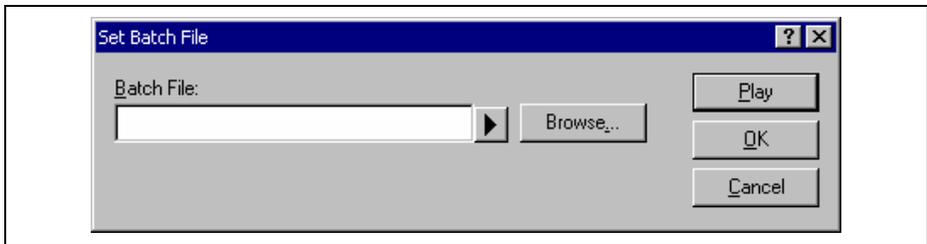


Figure 5.10 [Set Batch File] Dialog Box

5.3.3 Executing a Command File

Click the [Play] button in the [Set Batch File] dialog box or choose [Play] from the popup menu to execute the command file. The [Play] menu is displayed in gray while the file is running and can be used when the command file execution stops and control returns to the user.

5.3.4 Stopping Command Execution

Choose [Stop] from the popup menu to stop command execution. The [Stop] menu item becomes valid during command execution.

5.3.5 Specifying a Log File

Choose [Set Log File...] from the popup menu to open the [Open Log File] dialog box, in which a log file to store the command execution results can be specified.

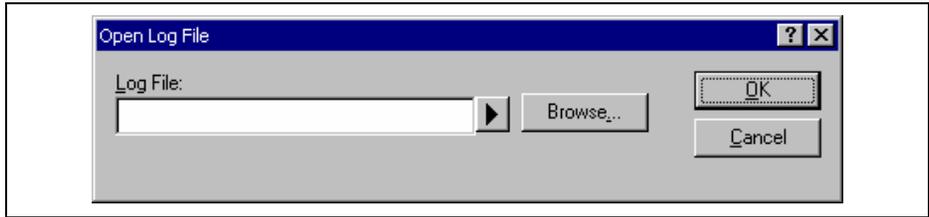


Figure 5.11 [Open Log File] Dialog Box

Enter the name of a log file (*.log). The logging option is automatically set and the name of the file is shown on the window title bar.

If you select the name of a previous log file, the system will ask whether you wish to append to or overwrite the existing file.

5.3.6 Starting or Stopping Logging

Choose [Logging] from the popup menu to toggle logging to file on and off. Note that the contents of the log file cannot be viewed until logging is completed, or temporarily disabled by clearing the check box. Re-enabling logging will append to the log file.

5.3.7 Entering a Full Path to the File

It is recommended that the full path to a file is specified as a file name in the [Command Line] window because the current directory can be moved. However, care must be taken to enter the correct full path to a file when it is entered from the keyboard. To solve this trouble, a full path can be easily specified by browsing through files.

Choose [Browse...] from the popup menu to open the [Browse] dialog box. Select a file and click [Open] to paste the full path to the selected file to the cursor location. This option can only be used when the cursor is located on the last line.

5.3.8 Pasting a Placeholder

Select a placeholder from the [Placeholder] submenu in the popup menu to paste the selected placeholder to the cursor location. This function is only available when the cursor is located on the last line.

5.4 Looking at Registers

If you are debugging at assembly-language level, then you will find it useful to see the contents of the CPU's general-purpose registers. You can do this by using the [Register] window.

5.4.1 Opening the [Register] Window

To open the [Register] window, choose [View->CPU->Registers] or click the [Register] toolbar button (). The [Register] window opens showing all of the CPU's general-purpose registers and the values, displayed in hexadecimal.

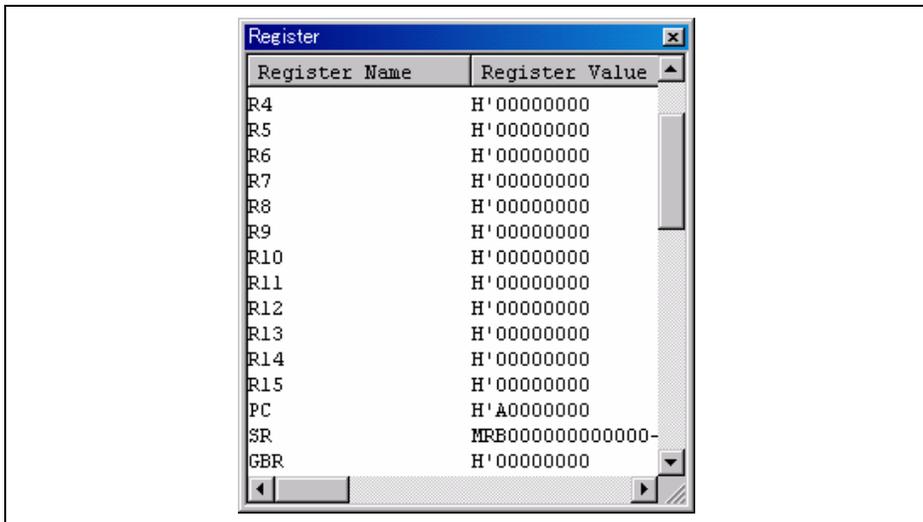


Figure 5.12 [Register] Window

5.4.2 Expanding a Bit Register

If a register is used as a set of flags at the bit level for the control of state, its one-character symbol rather than its state indicate each bit. Double-click on the register's name to display the [Edit Register] dialog box and switch each bit on or off. Checking the box for any bit specifies it as holding a 1, while removing the check specifies it as a 0.

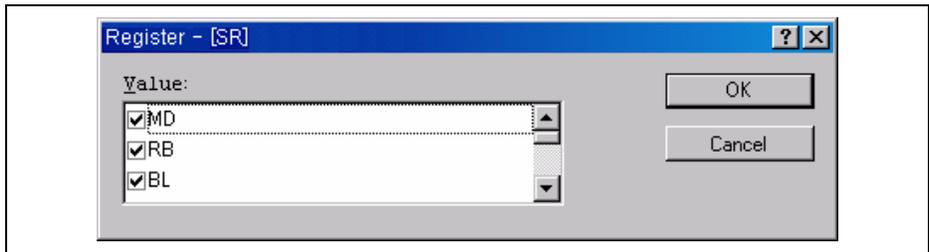


Figure 5.13 Expanding a Bit Register

5.4.3 Modifying Register Contents

To change a register's content, open the [Register] dialog box in one of the following methods:

- Double-click the register you want to change.
- Select the register you want to change, and choose [Edit...] from the popup menu.

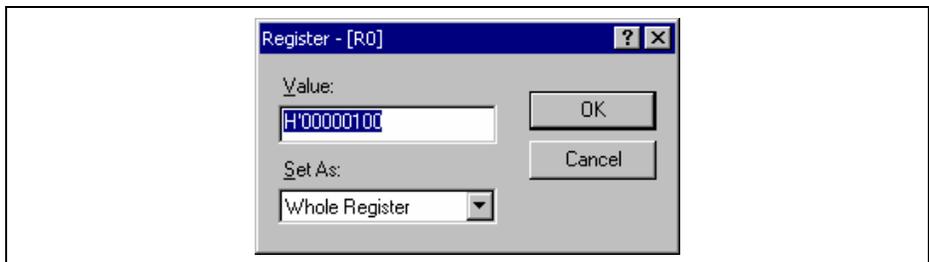


Figure 5.14 [Register] Dialog Box

You can enter a number or C/C++ expression in the [Value] field. You can choose whether to modify the whole register contents, a masked area, floating or flag bits by selecting an option from the combo box (the contents of this list depend on the CPU model and selected register).

When you have entered the new number or expression, click the [OK] button or press the [Enter] key; the dialog box closes and the new value is written into the register.

5.4.4 Using Register Contents

Use the value contained in a CPU register by specifying the register name prefixed by the “#” character, e.g.: #R1, #PC, #R6_BANK, or #R3 when you are entering a value elsewhere in the HEW, for example when displaying a specified address in the [Disassembly] or [Memory] window.

5.5 Operating Memory

This section describes how to look at memory areas in the CPU's address space. How to look at a memory area in different formats, how to fill and move a memory block, and how to load and verify a memory area with a disk file are described.

5.5.1 Viewing a Memory Area

To look at a memory area, choose [View -> CPU ->Memory...] or click the [View Memory] toolbar button () to open the [Memory] window. This will open the [Set Address] dialog box shown in figure 5.15.

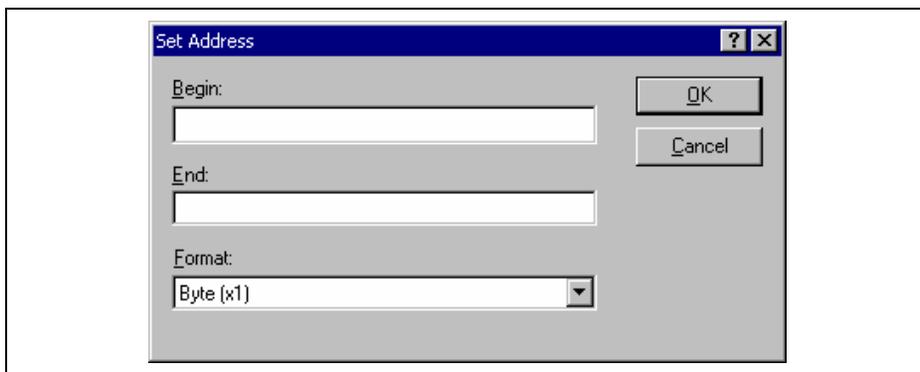
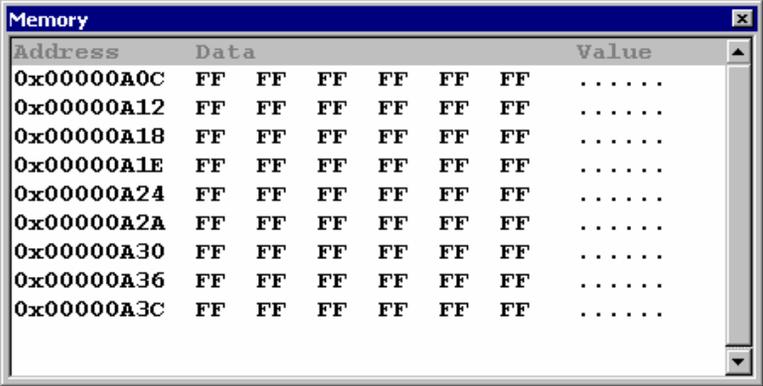


Figure 5.15 [Set Address] Dialog Box

Enter the range you wish to display as address values or equivalent symbols in the [Begin] and [End] fields. Select the data size for the display from the [Format] combo box. Click the [OK] button or press the [Enter] key, and the dialog box closes and the [Memory] window opens. The display can be scrolled within the range of the entered start and end addresses.



Address	Data	Value
0x00000A0C	FF FF FF FF FF FF
0x00000A12	FF FF FF FF FF FF
0x00000A18	FF FF FF FF FF FF
0x00000A1E	FF FF FF FF FF FF
0x00000A24	FF FF FF FF FF FF
0x00000A2A	FF FF FF FF FF FF
0x00000A30	FF FF FF FF FF FF
0x00000A36	FF FF FF FF FF FF
0x00000A3C	FF FF FF FF FF FF

Figure 5.16 [Memory] Window

Note: Each item in the [Data] column is called as the memory unit.

There are three display columns:

[Address]: Address of the first item in the [Data] column of this row.

[Data]: Data is read from the debugging platform's physical memory in the access width, and then converted to the display width.

[Value]: Data displayed in an alternative format.

5.5.2 Displaying Data in Different Formats

If you want to change the display format of the [Memory] window, select [Format] from the popup menu. The dialog box shown in figure 5.17 is displayed.

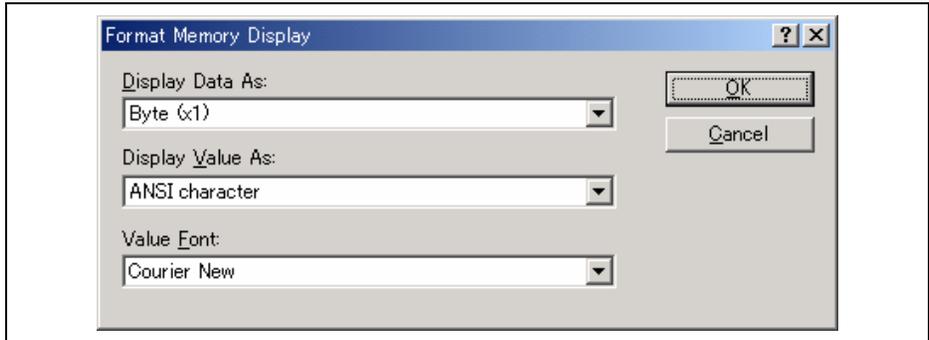


Figure 5.17 [Format Memory Display] Dialog Box

To display and edit memory in different widths, use the [Display Data As] combo box. For example, choose the [Byte] option and the display will be updated to show the memory area as individual bytes.

The data can be converted into different formats, as shown in the second column [Display Value As]. The list of formats depends on the data selection.

A separate font is selectable for the [Display Value As] column. This is useful for displaying two-byte character values when the data is being displayed in the [Word] format.

5.5.3 Splitting Up the Window Display

To vertically divide the [Memory] window display into two, select [Split] from the popup menu and move the split-up bar. Moving the split-up bar to the top end or bottom end of the window cancels the split-up display.

5.5.4 Viewing a Different Memory Area

To change the memory area displayed in the [Memory] window, use the scroll bars. To quickly look at a new address, use the [Set Address] dialog box. This can be opened by choosing [Start Address] from the popup menu.

Enter the new address value, and click the [OK] button or press the [Enter] key. The dialog box closes and the [Memory] window display is updated with the data at the new address. When an overloaded function or a class name is entered, the [Select Function] dialog box opens for you to select a function.

5.5.5 Modifying the Memory Contents

The memory contents can be modified via the [Edit Memory] dialog box. Move the cursor on the memory unit (according to the [Memory] window display choice) that you wish to change. Either double-click on the memory unit or press the [Enter] key. The dialog box shown in figure 5.18 is displayed.

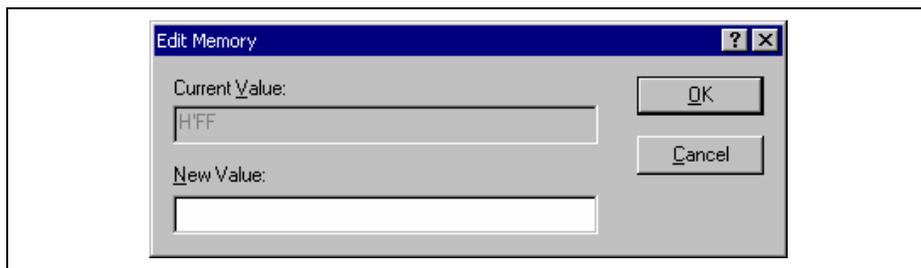


Figure 5.18 [Edit Memory] Dialog Box

A number or C/C++ expression can be entered in the [New Value] field. After you have entered the new number or expression, click the [OK] button or press the [Enter] key. Then the dialog box closes and the new value is written into memory.

The memory contents can also be modified by moving the cursor on the memory unit and entering the new value in hexadecimal through the keyboard.

5.5.6 Selecting a Memory Range

If the range to be selected is in the [Memory] window, you can select the range by clicking on the first memory unit (according to the [Memory] window display choice) and dragging the mouse to the last unit. The selected range is highlighted.

If the memory address range is larger than or outside the [Memory] window, you can enter the start address and byte count in the respective fields of the [Memory] dialog box.

5.5.7 Finding a Value in Memory

To find a value in memory, open the [Memory] window and select [Search] from the popup menu.

The [Search Memory] dialog box shown in figure 5.19 is displayed.

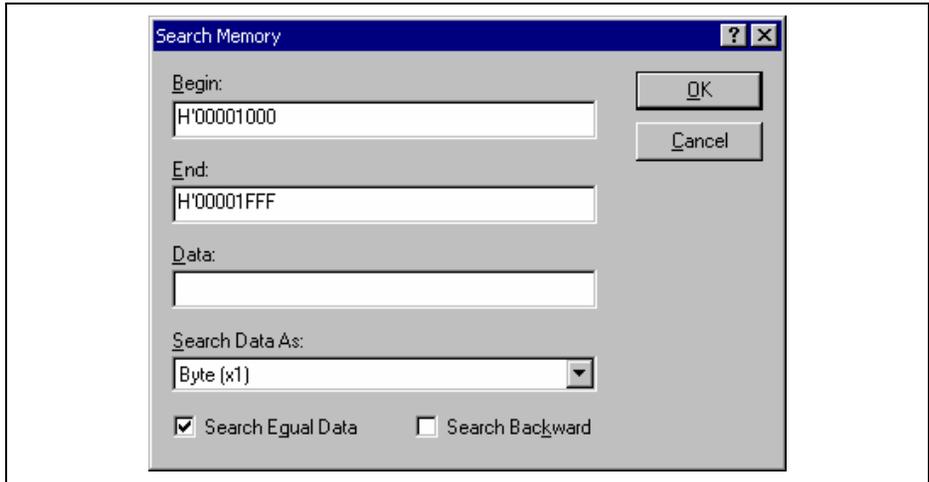


Figure 5.19 [Search Memory] Dialog Box

Enter the start and end addresses of the range in which to search (if a memory area was selected in the [Memory] window, the start and end address values will be automatically filled in) and the data value to search for, and select the search format. If pattern search is selected as the search format, a byte string of up to 256 bytes can be searched for. The end address can also be prefixed by a plus (+); the end address will become the (start address) + (entered value).

Search conditions other than pattern search are data match/mismatch and search direction. Note that only data match and forward direction can be selected with pattern search. Click the [OK] button or press the [Enter] key. The dialog box closes and the HEW searches the range for the specified data. If the data is found, the address at which the data has been found is displayed in the [Memory] window.

If the data could not be found, the [Memory] window display remains unchanged and a message box informing that the data could not be found is displayed.

If [Search Next] is selected from the popup menu in the state where data has been found, the search will be restarted from the next address.

5.5.8 Filling a Memory Area with a Value

A value can be set as the content of a memory address range using the memory fill function.

To fill a memory range with the same value, choose [Fill] from the popup menu of the [Memory] window or choose [Fill] from the [Memory] drop-down menu. The [Fill Memory] dialog box is shown in figure 5.20.

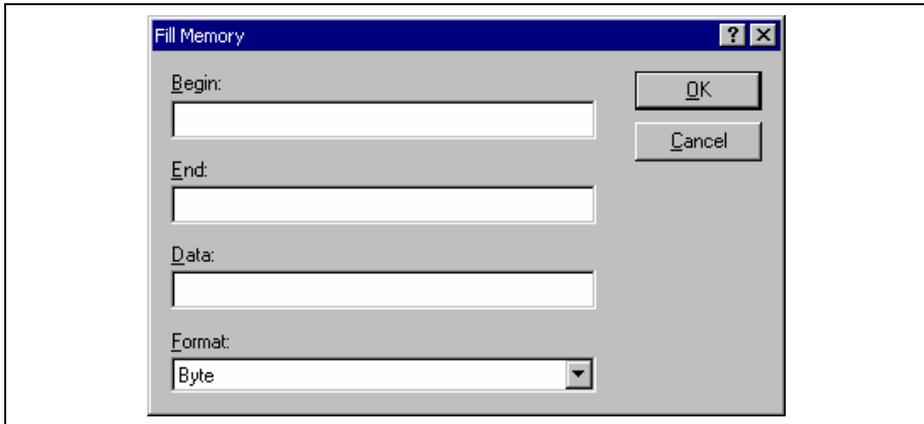


Figure 5.20 [Fill Memory] Dialog Box

If an address range has been selected in the [Memory] window, the specified start and end addresses will be displayed. Select a format from the [Format] combo box and enter the data value in the [Data] field. On clicking the [OK] button or pressing the [Enter] key, the dialog box closes and the new value is written into the memory range.

5.5.9 Copying a Memory Area

You can copy a memory area using the memory copy function. Select a memory range and then [Copy...] from the popup menu. The [Copy Memory] dialog box is opened (figure 5.21).

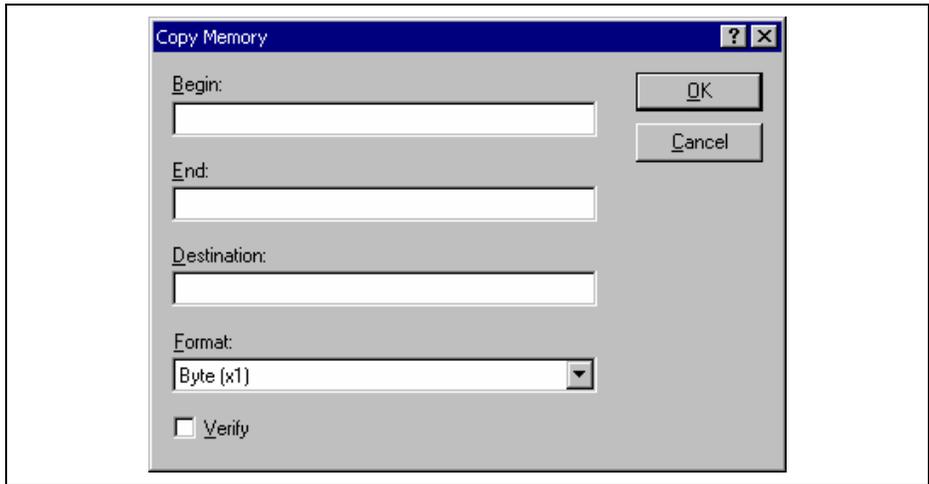


Figure 5.21 [Copy Memory] Dialog Box

The source start and end addresses selected in the [Memory] window will be displayed in the [Begin] and [End] fields. Checking the [Verify] check box enables copying while comparing the copy source and copy destination. The copy unit can be selected in the [Format] combo box. Enter the destination start address in the [Destination] field and click the [OK] button or press the [Enter] key. This will close the dialog box and copy the memory block to the new address.

5.5.10 Saving and Verifying a Memory Area

Use the memory-save function to save the memory in a specified address space to a disk file. Open the [Save Memory As] dialog box by choosing [File -> Save memory...].

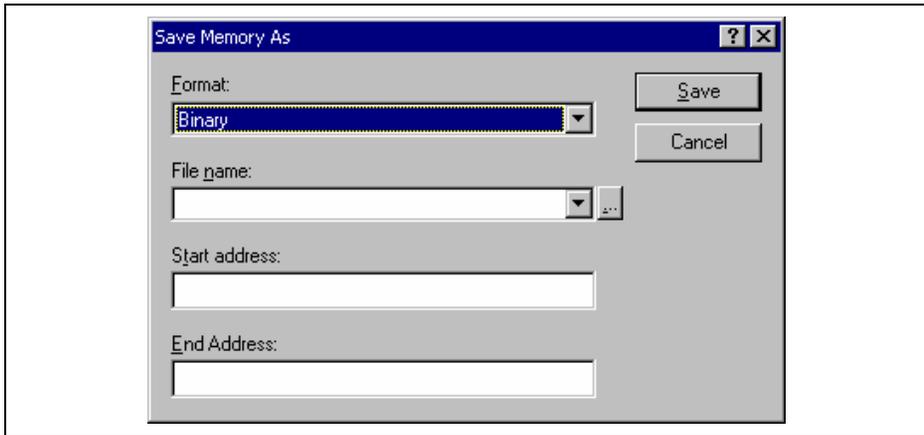


Figure 5.22 [Save Memory As] Dialog Box

Enter the start and end addresses of the memory block that you wish to save, and a name and format for the file. The [File name] combo box contains the previous four file names used for saving memory.

Clicking the [...] button can open the standard [File Save As] dialog box. On clicking the [OK] button or pressing the [Enter] key, the dialog box closes and the memory block will be saved into the disk as a file of the specified format type. When the file has been saved, a message box for confirmation is displayed.

Memory in the specified address space can be verified using the memory verify function. Open the [Verify Memory] dialog box by choosing [File -> Verify Memory...].

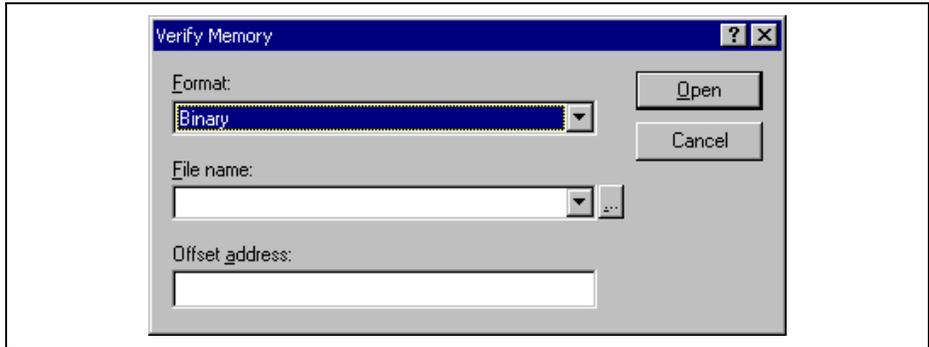


Figure 5.23 [Verify Memory] Dialog Box

5.5.11 Disabling Update of the Window Contents

Automatic update of the [Memory] window contents, which is performed when user program execution stops and in other cases, can be disabled. This is done by checking [Lock Refresh] in the popup menu.

5.5.12 Updating the Window Contents

The [Memory] window contents can be forcibly updated. This is done by checking [Refresh] in the popup menu.

5.5.13 Comparing the Memory Contents

The contents of two memory blocks can be compared. Open the [Compare Memory] dialog box by selecting [Memory -> Compare...] from the main menu or by selecting [Compare...] from the popup menu of the [Memory] window.

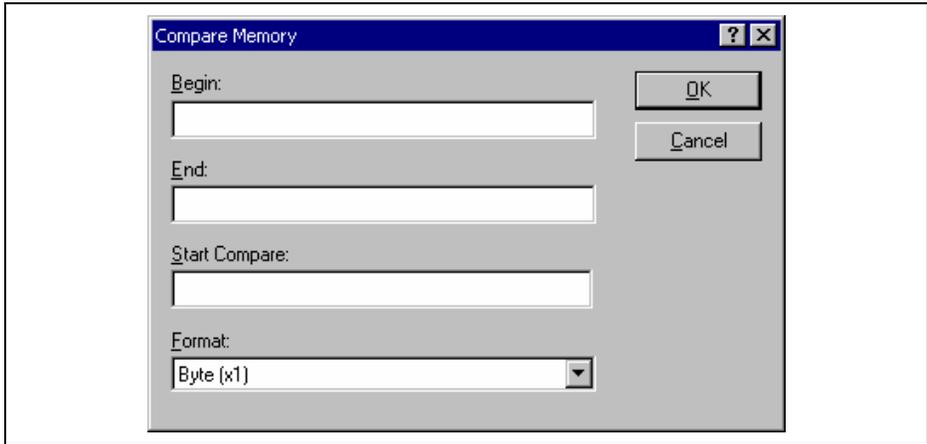


Figure 5.24 [Compare Memory] Dialog Box

Enter the comparison format ([Format]), the start address ([Begin]) and end address ([End]) of the source memory area, and the start address ([Start Compare]) of the destination memory area. If the memory block is already highlighted in the [Memory] window, the start and end addresses will be automatically filled in when the [Compare Memory] dialog box is opened.

If there is a mismatch, the address where it was found is displayed in a message box.

5.5.14 Loading a File to a Memory Area

A file can be loaded to the debugging platform's memory. Select [Load...] from the popup menu of the [Memory] window to open the [Load Program] dialog box.

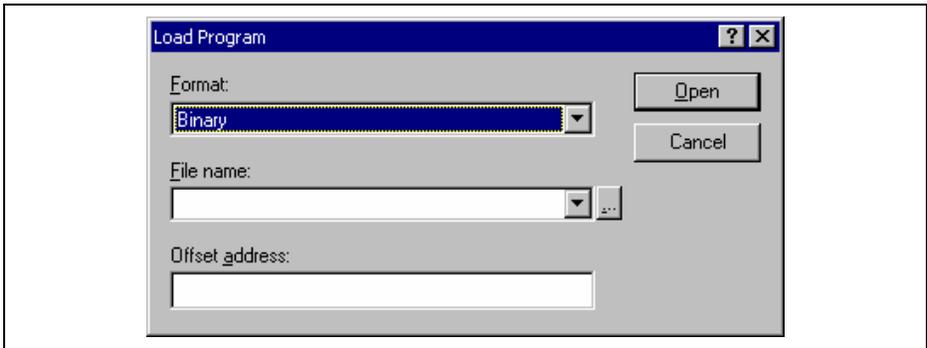


Figure 5.25 [Load Program] Dialog Box

Enter the file format ([Format]) and the file name ([File name]). If the load address value is to be changed, enter the offset value in the offset field ([Offset address]), otherwise enter zero.

5.6 Viewing the I/O Memory

A microcomputer contains on-chip peripheral modules. The exact number and type of peripheral modules differ depending on the device but the typical modules are a DMA controller, serial communications interface, A/D converter, bus state controller, and watchdog timer. Registers that are mapped to the microcomputer's address space controls the on-chip peripheral modules.

The [Memory] window enables you to look at data in continuous memory addresses as byte, word, longword, single-precision floating-point, double-precision floating-point, or ASCII values. However, registers of different sizes are allocated to non-continuous memory addresses in the I/O memory. To handle this memory, the HEW has the [IO] window to facilitate checking and setting up of these kinds of registers.

5.6.1 Opening the [IO] Window

To open the [IO] window, select [View -> CPU -> IO] or click the [View I/O] toolbar button (). Modules that match the on-chip peripheral modules organize the I/O register information. When the [IO] window is first opened, only a list of module names is displayed.

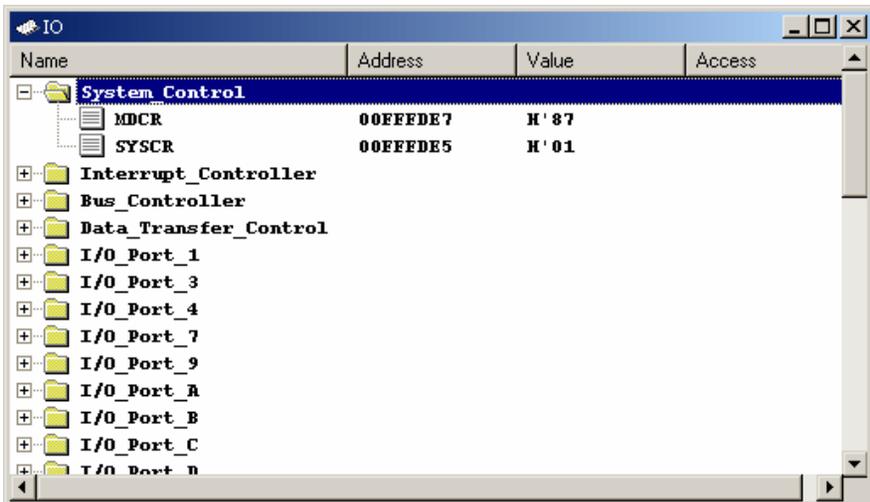


Figure 5.26 [IO] Window

5.6.2 Expanding the I/O Register Display

To display the names, addresses, and values of the I/O registers, double-click on the module name or select the module name by clicking on it or using the cursor keys and press the [Enter] key. The module display will expand to show the individual registers of that peripheral module and their names, addresses, and values. Double-clicking (or pressing the [Enter] key) again on the module name will close the I/O register display.

For a display in the bit level, expand the I/O register display in a similar way to the [Register] window.

5.6.3 Modifying the I/O Register Contents

To edit the value in an I/O register, type hexadecimal values directly into the window. To enter more complex expressions, double-click or press the [Enter] key on the register to open a dialog box to modify the register contents. When you have entered the new number or expression, click the [OK] button or press the [Enter] key; the dialog box closes and the new value is written into the register.

Note: The [IO] window displays the contents defined in [SHxxxx.io]. Editing those contents adds or deletes the registers to be displayed.

For the contents to be described as [SHxxxx.io], refer to appendix D, I/O File Format.

The following directory contains [SHxxxx.io] (xxxx means the name of emulator device group.):

<HEW2 installation directory>\Tools\Renesas\DebugComp\Platform\E10A\xxxx\IOFiles

5.7 Viewing the Current Status

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

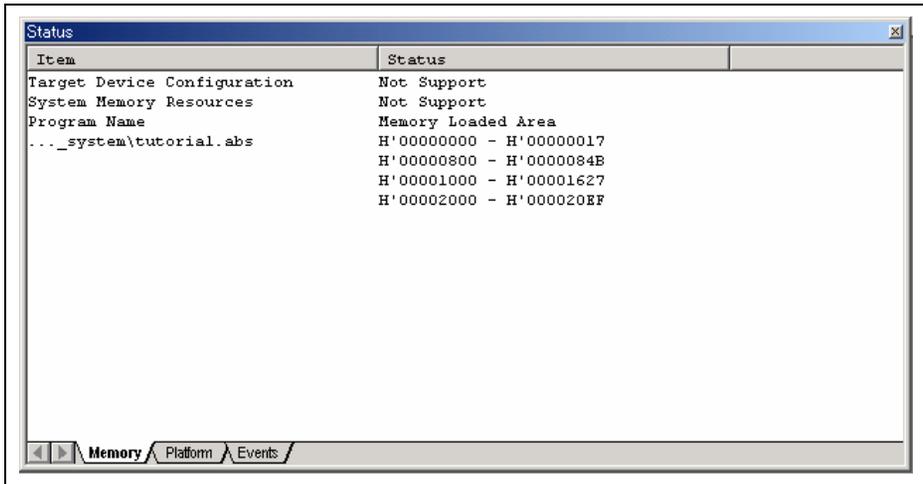


Figure 5.27 [Status] Window

The [Status] window has three sheets:

- [Memory] sheet
Contains information about the current memory status including the memory-mapping resources and the areas used by the currently loaded object file.
- [Platform] sheet
Contains information about the current status of the debugging platform, typically including CPU type and emulation mode, and the state of execution.
- [Events] sheet
Contains information about the current event (breakpoint) status, including resource information.

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

5.8 Looking at Labels

Symbol information is included in the debugging information, which is used when the HEW links the user program source code to the actual code in the memory. Symbol information is also included in the debug object file. This information is a list of names that indicate addresses in the program. These names are called labels in the HEW. The [Disassembly] window shows the first eight characters of each label instead of the corresponding address or as a part of an instruction operand.

Note: When a label value matches an operand, the corresponding instruction operand is replaced by the label. If two or more labels have the same value, the one that comes first in alphabetical order is displayed. When [Edit Control] accepts an address or a value, a label can be entered instead.

5.8.1 Looking at Label Lists

Choose [View -> Symbol -> Labels] or click the [View Labels] toolbar button  to list all labels defined in the current debugger session.

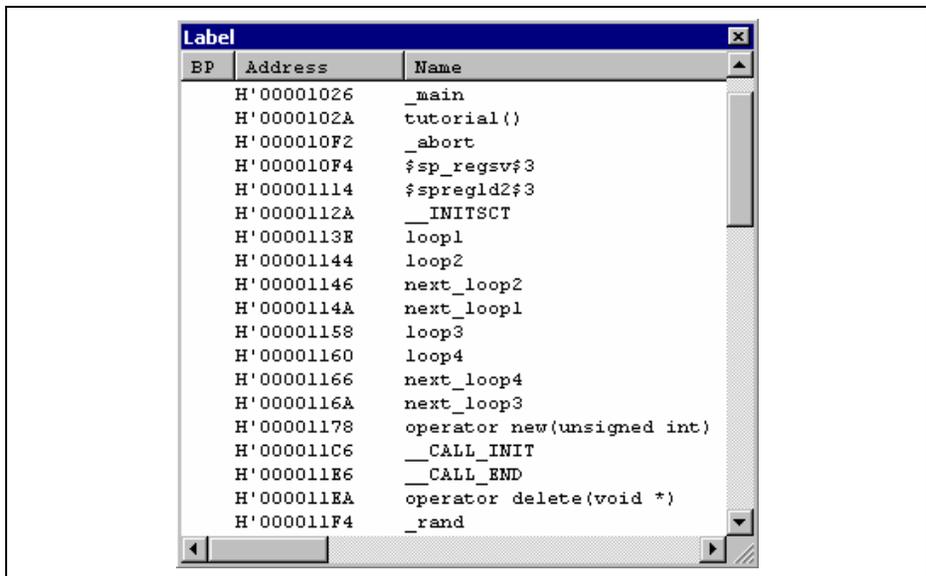


Figure 5.28 [Label] Window

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

Double-clicking on the [BP] column can set or clear a software breakpoint at the start of the function.

5.8.2 Adding a Label

Choose [Add...] from the popup menu and open the [Add Label] dialog box to add a label:

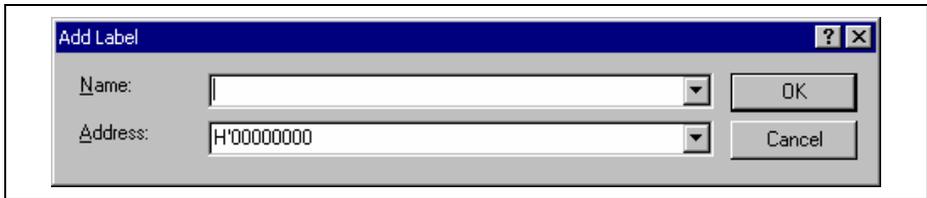


Figure 5.29 [Add Label] Dialog Box

Enter the new label name into the [Name] field and the corresponding value into the [Address] field and press [OK]. The [Add Label] dialog box closes and the label list is updated to show the new label. When an overloaded function or a class name is entered in the [Address] field, the [Select Function] dialog box opens for you to select a function. For details, refer to section 5.11.3, Supporting Duplicate Labels.

5.8.3 Editing a Label

Choose [Edit...] from the popup menu and open the [Edit Label] dialog box to edit a label:

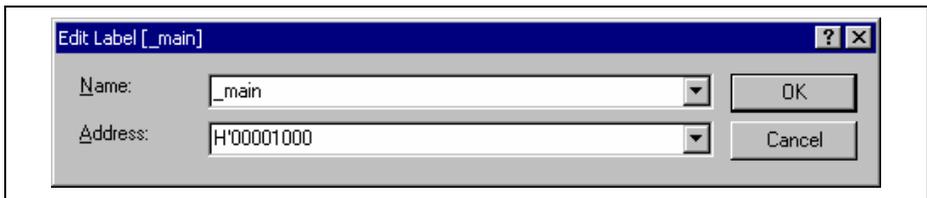


Figure 5.30 [Edit Label] Dialog Box

Edit the label name and value as required and then press [OK] to save the modified version in the label list. The list display is updated to show the new label details. When an overloaded function

or a class name is entered in the [Address] field, the [Select Function] dialog box opens for you to select a function. For details, refer to section 5.11.3, Supporting Duplicate Labels.

5.8.4 Deleting a Label

To delete a label, select the label and choose [Delete] from the popup menu. A confirmation message box appears:

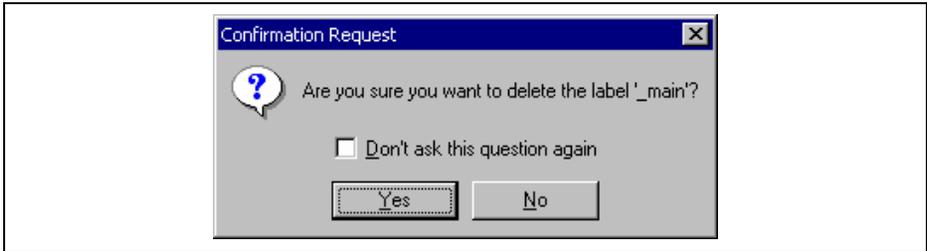


Figure 5.31 Message Box for Confirming Label Deletion

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

5.8.5 Deleting All Labels

To delete all the labels from the list, choose [Delete All] from the popup menu. A confirmation message box appears:

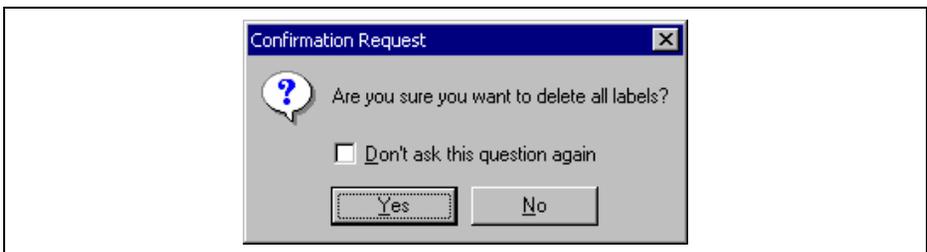


Figure 5.32 Message Box for Confirming All Label Deletion

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

5.8.6 Loading Labels from a File

A symbol file can be loaded and merged into the HEW's current symbol table. Choose [Load...] from the popup menu to open the [Open] dialog box:

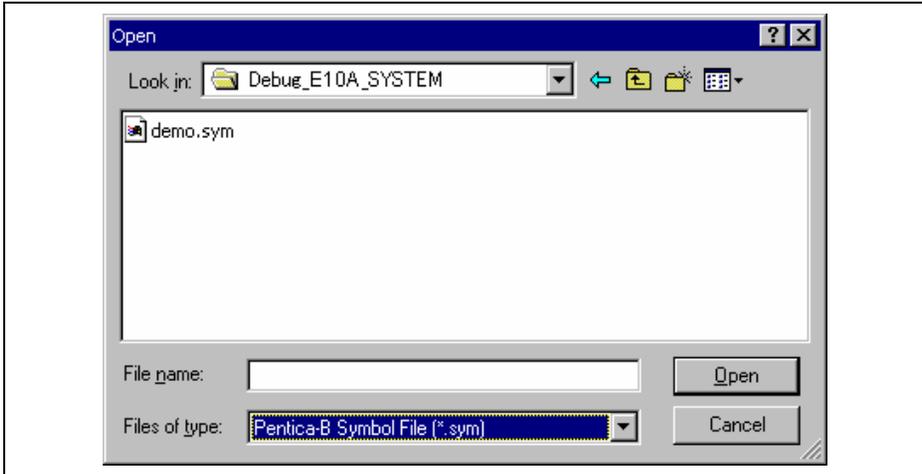


Figure 5.33 [Open] Dialog Box

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

5.8.7 Saving Labels into a File

Choose [Save As...] from the popup menu to open the [Save Symbols] dialog box. The [Save Symbols] dialog box operates like a standard Windows® [Save File As] dialog box. Enter the name for the file in the [File name] field and click [Save] to save the HEW's current label list to a symbol file. The standard file extension for symbol files is “.sym”.

See appendix E, Symbol File Format, for the symbol file format.

Once a file is specified by the [Save As...] menu, the current symbol table can be saved in the same symbol file just by choosing [Save] from the popup menu.

5.8.8 Searching for a Label

Choose [Find...] from the popup menu to open the [Find Label] dialog box:

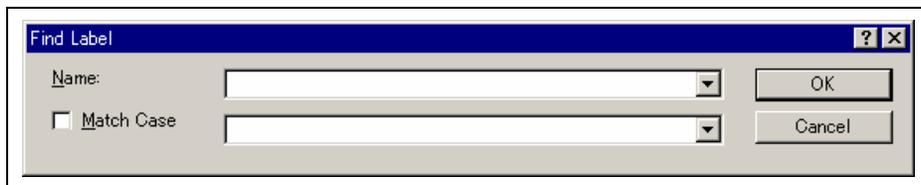


Figure 5.34 [Find Label] Dialog Box

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

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

5.8.9 Searching for the Next Label

Choose [Find Next] from the popup menu to find the next occurrence of the label containing the text that you entered.

5.8.10 Viewing the Source Corresponding to a Label

Select a label and choose [View Source] from the popup menu to open the [Editor] or [Disassembly] window containing the address corresponding to the label.

5.9 Executing Your Program

This section describes how you can execute your program's code. This section describes how to do this by either running your program continuously or stepping single or multiple instructions at a time.

5.9.1 Running from Reset

To reset the target microcomputer and run your program from the reset vector address, choose [Debug->Reset Go], or click the [Go Reset] toolbar button .

The program will run until it hits a breakpoint or a break condition is met. You can stop the program manually by choosing [Debug->Halt], or by clicking the [Halt] toolbar button .

Note: The program will start running from whatever address is stored in the reset vector location. Therefore, to run the program, it is required to make sure that this location contains the address of your startup code.

5.9.2 Continuing Run

When your program is stopped, the HEW will display a yellow arrow mark in the gutter of the line in the editor and [Disassembly] window that correspond to the CPU's current program counter (PC) address value. This will be the next instruction to be executed if you perform a step or continue running.

To continue running from the current PC address, click the [Go] toolbar button , or choose [Debug->Go].

To continue running from a specified address which is not the stop address, change the PC value in one of the following ways, and click the [Go] toolbar button  or choose [Debug->Go].

- Change the PC value in the [Register] window. Refer to section 5.4.3, Modifying Register Contents.
- Place the text cursor (not the mouse cursor) on a target line in the [Editor] or [Disassembly] window, and choose [Set PC Here] from the popup menu.

5.9.3 Running to the Cursor

Sometimes as you are going through your application you may only want to run a small section of code, that would require many single steps to execute. You can do this using the Go To Cursor feature.

➤ How to use the Go To Cursor

1. Make sure that the [Editor] or [Disassembly] window is open showing the address at which you wish to stop.
2. Position the text cursor on the address at which you wish to stop by either clicking in the [Address] field or using the cursor keys.
3. Choose [Go To Cursor] from the popup menu.

The debugging platform will run your code from the current PC value until it reaches the address indicated by the cursor's position.

- Notes:
1. If your program never executes the specified code at this address, the program will not stop. If this happens, code execution can be stopped by pressing the Esc key, choosing [Debug->Halt], or clicking on the [Halt] toolbar button ().
 2. The Go To Cursor feature requires a PC breakpoint - if you have already used all those available, then the feature will not work.

5.9.4 Running from a Specified Address

The [Run Program] dialog box allows the user to run the program from any address. Choose [Debug -> Run...] to open the [Run Program] dialog box.

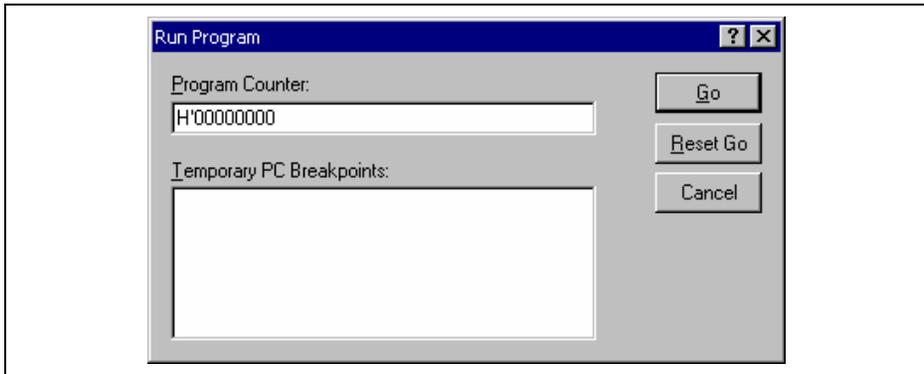


Figure 5.35 [Run Program] Dialog Box

The following execution conditions can be specified in this dialog box:

[Program Counter]: Instruction address to start execution. The initial value is the current PC value.

[Temporary PC Breakpoints]: A temporary PC breakpoint. When execution started by this dialog box stops, this breakpoint is cleared.

Note: The [Temporary PC Breakpoints] feature requires a PC breakpoint - if you have already used all those available then the feature will not work.

Clicking the [Go] button starts execution according to the settings. Clicking the [Reset Go] button starts execution from the reset vector. Clicking the [Cancel] button closes this dialog box without executing instructions.

5.9.5 Single Step

To debug your code it is very useful to be able to step a single line or instruction at a time and examine the effect of that instruction on the system. In the [Editor] window, a step operation will step a single source line. In the [Disassembly] window, a step operation will step a single assembly-language instruction. If the instruction calls another function or subroutine, you have the option to either step into or step over the function. If the instruction does not perform a call, then either option will cause the debugger to execute the instruction and stop at the next instruction.

- **Stepping Into a Function**

If you choose to step into the function the debugger will execute the call and stop at the first line or instruction of the function. To step into the function either click the [Step In] toolbar button , or choose [Debug->Step In].

- **Stepping Over a Function Call**

If you choose to step over the function the debugger will execute the call and all of the code in the function (and any function calls that that function may make) and stop at the next line or instruction of the calling function. To step over the function either click the [Step Over] toolbar button , or choose [Debug->Step Over].

- **Stepping Out of a Function**

There are occasions when you may have entered a function, finished stepping through the instructions that you want to examine and would like to return to the calling function without tediously stepping through all the remaining code in the function.

To step out of the current function either click the [Step Out] toolbar button , or choose [Debug->Step Out].

5.9.6 Multiple Steps

You can step several instructions at a time by using the [Step Program] dialog box. The dialog box also provides an automated step with a selectable delay between steps. Open it by choosing [Debug-> Step...].

The [Step Program] dialog box is displayed:

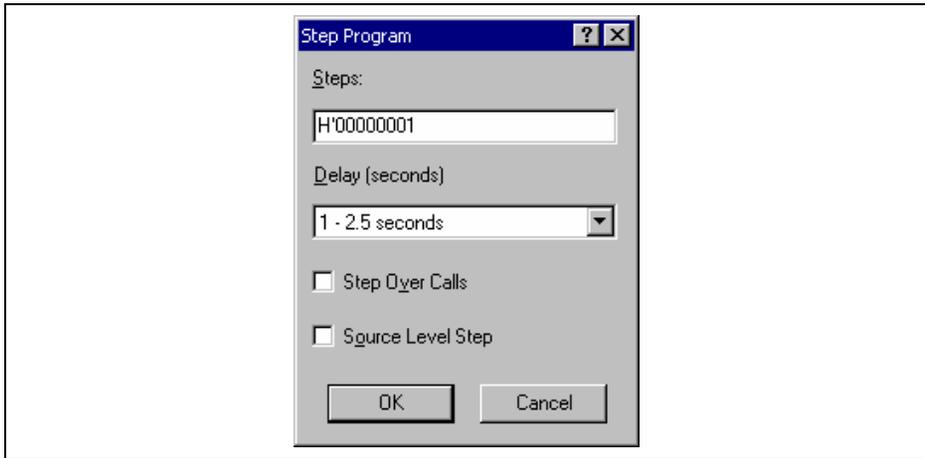


Figure 5.36 [Step Program] Dialog Box

- [Steps]: Number of steps to be executed.

- [Delay (seconds)]: Delay between steps when the program is automatically stepped.
Value 0 to 6 can be specified where value 0 indicates the longest delay.

- [Step Over Calls]: Selecting this box steps over function calls.

- [Source Level Step]: Selecting this box steps the program at the source level.

- Clicking the [OK] button or pressing the [Enter] key starts step execution.

5.10 Stopping Your Program

This section describes how you can halt execution of your application's code. This section describes how to do this directly by using the [Halt] button and by setting breakpoints at specific locations in your code.

5.10.1 Stopping the Program by the [STOP] Toolbar Button

When your program is running, the [Halt] toolbar button is enabled () (a red STOP sign), and when the program has stopped it is disabled () (the STOP sign is grayed out). Click on the [Halt] toolbar button, or choose [Debug->Halt Program].

When the program has been stopped by [Halt], "Stop" is displayed in the [Debug] sheet of the [Output] window.

5.10.2 Standard Breakpoints (PC Breakpoints)

When you are trying to debug your program you will want to be able to stop the program running when it reaches a specific point or points in your code. You can do this by setting a PC breakpoint on the line or instruction at which you want the execution to stop. The following instructions will show you how to quickly set and clear simple PC breakpoints. If more complex breakpoint operation is required, use the [Event] window, which can be opened by clicking the () button or choosing [View -> Code -> Eventpoints]. For details, refer to section 5.13, Using the Event Points.

- To set a PC breakpoint in the [Editor] window
 1. Make sure that the [Disassembly] or [Editor] window is open at the place you want to set a PC breakpoint.
 2. Choose [Toggle Breakpoint] from the popup menu, or press F9, at the line showing the address at which you want the program to stop.
 3. You will see a red circle appear in the gutter to indicate that a PC breakpoint has been set.
 4. The current breakpoint set can be enabled or disabled by using [Enable/Disable Breakpoint] in the popup menu.

Now when you run your program and it reaches the address at which you set the PC breakpoint, execution halts with the message "BREAKPOINT" displayed in the [Debug] sheet of the [Output] window, and the [Editor] or [Disassembly] window is updated with the PC breakpoint line marked with an arrow in the gutter.

Note: When a break occurs, the program stops just before it is about to execute the line or instruction at which you set a program PC breakpoint. If you choose Go or Step after stopping at the PC breakpoint, then the line marked with an arrow will be the next instruction to be executed.

- To set a PC breakpoint by using the [Breakpoints] dialog box
Selecting [Edit -> Source Breakpoint...] displays the [Breakpoints] dialog box.

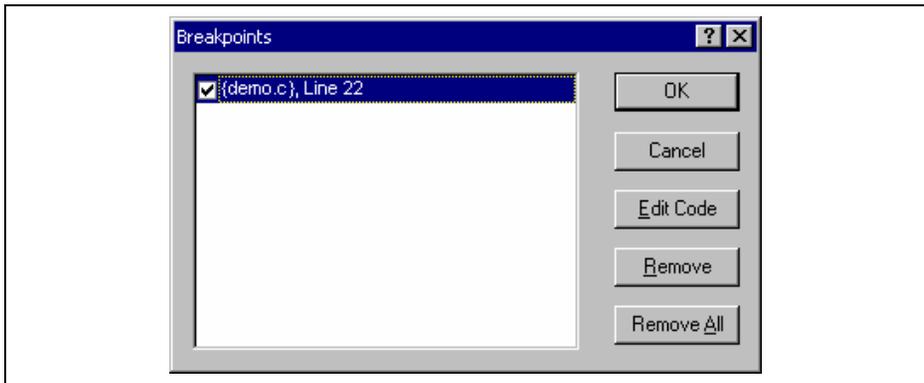


Figure 5.37 [Breakpoints] Dialog Box

The [Breakpoints] dialog box allows the user to view the current breakpoints set. Clicking the [Edit Code] button displays the source where each breakpoint is set. The [Remove] or [Remove All] button deletes one or all breakpoints, respectively. The check box of each breakpoint enables or disables the breakpoint.

- To toggle PC breakpoints
It is possible to toggle the [PC Breakpoints] setting by either double-clicking in the [BP] column of the line where the PC breakpoint is set or placing the cursor on the line and pressing the [F9] key. The setting to be toggled depends on the debugging platform.

5.11 Elf/Dwarf2 Support

The HEW supports the Elf/Dwarf2 object file format for debugging applications written in C/C++ and assembly language. It provides a powerful way of accessing, observing and modifying the symbolic level debugging information about the user application that is running.

Key Features

- Source level debugging
- C/C++ operators
- C/C++ expression (casting, pointers, references, etc.)
- Ambiguous function names
- Overlay memory loading
- Watch - locals, and user definition
- Stack trace

5.11.1 C/C++ Operators

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

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

```
Buffer_start + 0x1000
#R1 | B'10001101
((pointer + (2 * increment_size)) & H'FFFF0000) >> D'15
!(flag ^ #ER4)
```

5.11.2 C/C++ Expressions

Expression Examples

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

5.11.3 Supporting Duplicate Labels

In some languages, for example C++ overloaded functions, a label may represent more than one address. When such a label name is entered in a dialog box, the HEW will display the [Select Function] dialog box to display overloaded functions and member functions.

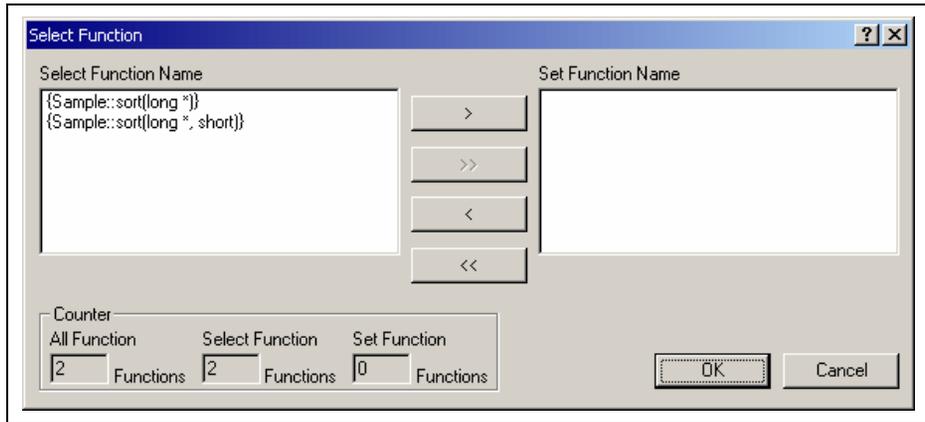


Figure 5.38 [Select Function] Dialog Box

Select overloaded functions or member functions in the [Select Function] dialog box. Generally, one function can be selected at one time; only for setting breakpoints, multiple functions can be selected. This dialog box has three areas.

- [Select Function Name]: Displays the same-name functions or member functions and their detailed information.
- [Set Function Name]: Displays the function to be set and their detailed information.
- [Counter]:
- | | |
|-------------------|--|
| [All Function] | Displays the number of same-name functions or member functions. |
| [Select Function] | Displays the number of functions displayed in the [Select Function Name] list box. |
| [Set Function] | Displays the number of functions displayed in the [Set Function Name] list box. |

Selecting a Function

Click the function you wish to select in the [Select Function Name] list box, and click the [>] button. You will see the selected function in the [Set Function Name] list box. To select all functions in the [Select Function Name] list box, click the [>>] button.

Deselecting a Function

Click the function you wish to deselect from the [Set Function Name] list box, and click the [<] button. To deselect all functions, click the [<<] button. The deselected function will be moved from [Set Function Name] list box back to the [Select Function Name] list box.

Setting a Function

Click the [OK] button to set the functions displayed in the [Set Function Name] list box. The functions are set and the [Select Function] dialog box closes.

Clicking the [Cancel] button closes the dialog box without setting the functions.

5.11.4 Debugging an Overlay Program

This section explains the settings for using the overlay functions.

Displaying Section Group

When the overlay mode is used, that is, when several section groups are assigned to the same address range, the address ranges and section groups are displayed in the [Overlay] dialog box.

Open the [Overlay] dialog box by choosing [Memory->Configure Overlay].

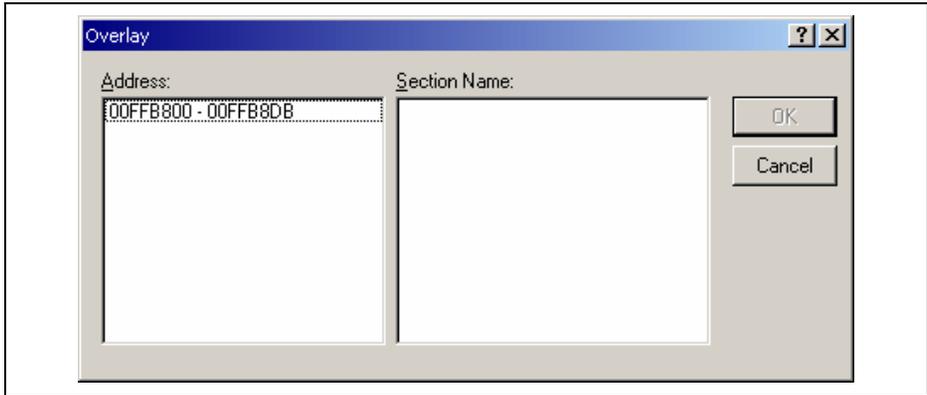


Figure 5.39 [Overlay] Dialog Box (at Opening)

This dialog box has two areas: the [Address] list box and the [Section Name] list box.

The [Address] list box displays the address ranges used in the overlay mode. Click to select one of the address ranges in the [Address] list box.

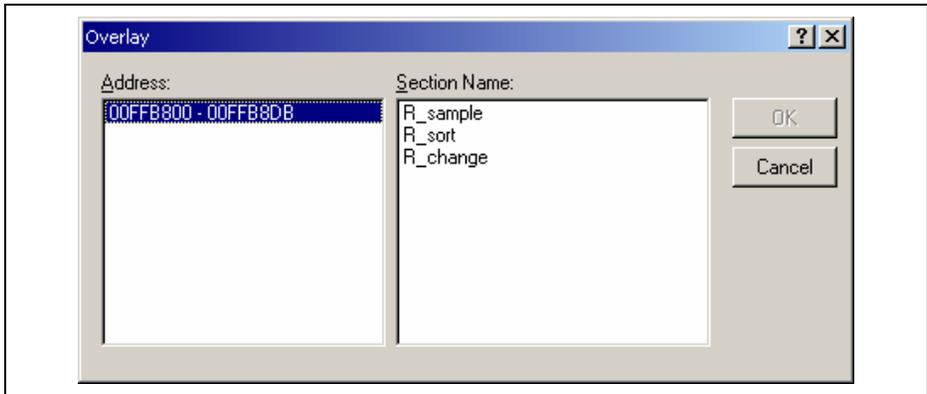


Figure 5.40 [Overlay] Dialog Box (Address Range Selected)

The [Section Name] list box displays the section groups assigned to the selected address range.

➤ Setting section group

When using the overlay function, the highest-priority section group must be selected in the [Overlay] dialog box; otherwise the HEW will operate incorrectly.

First click one of the address ranges displayed in the [Address] list box. The section groups assigned to the selected address range will then be displayed in the [Section Name] list box.

Click to select the section group with the highest-priority among the displayed section groups.

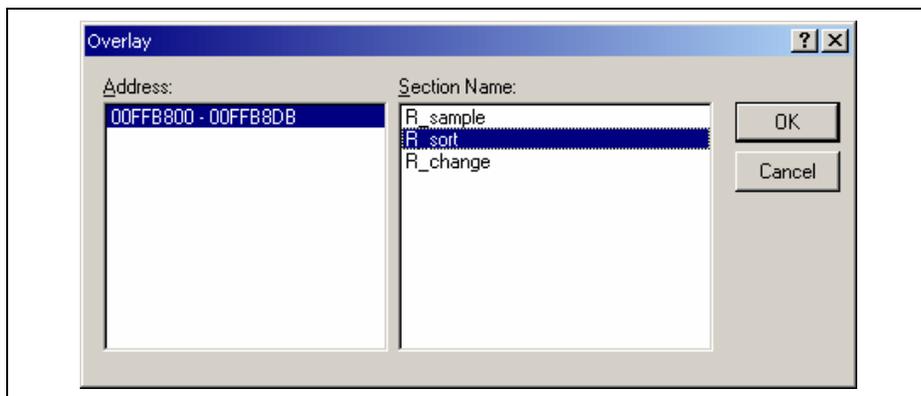


Figure 5.41 [Overlay] Dialog Box (Highest-Priority Section Group Selected)

After selecting a section group, clicking the [OK] button stores the priority setting and closes the dialog box. Clicking the [Cancel] button closes the dialog box without storing the priority setting.

Note: Within the address range used by the overlay function, the debugging information for the section specified in the [Overlay] dialog box is referred to. Therefore, the same section of the currently loaded program must be selected in the [Overlay] dialog box.

5.12 Looking at Variables

This section describes how you can look at variables in the source program.

5.12.1 Tooltip Watch

The quickest way to look at a variable in your program is to use the Tooltip Watch feature.

➡ To use Tooltip Watch:

Open the [Editor] window showing the variable that you want to examine.

Rest the mouse cursor over the variable name that you want to examine - a tooltip will appear near the variable containing basic watch information for that variable.

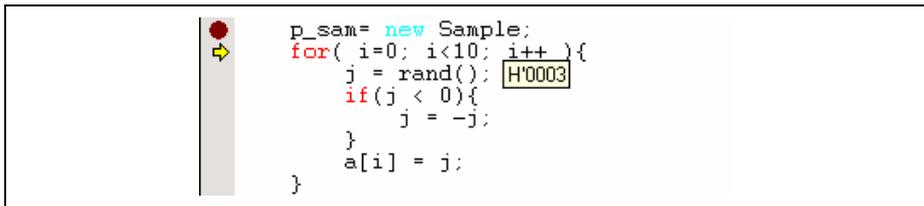


Figure 5.42 Tooltip Watch

5.12.2 Instant Watch

Open the [Editor] window showing the variable that you want to examine.

Rest the mouse cursor over the variable name that you want to examine and choose [Instant Watch...] from the popup menu; the [Instant Watch] dialog box will appear and display the variable at the cursor location.

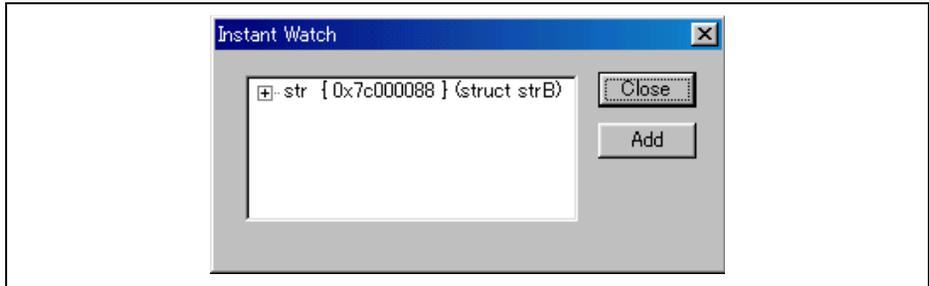


Figure 5.43 [Instant Watch] Dialog Box

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

5.12.3 [Watch] Window

You can view any value in the [Watch] window.

Opening a [Watch] Window

To open a [Watch] window, choose [View->Symbol->Watch] or click on the [Watch] toolbar button () if it is visible. A [Watch] window opens. Initially the content of the window will be blank.

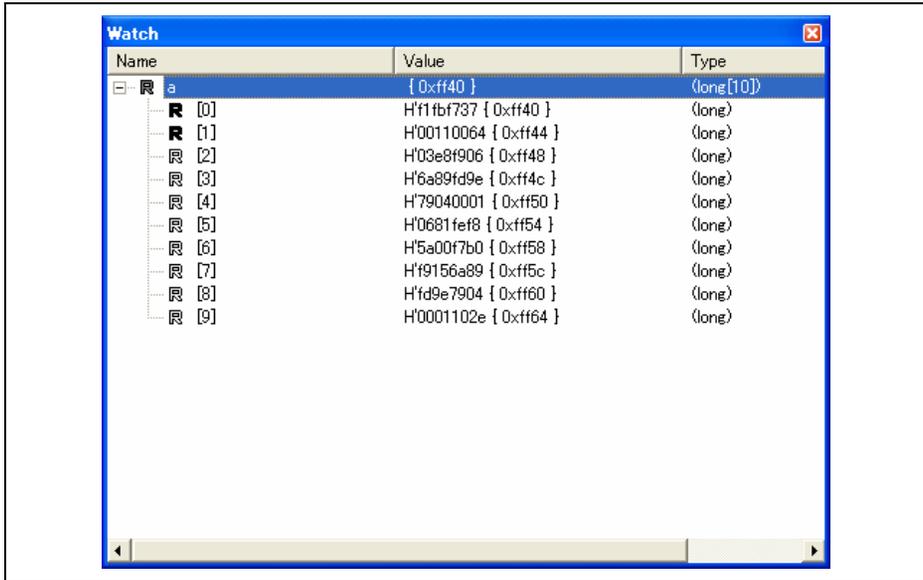


Figure 5.44 [Watch] Window

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

The following items are displayed.

- [Name]: Name of the variable
- [Value]: Value and assigned location.
 The assigned location is enclosed by { }.
- [Type]: Type of the variable

The [R] mark shows that the value of the variable can be updated during user program execution.

For updating of the content of the variable that has been registered in the [Watch] window, read the data after the break in user program execution.

Note: The realtime operation for the user program is disabled because the user program is temporarily stopped.

When the color of the [R] mark is black, a value has been updated by reading the data.

- Notes:
1. This function can be set per variable or per element or body for structures of data.
 2. The information is lost when it is scrolled out of the [Watch] window or when the window is closed.
 3. A variable that is allocated to a register cannot be selected for monitoring.

Adding a Watch Item

Use the [Add Watch] dialog box in the [Watch] window to add Watch items to the [Watch] window.

➡ To use Add Watch from the [Watch] window:

Open the [Watch] window.

Choose [Add Watch] from the popup menu.

The [Add Watch] dialog box opens:

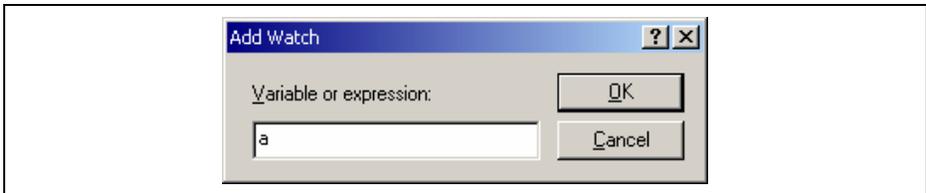


Figure 5.45 [Add Watch] Dialog Box

Enter the name of the variable that you wish to watch and click [OK]. The variable is added to the [Watch] window. A variable can be dragged from the [Editor] window and dropped into the [Watch] window.

Note: If the variable that you have added is a local variable that is not currently in scope, the HEW will add it to the [Watch] window but its value will be blank, or set to a question mark, '?'.

Expanding a Watch Item

If a watch item is a pointer, array, or structure, then you will see a plus sign (+) expansion indicator to left of its name, this means that you can expand the watch item. To expand a watch item, click on it. The item expands to show the elements (in the case of structures and arrays) or data value (in the case of pointers) indented by one tab stop, and the plus sign changes to a minus sign (-). If the elements of the watch item also contain pointers, structures, or arrays then they will also have expansion indicators next to them.

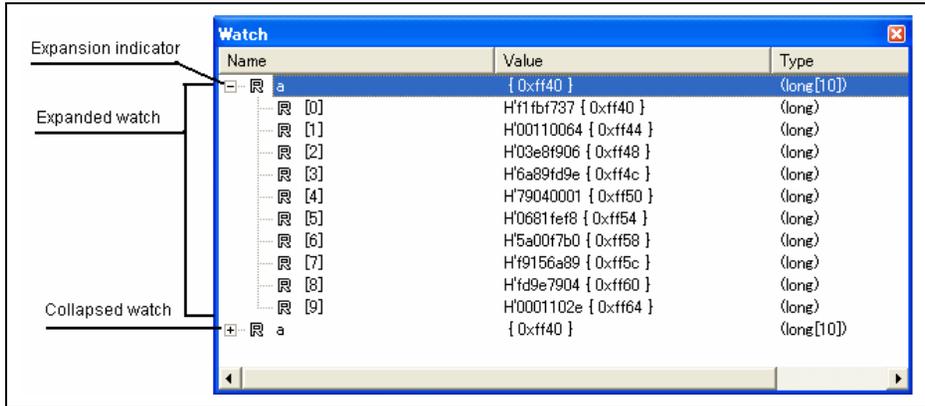


Figure 5.46 Expanding a Watch Item

To collapse an expanded watch item, double-click on the item again. The item's elements will collapse back to the single item and the minus sign changes back to a plus sign.

Editing a Watch Item's Value

You may wish to change the value of a watch variable, e.g. for testing purposes or if the value is incorrect due to a bug in your program. To change a watch item's value use the Edit Value function.

➡ Editing a watch item's value:

Enter a value directly in the window.

In another way, select the item to edit by clicking on it, you will see a flashing cursor on the item. Choose [Edit Value] from the popup menu.

The [Edit Value] dialog box opens:

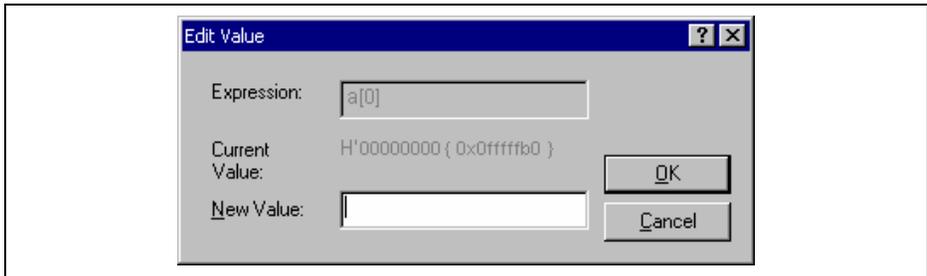


Figure 5.47 [Edit Value] Dialog Box

Enter the new value or expression in the [New Value] field and click [OK]. The [Watch] window is updated to show the new value.

Deleting a Watch Item

To delete a watch item, select it and choose [Delete] from the popup menu. The item is deleted and the [Watch] window is updated.

To delete all watch items, choose [Delete All] from the popup menu. All items are deleted and the [Watch] window is updated.

Specifying Automatic Update

The [R] mark shown to the left of each variable indicates whether the variable is updated in real time. When an [R] mark is displayed in bold face, the value of the variable will be updated in realtime during user program execution.

A popup menu containing the following options is available in the [Watch] window:

- Auto Update
Marks the selected variable with a bold [R] and updates the variable automatically.
- Auto Update All
Marks all variables with bold [R]s and updates all variables automatically.
- Delete Auto Update
Marks the selected variable with an outlined [R] and cancels automatic update.
- Delete Auto Update All
Marks all variables with outlined [R]s and cancels automatic update.

Modifying the Radix

The radix for the selected variable display can be modified by choosing [Radix] from the popup menu.

Saving the [Watch] Window Contents in a File

To save the contents of the [Watch] window, choose [Save As...] from the popup menu; the [Save As] dialog box opens. It allows the user to specify the name of a file and to save the contents of the [Watch] window in the file. If the [Append] check box is selected, the window contents are appended to the existing file, and if it is not selected, the existing file is overwritten.

Opening a [Memory] Window

The contents of the memory area to which the selected variable is assigned can be displayed in the [Memory] window. Choose [Go To Memory...] from the popup menu; the [Set Address] dialog box opens, showing the information (start address, end address, and size) of the selected variable as default. Clicking [OK] opens the [Memory] window.

5.12.4 [Locals] Window

The local variables and their values can be displayed in the [Locals] window.

Opening the [Locals] Window

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

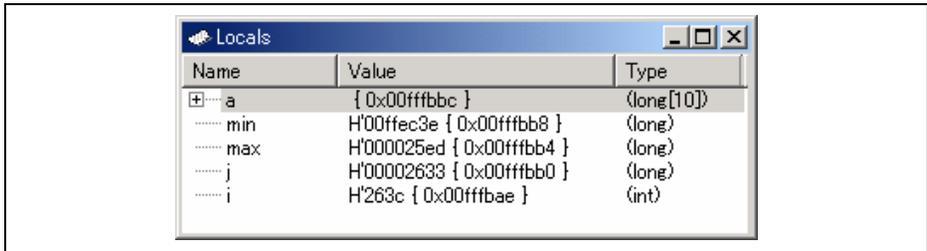


Figure 5.48 [Locals] Window

If a local variable is not initialized when defined, then the value of the local variable will be incorrect until another value is assigned to the local variable.

The local variable values and the radix for local variable display can be modified in the same manner as in the [Watch] window.

5.13 Using the Event Points

The emulator has the event point function that performs breaking, tracing, and execution time measurement by specifying higher-level conditions along with the PC breakpoints standard for the HEW.

5.13.1 PC Breakpoints

When the instruction of the specified address is fetched, the user program is stopped. Up to 255 points can be set.

5.13.2 Break Conditions

Break conditions can be used for higher-level conditions such as the data condition as well as specification of the single address.

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

Several break conditions can be used to set more complex conditions.

Note: The break conditions that can be set vary according to the emulator in use. For details, refer to the online help.

5.13.3 Opening the [Event] Window

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

The [Event] window has the following two sheets:

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

[Break condition] sheet: Displays or sets the settings made for break condition channels.

5.13.4 Setting PC Breakpoints

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

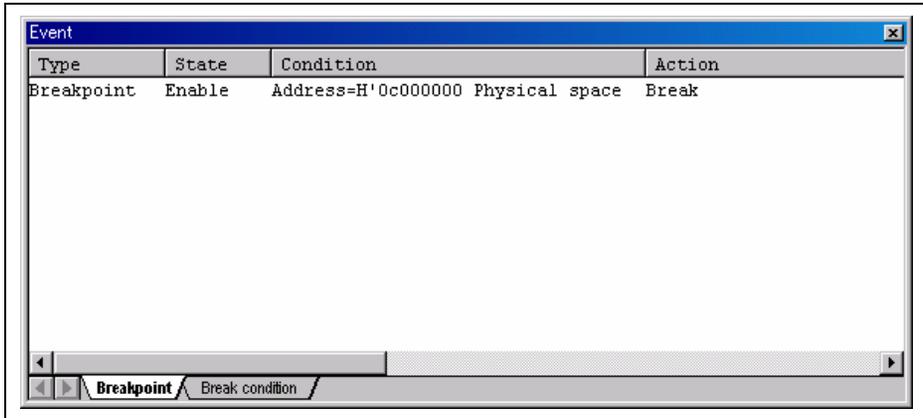


Figure 5.49 [Event] Window (Breakpoint Sheet)

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

[Type] Breakpoint

[State] Whether the breakpoint is enabled or disabled

[Condition] An address that the breakpoint is set
Address = Program counter (Corresponding file name, line, and symbol name)

[Action] Operation of the emulator when a break condition is satisfied
Break: Halts execution

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

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

5.13.5 Add

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

5.13.6 Edit

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

5.13.7 Enable

Enables the selected breakpoint(s).

5.13.8 Disable

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

5.13.9 Delete

Removes the selected breakpoint. To retain the details of the breakpoint but not have it cause a break when its conditions are met, use the Disable option (see section 5.13.8, Disable).

5.13.10 Delete All

Removes all breakpoints.

5.13.11 Go to Source

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

5.13.12 [Breakpoint] Dialog Box

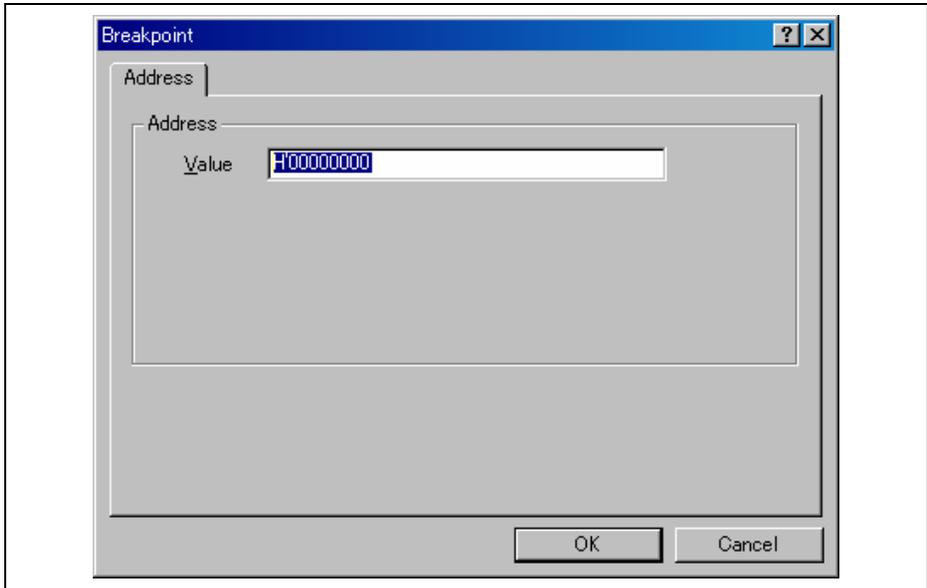


Figure 5.50 [Breakpoint] Dialog Box

This dialog box specifies break conditions.

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

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

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

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

5.13.13 Setting Break Conditions

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

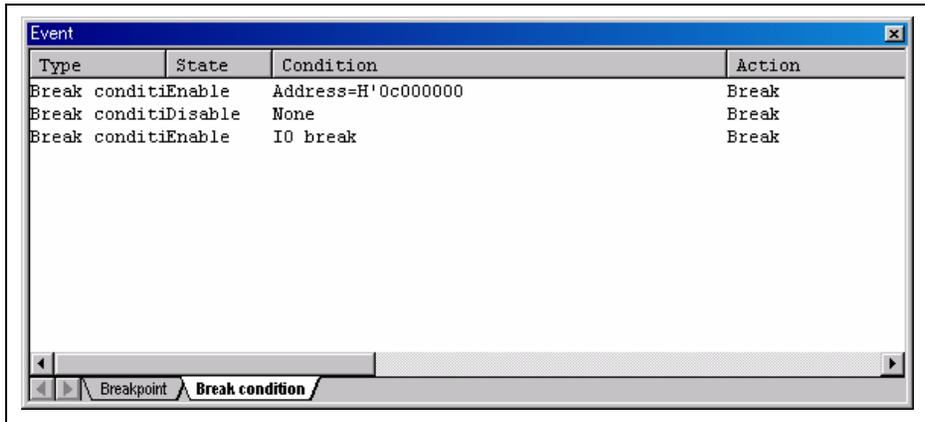


Figure 5.51 [Event] Window ([Break condition] Sheet)

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

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

[Type] Break channel number

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

[Condition] A condition that satisfies a break. The displayed contents differ depending on the break type.

[Action] Operation of the emulator when a break condition is satisfied.
 Break: Halts execution

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

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

5.13.14 Edit...

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

5.13.15 Enable

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

5.13.16 Disable

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

5.13.17 Delete

Initializes the condition of the selected break channel. To retain the details of the break channel but not have it cause a break when its conditions are met, use the Disable option (see section 5.13.16, Disable).

5.13.18 Delete All

Initializes conditions of all break channels.

5.13.19 Go to Source

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

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

5.13.20 Sequential Conditions

Sets the sequential condition of the break channel.

5.13.21 Editing Break Conditions

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

5.13.22 Modifying Break Conditions

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

5.13.23 Enabling Break Conditions

Select a break condition and choose [Enable] from the popup menu to enable the selected break condition.

5.13.24 Disabling Break Conditions

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

5.13.25 Deleting Break Conditions

Select a break condition and choose [Delete] from the popup menu to remove the selected break condition. To retain the break condition but not have it cause an event when its conditions are met, use the [Disable] option (see section 5.13.24, Disabling Break Conditions).

5.13.26 Deleting All Break Conditions

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

5.13.27 Viewing the Source Line for Break Conditions

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

5.14 Viewing the Trace Information

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

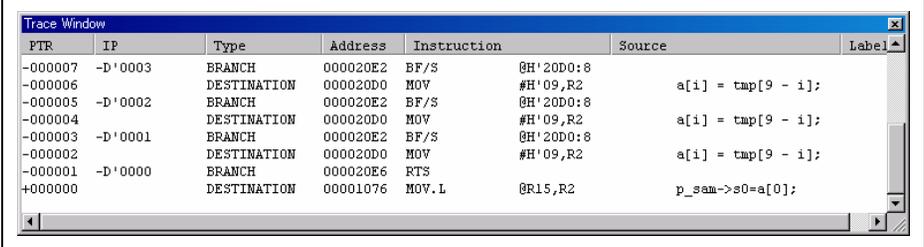
5.14.1 Opening the [Trace] Window

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

5.14.2 Acquiring Trace Information

When the emulator does not set the acquisition condition of the trace information, the trace information is acquired by the internal trace function in default.

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



PTR	IP	Type	Address	Instruction	Source	Label
-000007	-D'0003	BRANCH	000020E2	BF/S	@H'20D0:8	
-000006		DESTINATION	000020D0	MOV	#H'09,R2	a[i] = tmp[9 - i];
-000005	-D'0002	BRANCH	000020E2	BF/S	@H'20D0:8	
-000004		DESTINATION	000020D0	MOV	#H'09,R2	a[i] = tmp[9 - i];
-000003	-D'0001	BRANCH	000020E2	BF/S	@H'20D0:8	
-000002		DESTINATION	000020D0	MOV	#H'09,R2	a[i] = tmp[9 - i];
-000001	-D'0000	BRANCH	000020E6	RTS		
+000000		DESTINATION	00001076	MOV.L	@R15,R2	p_sam->s0=a[0];

Figure 5.52 [Trace] Window (Internal Trace)

This window displays the following trace information items:

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

[IP] The amount of acquired trace information

[Type] Type of branch:
 BRANCH: Branch source
 DESTINATION: Branch destination

[Address] Instruction address

[Instruction] Instruction mnemonic

[Data]	The data of the generated data access. When [Type] is S_TRACE, value x, a variable of function Trace(x), is displayed.
[Instruction]	Instruction mnemonic
[Repeat]	Displayed only when the Repeat filter is used. This item shows the number of consecutive branch operations.
[Probe]	State of the input probe
[Timestamp]	Timestamp value
[Source]	The C/C++ or assembly-language source program
[Label]	Label information

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

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

5.14.3 Specifying Trace Acquisition Conditions

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

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

The [Acquisition] dialog box has the following pages:

Table 5.1 [Acquisition] Dialog Box Pages

Page	Item
[Trace mode]	Sets trace acquisition conditions.
[Window trace]	Sets window trace acquisition conditions.* ²
[AUD Branch trace]	Sets branch conditions acquired by the AUD trace function.* ¹
[Branch Trace]	Sets branch conditions acquired by the internal trace function.

Notes: 1. This dialog box is not supported by the products that do not support the AUD trace function.
 2. Some products do not support this page. For details, refer to the online help for each product.

(1) [Trace mode] page

Sets trace acquisition conditions.

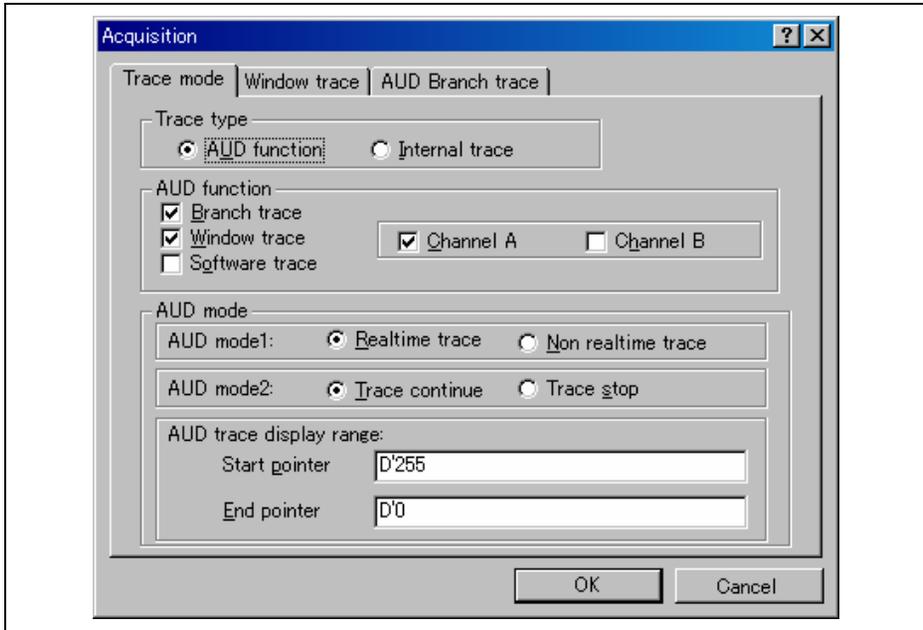


Figure 5.54 [Acquisition] Dialog Box ([Trace mode] Page)

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

[Trace type]: Selects the type of the trace function.

[AUD function] Uses the AUD trace function.

[Internal trace] Uses the internal trace function.

The following can only be set when the AUD trace function is used.

[AUD function]: Sets the trace acquisition condition.

For a description of the conditions for trace acquisition, refer to section 2.2.2, AUD Trace Function, in the Debugger Part.

- [AUD mode]: Sets the acquisition mode of the AUD trace.
- [AUD mode 1]: An option to determine the operation when the trace information is continuously generated.
- [Realtime trace] Some of the trace information will not be output.
 - [Non realtime trace] Where necessary, the CPU waits until each item of trace information has been output.
- [AUD mode 2]: An option that determines the operation when the trace buffer of the emulator becomes full.
- [Trace continue] The oldest trace information is overwritten by the latest information.
 - [Trace stop] Trace information is not acquired after the buffer has been filled.
- [AUD trace display range]: Set the display range of the trace window.
- [Start pointer] The trace is displayed from the specified value.
 - [End pointer] The trace is displayed to the specified value.

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

(2) [Window Trace] page

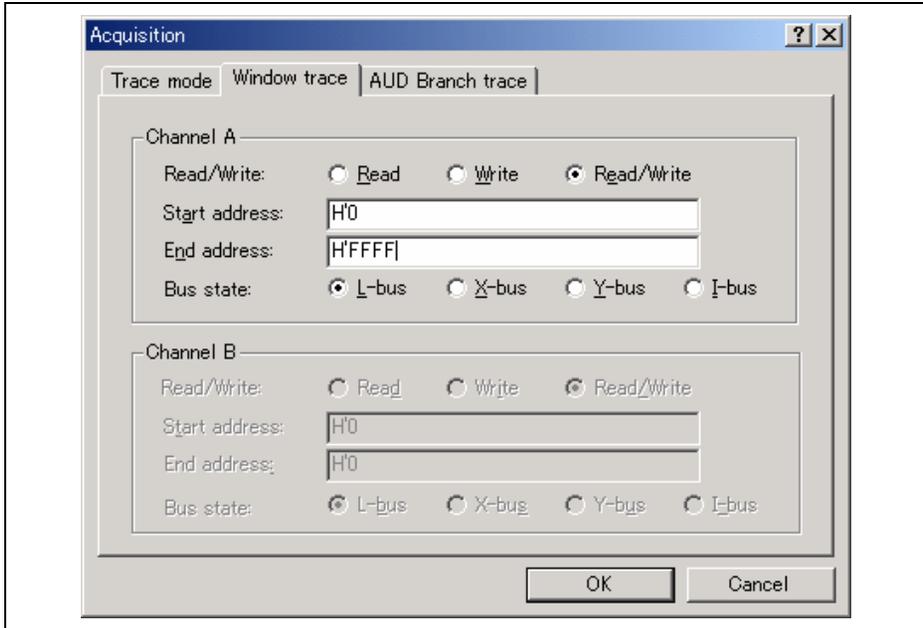


Figure 5.55 [Acquisition] Dialog Box ([Window trace] Page)

This dialog box is used to specify conditions for the acquisition of trace information. For a description of the conditions for trace acquisition, refer to section 2.2.2, AUD Trace Functions, in the Debugger Part.

[Channel A]: Enables channel A of the window trace.

- [Bus state] Selects a bus for trace acquisition.
- [Read/Write] Sets tracing of read or write access or both.
- [Trace start address] Sets an address range for the tracing of data access. The start address is set here.
- [Trace end address] Sets an address range for the tracing of data access. The end address is set here.

[Channel B]: Enables channel B of the window trace.

- [Bus state] Selects a bus for trace acquisition.

High-performance Embedded Workshop User's Manual

[Read/Write]	Sets tracing of read or write access or both.
[Trace start address]	Sets an address range for the tracing of data access. The start address is set here.
[Trace end address]	Sets an address range for the tracing of data access. The end address is set here.

(3) [AUD Branch trace] page

Selects the type of branches to be acquired.

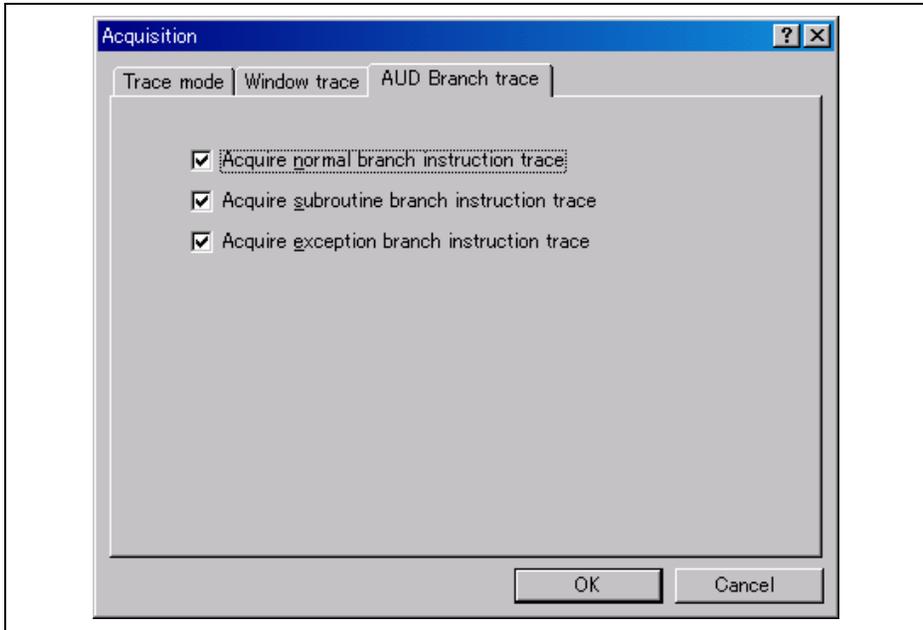


Figure 5.56 [Acquisition] Dialog Box ([AUD Branch trace] Page)

5.14.4 Searching for a Trace Record

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

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

Table 5.2 [Trace Find] Dialog Box Pages

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

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

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

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

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

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

(1) [General] page

Set the range for searching.

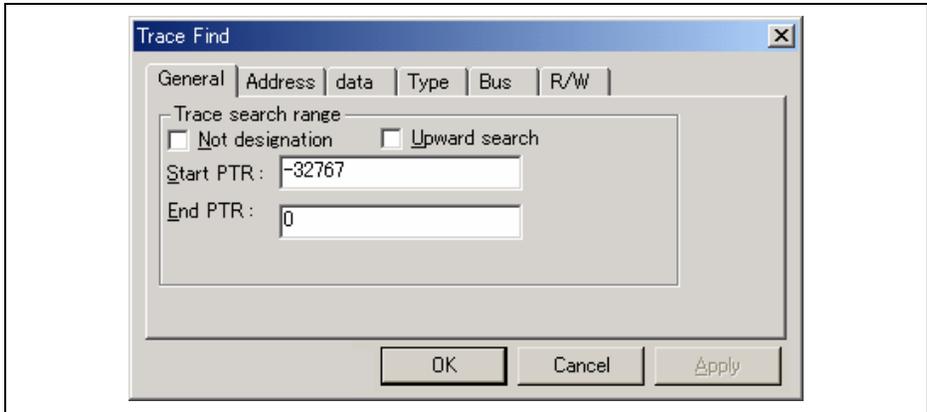


Figure 5.57 [Trace Find] Dialog Box ([General] Page)

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

[Not designation]: Searches for information that does not match the conditions set in other pages when this box is checked.

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

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

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

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

(2) [Address] page

Set an address condition.

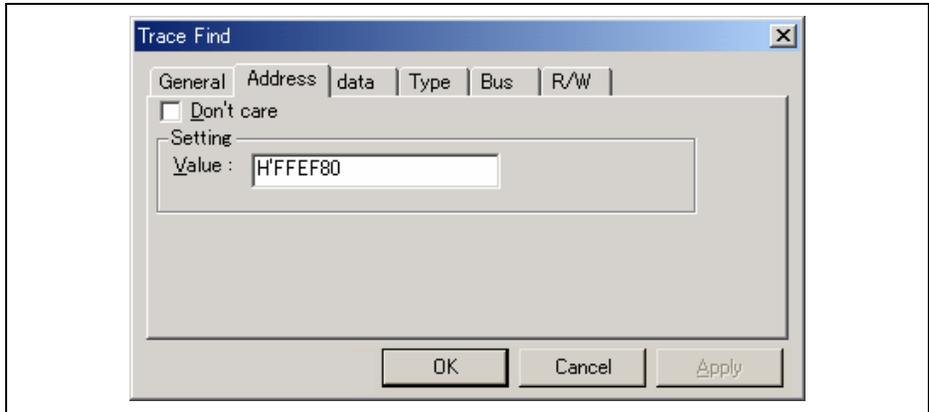


Figure 5.58 [Trace Find] Dialog Box ([Address] Page)

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

[Setting]: Detects the specified address.

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

(3) [Data] page

Set a data condition.

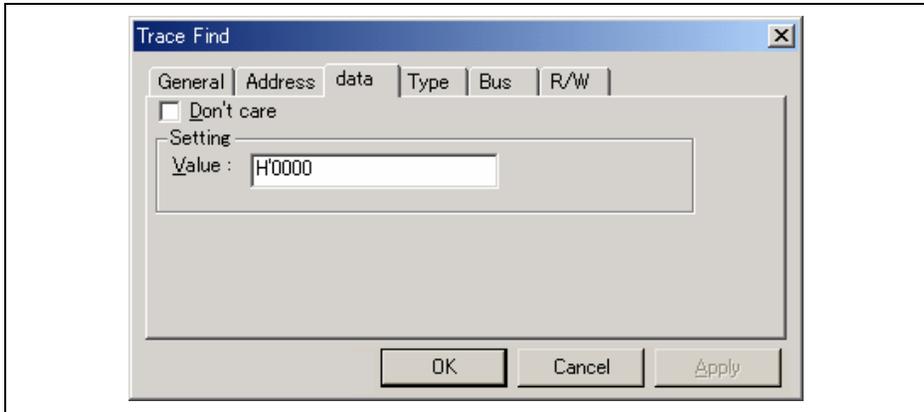


Figure 5.59 [Trace Find] Dialog Box ([data] Page)

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

[Setting]: Detects the specified data.

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

(4) [R/W] page

Select the type of access cycles.

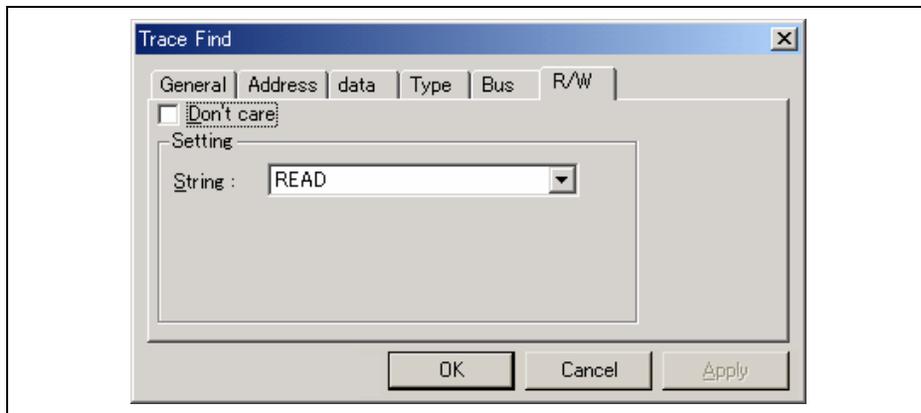


Figure 5.60 [Trace Find] Dialog Box ([R/W] Page)

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

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

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

 READ: Read cycle
 WRITE: Write cycle

(5) [Type] page

Select the type being accessed. The selection is not available when a time stamp is acquired.

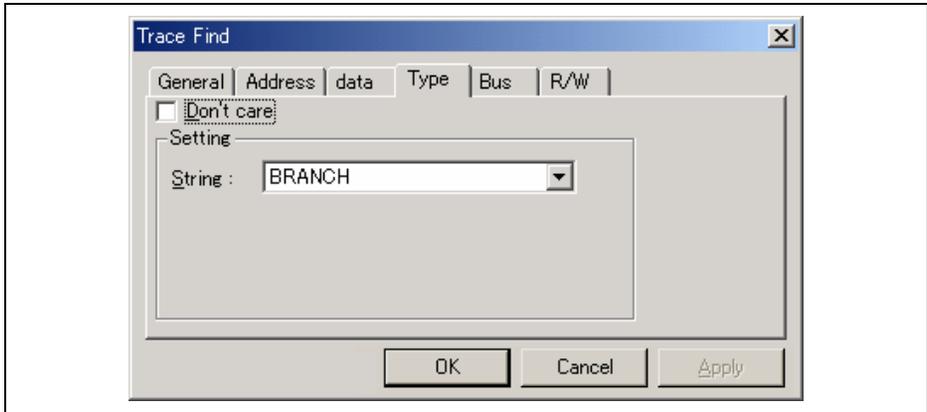


Figure 5.61 [Trace Find] Dialog Box ([Type] Page)

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

[Setting]: Detects the specified type condition.

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

(6) [Bus] page

Select the status of a bus.

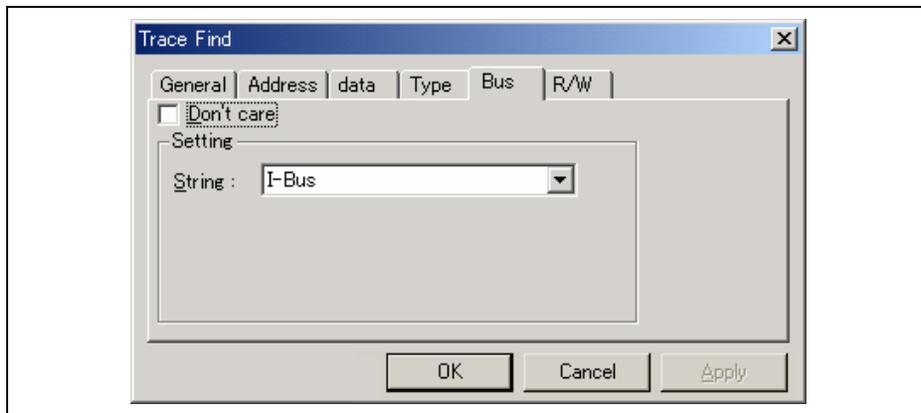


Figure 5.62 [Trace Find] Dialog Box ([Bus] Page)

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

[Setting]: Detects the specified bus condition.

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

5.14.5 Clearing the Trace Information

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

5.14.6 Saving the Trace Information in a File

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

Note: In filtering of trace information, the range to be saved cannot be selected. All the trace information displayed in the [Trace] window after filtering will be saved. Select a filtering range on the [General] page in the [Trace Filter] dialog box if you want to save the selected range. For details on the filtering function, refer to section 5.14.12, Extracting Records from the Acquired Information.

5.14.7 Viewing the [Editor] Window

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

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

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

5.14.8 Trimming the Source

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

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

5.14.9 Temporarily Stopping Trace Acquisition

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

5.14.10 Extracting Records from the Acquired Information

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

Use the filtering function in the [Trace Filter] dialog box. To open the [Trace Filter] dialog box, select [Filter...] from the popup menu.

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

Table 5.3 [Trace Filter] Dialog Box Pages

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

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

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

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

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

(1) [General] page

Set the range for filtering.

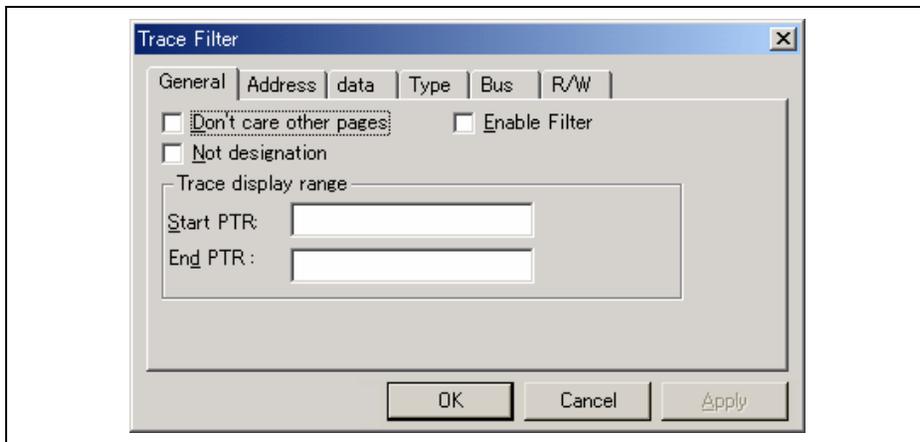


Figure 5.63 [Trace Filter] Dialog Box ([General] Page)

[Don't care other pages]: Only selects the cycle number when this box is checked. Other options become invalid.

[Enable Filter]: Enables the filter when this box is checked.

[Not designation]: Filters information that does not match the conditions set in those pages when this box is checked.

[Trace display range]: Sets the range for filtering.

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

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

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

(2) [Address] page

Set address conditions.

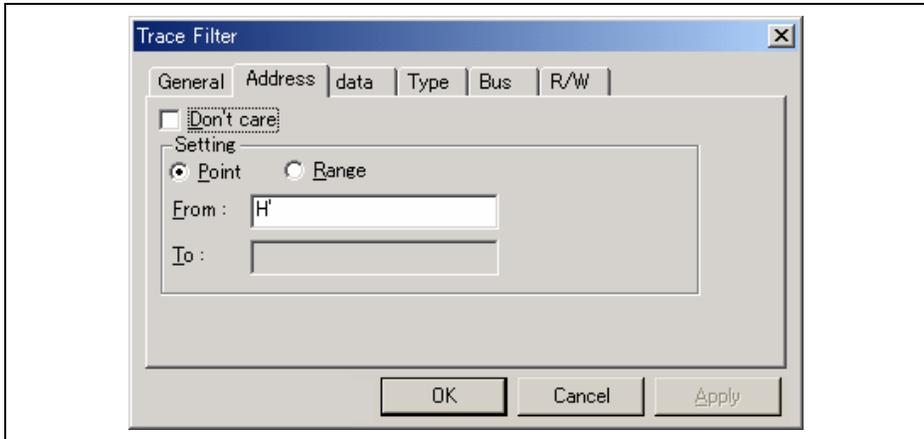


Figure 5.64 [Trace Filter] Dialog Box ([Address] Page)

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

[Setting]: Detects the specified address.

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

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

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

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

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

(3) [Data] page

Set a data condition.

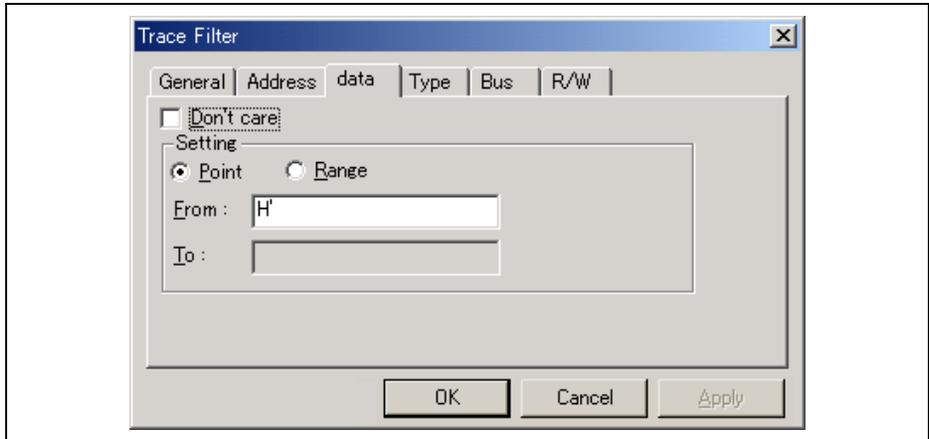


Figure 5.65 [Trace Filter] Dialog Box ([data] Page)

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

[Setting]: Detects the specified data.

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

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

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

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

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

(4) [R/W] page

Select the type of access cycles.

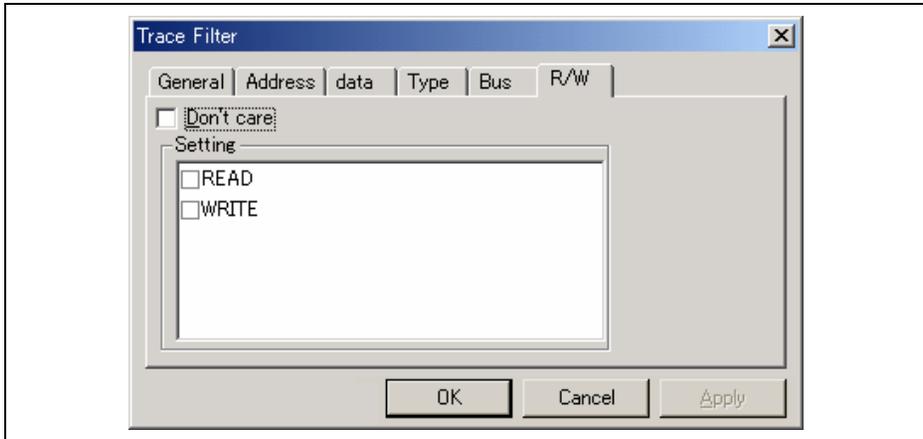


Figure 5.66 [Trace Filter] Dialog Box ([R/W] Page)

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

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

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

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

(5) [Type] page

Select the type being accessed. The selection is not available when a time stamp is acquired.

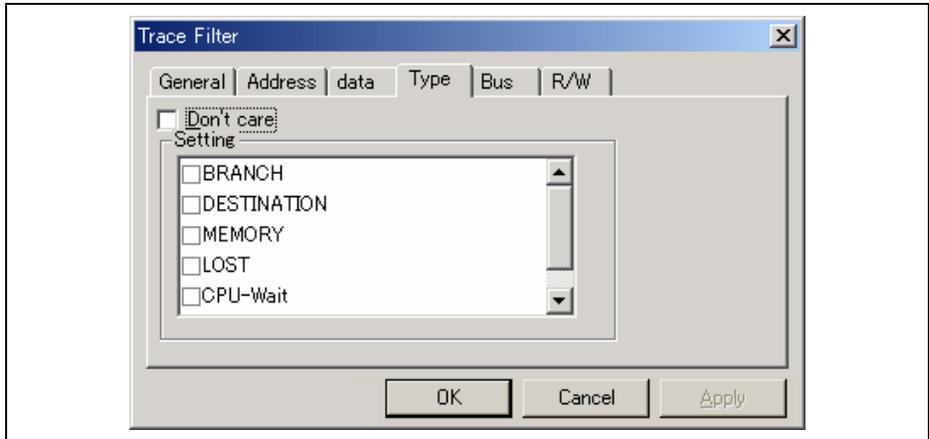


Figure 5.67 [Trace Filter] Dialog Box ([Type] Page)

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

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

(6) [Bus] page

Select the status of a bus. The selection is not available when a time stamp is acquired.

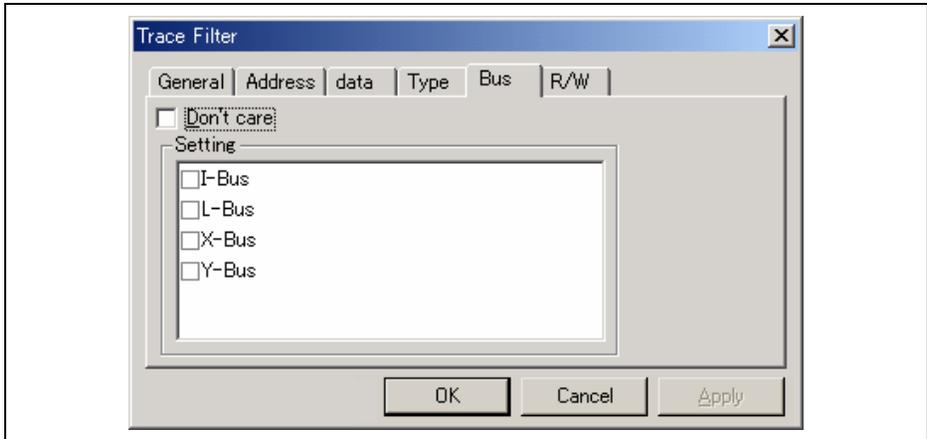


Figure 5.68 [Trace Filter] Dialog Box ([Bus] Page)

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

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

5.14.11 Analyzing Statistical Information

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

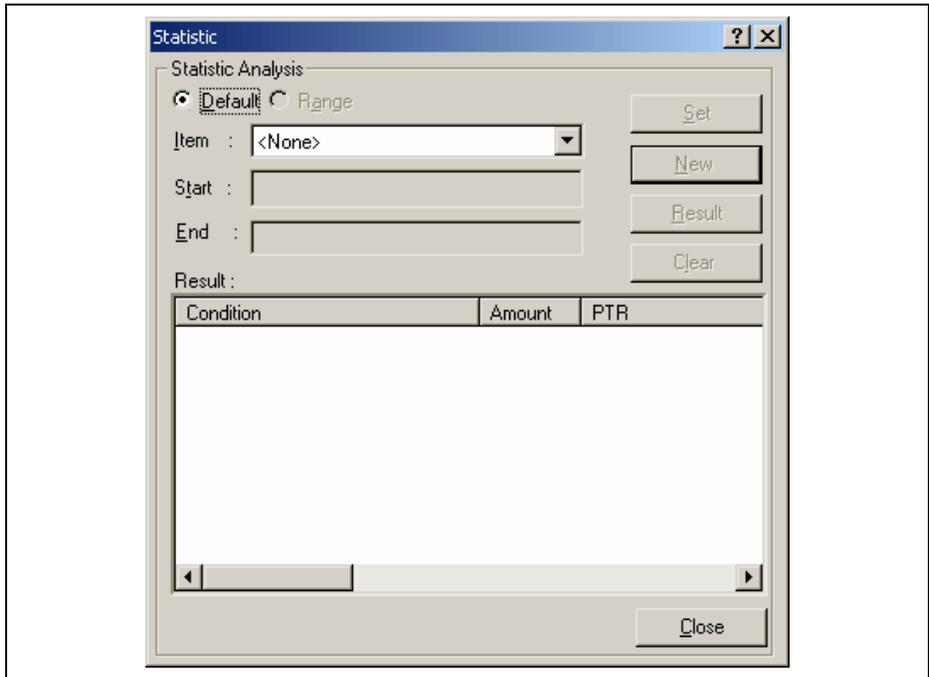


Figure 5.69 [Statistic] Dialog Box

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

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

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

[Item]: Sets the item for analysis.

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

High-performance Embedded Workshop User's Manual

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

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

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

5.14.12 Extracting Function Calls from the Acquired Trace Information

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

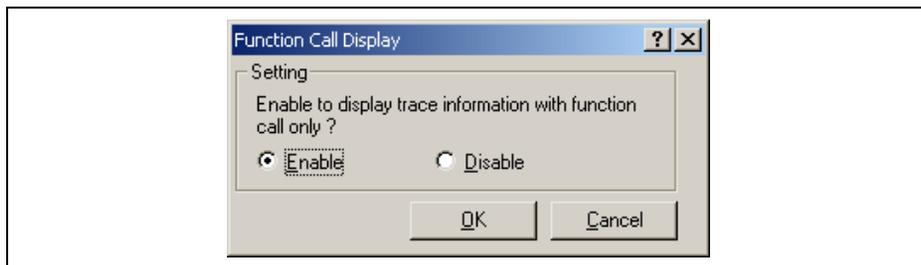


Figure 5.70 [Function Call Display] Dialog Box

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

 [Enable]: Extracts function calls.

 [Disable]: Does not extract function calls.

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

5.15 Viewing the Function Call History

The [Stack Trace] window shows the function call history.

5.15.1 Opening the [Stack Trace] Window

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

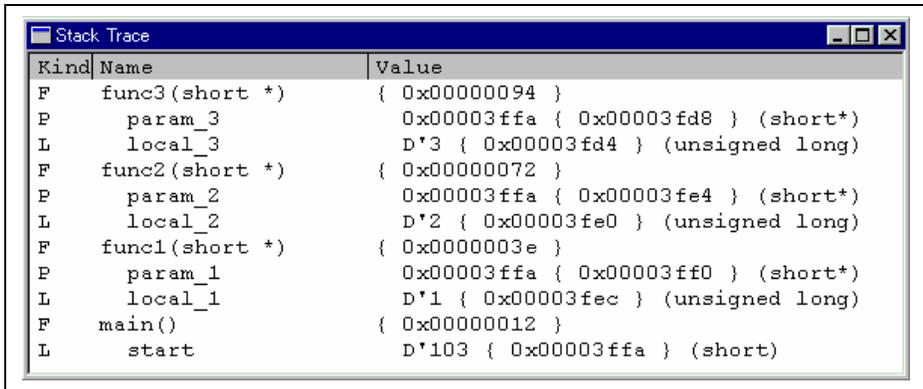


Figure 5.71 [Stack Trace] Window

The following items are displayed.

[Kind]: Indicates the type of the symbol.
 F: Function
 P: Function parameter
 L: Local variable

[Name]: Indicates the symbol name.

[Value]: Indicates the value, address, and type of the symbol.

5.15.2 Viewing the Source Program

Select a function and choose [Go to Source] from the popup menu to display, in the [Editor] window, the source program corresponding to the selected function.

5.15.3 Specifying the View

Choose [View Setting...] from the popup menu to open the [Stack Trace Setting] dialog box, which allows the user to specify the [Stack Trace] window settings.

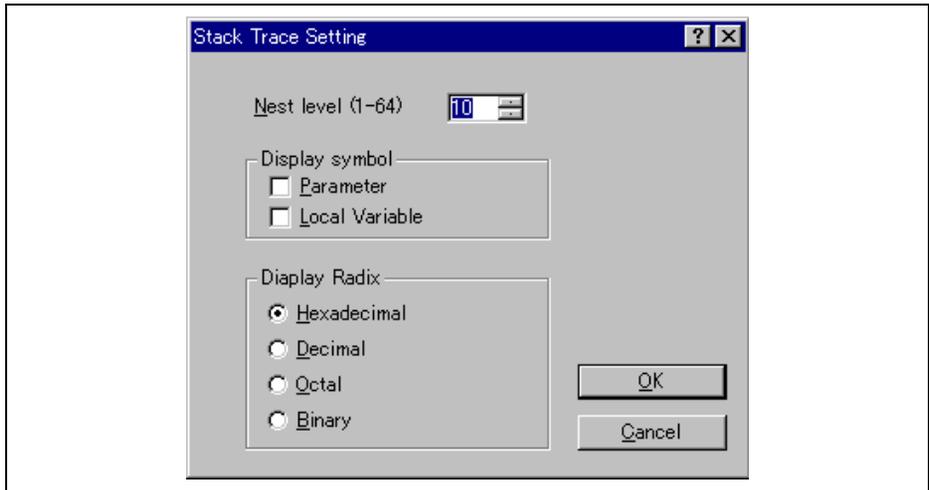


Figure 5.72 [Stack Trace Setting] Dialog Box

[Nest level]: Specifies the level of function call nesting to be displayed in the [Stack Trace] window.

[Display symbol]: Specifies the symbol types to be displayed in addition to functions.

[Display Radix]: Specifies the radix for displays in the [Stack Trace] window.

5.16 Displaying Memory Contents as an Image

The memory contents can be displayed as an image in the [Image] window.

5.16.1 Opening the [Image] Window

Choose [View -> Graphic -> Image...] or click the [Image] toolbar button () to open the [Image Properties] dialog box shown in figure 5.73.

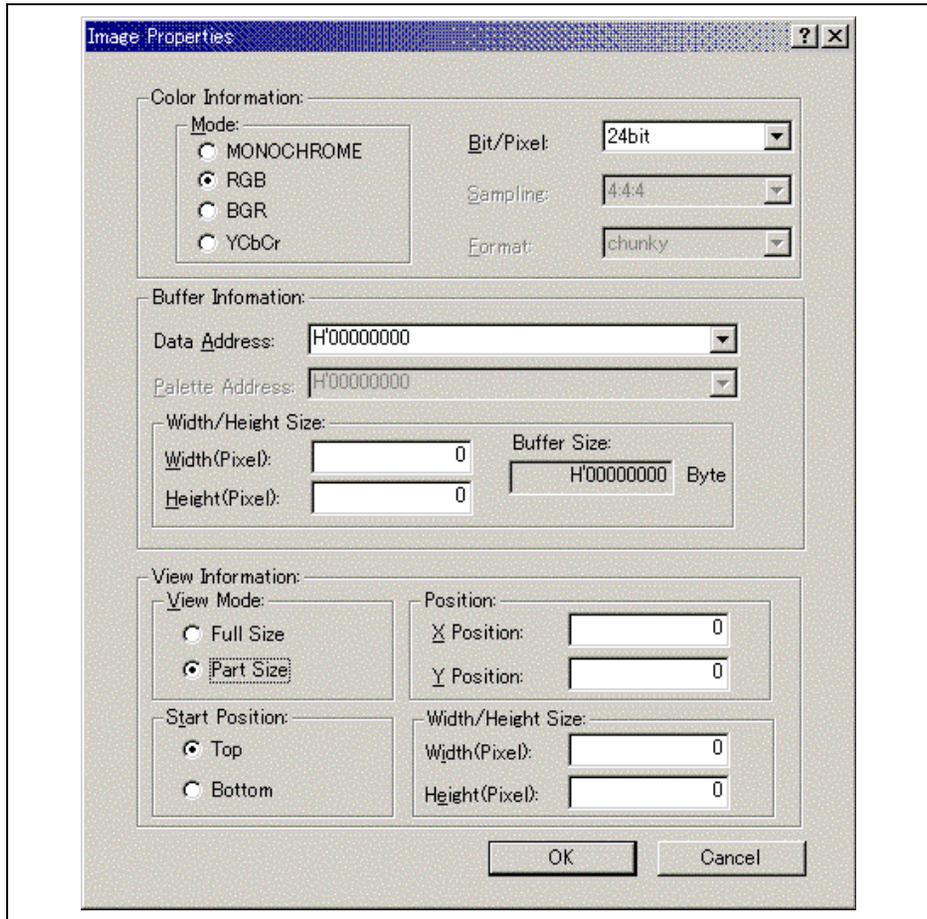


Figure 5.73 [Image Properties] Dialog Box

The [Image Properties] dialog box is used to specify the display method of the [Image] window.

The following items are to be specified:

- [Color Information]: Specifies the color information of the image to be displayed.
- [Mode]: Specifies the format.
- [MONOCHROME]: Black and white.
 - [RGB]: R (red), G (green), and B (blue)
 - [BGR]: B (blue), G (green), and R (red)
 - [YCbCr]: Y (luminance), Cb (color difference for blue), and Cr (color difference for red)
- [Bit/Pixel]: Specifies bits/pixel (valid when RGB or BGR is selected).
- [Sampling]: Specifies the format of sampling (valid when YCbCr is selected).
- [Format]: Specifies chunky/planar (valid when YCbCr is selected).
- [Buffer Information]: Specifies the area to store data, size, and the address of the palette.
- [Data Address]: Specifies the first address in memory of the area for display as image data (in hexadecimal notation).
- [Palette Address]: Specifies the first address in memory of the color-palette data (in hexadecimal notation; valid when "8Bit" has been selected for RGB or BGR).
- [Width/Height Size]: Specifies the width and height of the image.
- [Width (Pixel)]: Specifies the width of the image (unless a prefix is included, values are treated as decimal numbers).
 - [Height (Pixel)]: Specifies the height of the image (unless a prefix is included, values are treated as decimal numbers).
 - [Buffer Size]: Displays the size of the buffer required for image display as obtained from the width and height (in hexadecimal notation).
- [View Information]: Specifies the location, size, and data start location of the part to be displayed among the entire image.
- [View Mode]: Specifies whether display is on all or part of the screen.
- [Full Size]: The image is displayed on the whole screen.
 - [Part Size]: The image is displayed on part of the screen.

[Start Position]:

- [Top]: Display of data starts at the upper-left position.
- [Bottom]: Display of data starts at the lower-left position.

[Position]: Specifies the position on the screen from which image display is to start (valid when [Part Size] is selected).

- [X Position]: Specifies the X-coordinate of the start position (unless a prefix is included, values are treated as decimal numbers).
- [Y Position]: Specifies the Y-coordinate of the start location (unless a prefix is included, values are treated as decimal numbers).

[Width/Height Size]: Specifies the height and width of an image to be displayed on part of the screen.

- [Width (Pixel)]: Specifies the width of the display (unless a prefix is included, values are treated as decimal numbers).
- [Height (Pixel)]: Specifies the height of the display (unless a prefix is included, values are treated as decimal numbers).

After the settings have been made in the [Image Properties] dialog box, clicking the [OK] button opens the [Image] window.

Even after the [Image] window is displayed, the display contents can be modified by opening this dialog box by choosing [Properties...] from the popup menu.



Figure 5.74 [Image] Window

The memory content is displayed as an image.

5.16.2 Automatically Updating the Window Contents

Checking [Auto Refresh] in the popup menu will allow the window contents to be automatically updated when user program execution stops.

5.16.3 Updating the Window Contents

Selecting [Refresh Now] from the popup menu immediately updates the window contents.

5.16.4 Displaying the Pixel Information

Double-clicking within the window displays information on the pixel on which the mouse pointer is located in the [Pixel Information] dialog box.

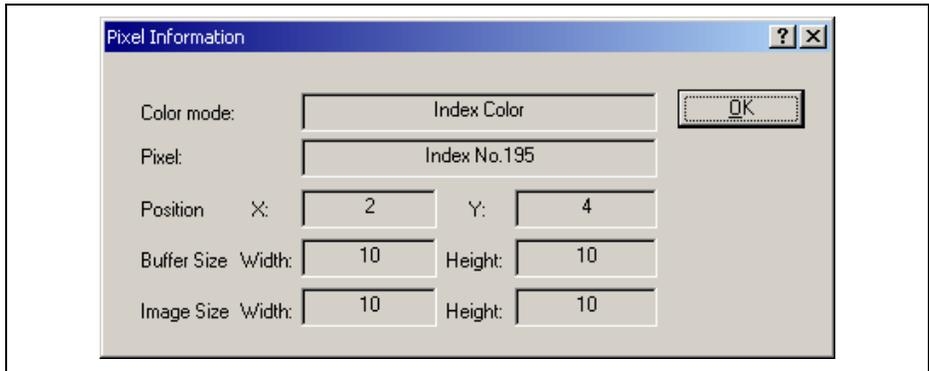


Figure 5.75 [Pixel Information] Dialog Box

This dialog box displays information on the pixel under the cursor.

[Color Mode]: Displays the format of the image.

[Pixel]: Displays color information on the pixel under the cursor (in decimal notation).

[Position]: Displays the cursor location as X and Y coordinates (in decimal notation).

[X]: X coordinate of the cursor.

[Y]: Y coordinate of the cursor.

[Buffer Size]: Displays the buffer size in decimal notation.

[Width]: Width of the buffer.

[Height]: Height of the buffer.

[Image Size]: Displays the width and height of the display in decimal notation.

[Width]: Width of the display.

[Height]: Height of the display.

5.17 Displaying Memory Contents as Waveforms

Memory contents can be displayed as waveforms in the [Waveform View] window.

5.17.1 Opening the [Waveform View] Window

Choose [View -> Graphic -> Waveform...] or click the [Waveform] toolbar button  to open the [Waveform Properties] dialog box shown in figure 5.76.

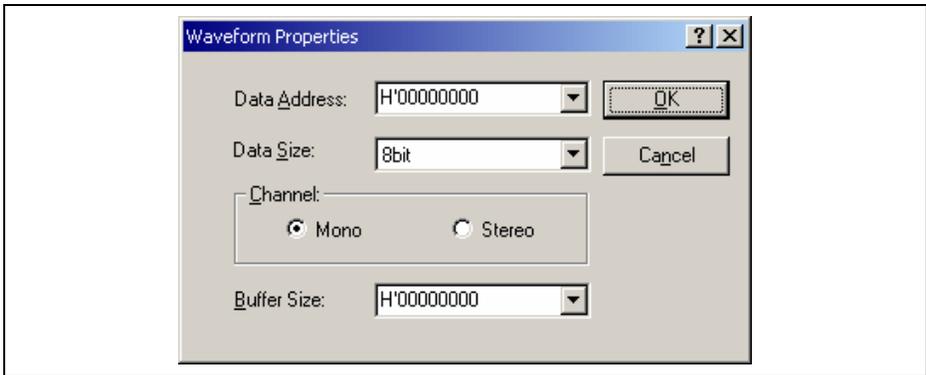


Figure 5.76 [Waveform Properties] Dialog Box

Specifies the waveform format. The following items can be specified.

[Data Address]: Specifies the start address of data in memory (displayed in hexadecimal).

[Data Size]: Selects 8Bit or 16Bit.

[Channel]: Specifies Mono or Stereo.

[Buffer Size]: Specifies the buffer size of data (displayed in hexadecimal).

After the settings have been made in the [Waveform Properties] dialog box, clicking the [OK] button opens the [Waveform View] window.

Even after the [Waveform View] window is displayed, the display contents can be modified by opening this dialog box by choosing [Properties...] from the popup menu.

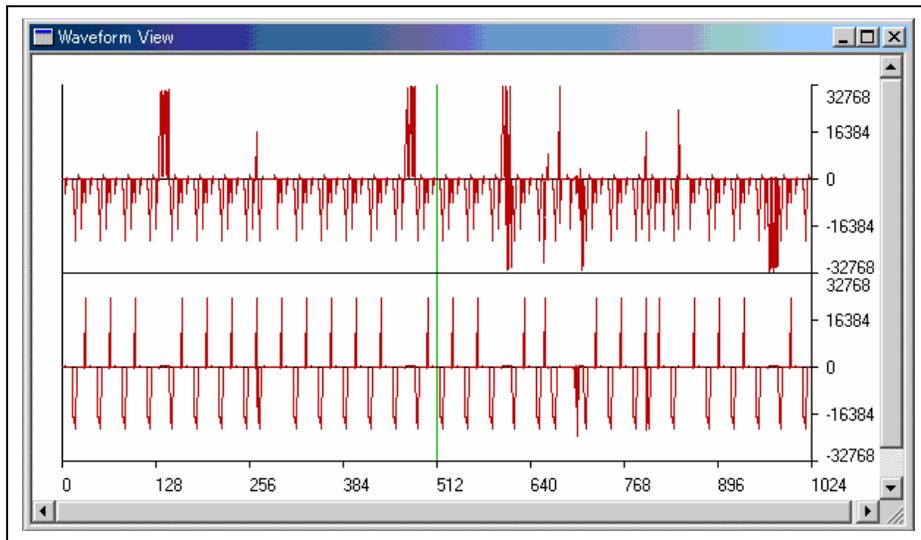


Figure 5.77 [Waveform View] Window

Displays the memory contents as waveforms. The X axis shows the number of sampling data and the Y axis shows the sampling value.

5.17.2 Automatically Updating the Window Contents

Checking [Auto Refresh] in the popup menu will allow the window contents to be automatically updated when user program execution stops.

5.17.3 Updating the Window Contents

Selecting [Refresh Now] from the popup menu immediately updates the window contents.

5.17.4 Zoom-In Display

Selecting [Zoom In] from the popup menu displays the waveforms with the horizontal axis enlarged.

5.17.5 Zoom-Out Display

Selecting [Zoom Out] from the popup menu displays the waveforms with the horizontal axis reduced.

5.17.6 Resetting the Zoom Display

Selecting [Reset Zoom] from the popup menu displays the waveforms in its original size.

5.17.7 Setting the Zoom Magnification

In the [Zoom Magnification] submenu of the popup menu, the zoom magnification can be selected from 2, 4, or 8.

5.17.8 Setting the Horizontal Scale

In the [Scale] submenu of the popup menu, the size of the X axis can be selected from 128, 256, or 512 pixels.

5.17.9 Non-Display of Cursor

Selecting [Clear Cursor] from the popup menu hides the cursor display.

5.17.10 Displaying the Sampling Information

Selecting [Sample Information...] from the popup menu displays the [Sample Information] dialog box.

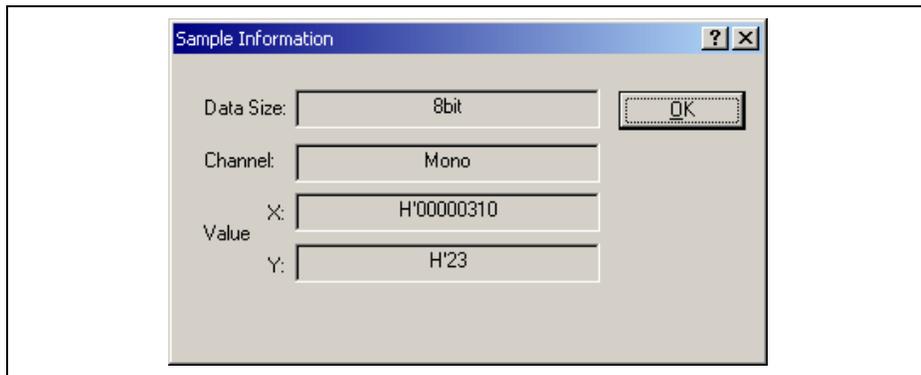


Figure 5.78 [Sample Information] Dialog Box

Displays the sampling information of the cursor location in the [Waveform View] window. The following information is displayed.

[Data Size]: Displays 8bit or 16bit.

[Channel]: Displays the data channel.

[Value]: [X] Displays the X axis of cursor location.

[Y] Displays the Y axis of cursor location (displays the Y axis for both the upper and lower plots when Stereo is selected).

5.18 Analyzing Performance

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

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

5.18.1 Opening the [Performance Analysis] Window

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



Figure 5.79 [Select Performance Analysis Type] Window

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

No	Name	Condition	Rate	RUN-TIME	MAX-MIN-TIME	Count
1	PA1	Range H'00001038 H'00001108	9%	00h 00min 00s 055ms 764us 160ns		1234
2	PA2	PtoP H'00002068 H'000020D4	25%	00h 00min 00s 147ms 682us 880ns	00h 00min...	1234
3	PA3	RtoR H'00001038 H'00001108	65%	00h 00min 00s 381ms 580us 480ns		2468
4		+ H'00002068 H'00002138				
5						
6						
7						
8						

Figure 5.80 [Performance Analysis] Window

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

5.18.2 Setting Conditions for Measurement

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

5.18.3 Starting Performance Data Acquisition

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

5.18.4 Deleting a Measurement Condition

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

5.18.5 Deleting All Measurement Conditions

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

5.19 Viewing the Profile Information

The profile function enables function-by-function measurement of the performance of the application program in execution. This makes it possible to identify parts of an application program that degrade its performance and the reasons for such degradation.

The HEW displays the results of measurement in three windows, according to the method and purpose of viewing the profile data.

5.19.1 Stack Information Files

The profile function allows the HEW to read the stack information files (extension: “.SNI”) which are output by the optimizing linker (ver. 7.0 or later). Each of these files contains information related to the calling of static functions in the corresponding source file. Reading the stack information file makes it possible for the HEW to display this information to do with the calling of functions without executing the user application (i.e. before measuring the profile data). However, this feature is not available when [Setting->Only Executed Functions] is checked in the pop-up menu of the [Profile] window.

When the HEW does not read any stack information files, the data about the functions executed during measurement will be displayed by the profile function.

To make the linker create a stack information file, choose [Options -> Hitachi SuperH Risc engine Standard Toolchain...], and select [Other] from the [Category] list box and check the [Stack information output] box in the [Link/Library] sheet of the [Standard Toolchain] dialog box.

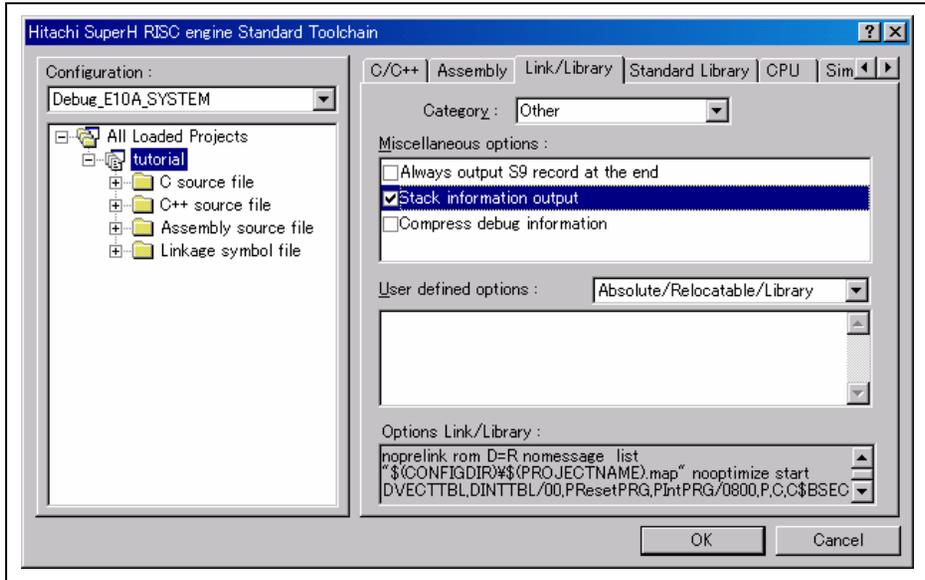


Figure 5.81 [Standard Toolchain] Dialog Box (1)

5.19.2 Profile Information Files

To create a profile information file, choose the [Output Profile Information Files...] menu option from the pop-up menu of the [Profile] window and specify the file name, after measuring a profile data of the application program.

This file contains information on the number of times functions are called and global variables are accessed. The optimizing linker (ver. 7.0 or later) is capable of reading the profile information file and optimizing the allocation of functions and variables in correspondence with the status of the actual operation of the program.

To input the profiler information file to the linker, choose [Optimize] from the [Category] list box and check the [Include Profile] box in the [Link/Library] sheet of the [Standard Toolchain] dialog box, and specify the name of the profile information file.

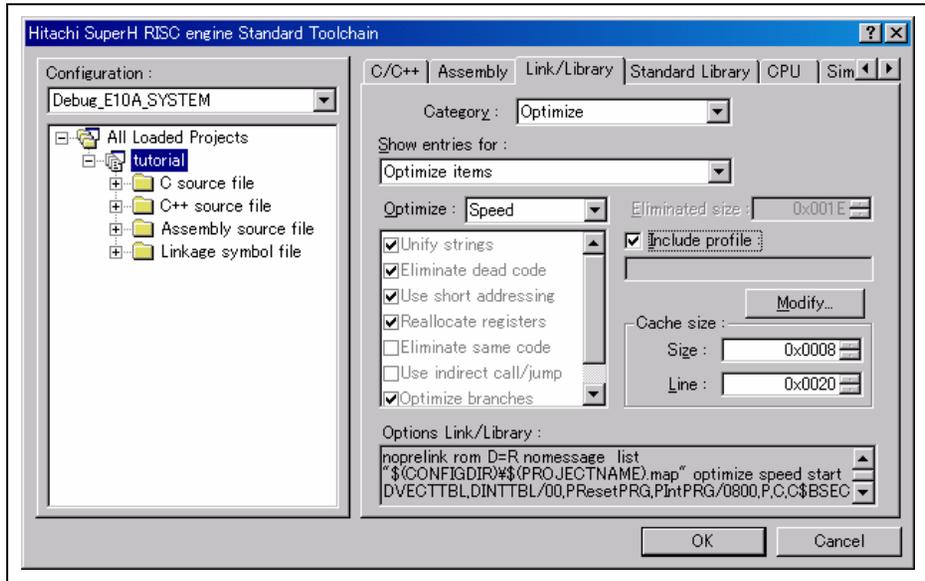


Figure 5.82 [Standard Toolchain] Dialog Box (2)

To enable the settings in the [Include Profile] box, specify the [Optimize] list box as some setting other than [None].

5.19.3 Loading Stack Information Files

You can select whether or not to read the stack information file in a message box for confirmation that is displayed when a load module is loaded. Clicking the [OK] button of the message box loads the stack information file. The message box for confirmation will be displayed when:

- There are stack information files (extension: “*.SNI”).
- The [Load Stack Information Files (SNI files)] check box is checked in the [Confirmation] sheet of the [Options] dialog box (figure 5.83) that can be opened by choosing [Tools ->Options] from the main menu.

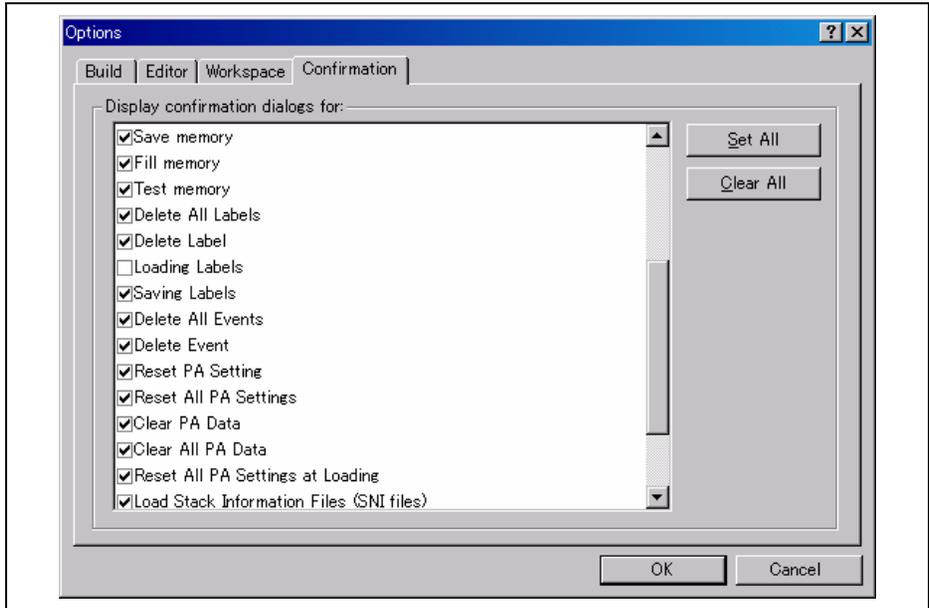


Figure 5.83 [Options] Dialog Box

5.19.4 Enabling the Profile

Choose [View->Performance->Profile] to open the [Profile] window.

Choose [Enable Profiler] from the pop-up menu of the [Profile] window. The item on the menu will be checked.

5.19.5 Specifying Measuring Mode

You can specify whether to trace functions calls while profile data is acquired. When function calls are traced, the relations of function calls during user program execution are displayed as a tree diagram. When not traced, the relations of function calls cannot be displayed, but the time for acquiring profile data can be reduced.

To stop tracing function calls, choose [Disable Tree (Not traces function call)] from the pop-up menu in the [Profile] window (a check mark is shown to the left of the menu item).

When acquiring profile data of the program in which functions are called in a special way, such as task switching in the OS, stop tracing function calls.

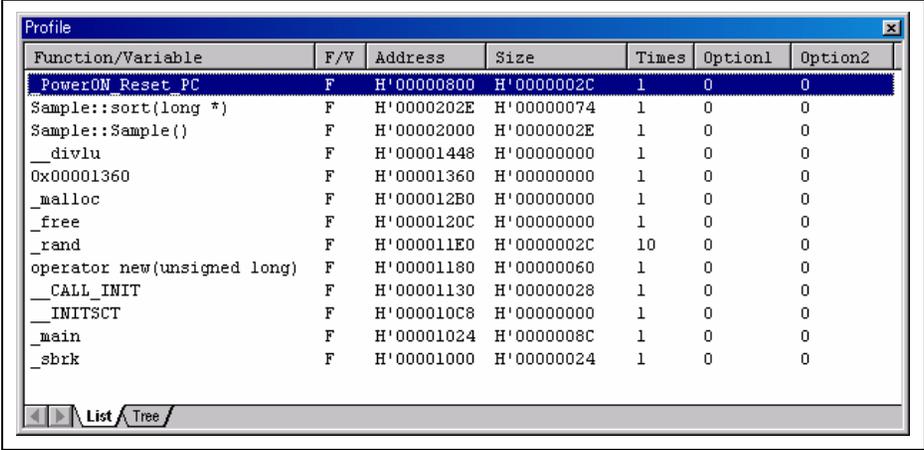
5.19.6 Executing the Program and Checking the Results

After the user program has been executed and execution has been halted, the results of measurement are displayed in the [Profile] window.

The [Profile] window has two sheets; a [List] sheet and a [Tree] sheet.

5.19.7 List Sheet

This sheet lists functions and global variables and displays the profile data for each function and variable.



Function/Variable	F/V	Address	Size	Times	Option1	Option2
PowerON_Reset_PC	F	H'00000800	H'0000002C	1	0	0
Sample::sort(long *)	F	H'0000202E	H'00000074	1	0	0
Sample::Sample()	F	H'00002000	H'0000002E	1	0	0
__divlu	F	H'00001448	H'00000000	1	0	0
0x00001360	F	H'00001360	H'00000000	1	0	0
__malloc	F	H'000012B0	H'00000000	1	0	0
__free	F	H'0000120C	H'00000000	1	0	0
__rand	F	H'000011E0	H'0000002C	10	0	0
operator new(unsigned long)	F	H'00001180	H'00000060	1	0	0
__CALL_INIT	F	H'00001130	H'00000028	1	0	0
__INIT SCT	F	H'000010C8	H'00000000	1	0	0
__main	F	H'00001024	H'0000008C	1	0	0
__sbrk	F	H'00001000	H'00000024	1	0	0

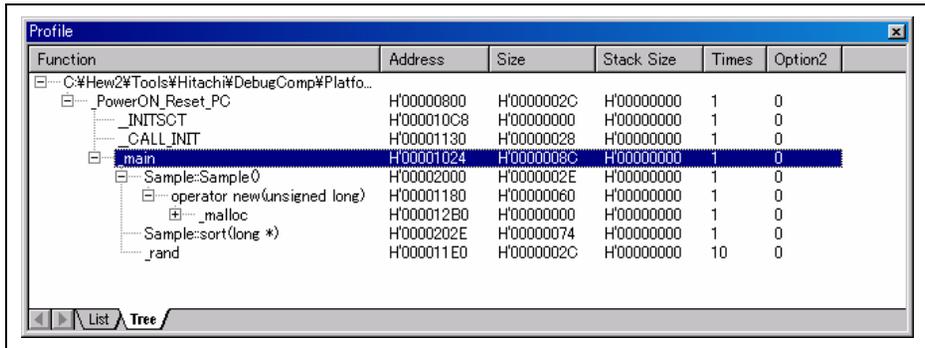
Figure 5.84 [List] Sheet

Clicking the column header sorts the items in an alphabetical or ascending order. Double-clicking the [Function/Variable] or [Address] column displays the source program corresponding to the address in the line.

Right-clicking on the mouse within the window displays a pop-up menu. For details on this pop-up menu, refer to section 5.19.8, Tree Sheet, in the Debugger Part.

5.19.8 [Tree] Sheet

This sheet displays the relation of function calls along with the profile data that are values when the function is called. This sheet is available when [Disable Tree (Not traces function call)] is not selected from the pop-up menu in the [Profile] window.



Function	Address	Size	Stack Size	Times	Option2
[-] C:\Hew2#Tools#Hitachi#DebugComp#Platfo...					
[-] _PowerON_Reset_PC	H'00000800	H'0000002C	H'00000000	1	0
[-] _INITSCT	H'000010C8	H'00000000	H'00000000	1	0
[-] _CALL_INIT	H'00001130	H'00000028	H'00000000	1	0
[-] main	H'00001024	H'0000008C	H'00000000	1	0
[-] Sample:Sample()	H'00002000	H'0000002E	H'00000000	1	0
[-] operator new(unsigned long)	H'00001180	H'00000060	H'00000000	1	0
[-] _malloc	H'000012B0	H'00000000	H'00000000	1	0
[-] Sample:sort(long *)	H'0000202E	H'00000074	H'00000000	1	0
[-] _rand	H'000011E0	H'0000002C	H'00000000	10	0

Figure 5.85 [Tree] Sheet

Double-clicking a function in the [Function] column expands or reduces the tree structure display. The expansion or reduction is also provided by the “+” or “-“ key. Double-clicking the [Address] column displays the source program corresponding to the specific address.

Right-clicking on the mouse within the window displays a pop-up menu. Supported menu options are described in the following:

- View Source
Displays the source program or disassembled memory contents for the address in the selected line.
- View Profile-Chart
Displays the [Profile-Chart] window focused on the function in the specified line.
- Enable Profiler
Toggles acquisition profile data. When profile data acquisition is active, a check mark is shown to the left of the menu text.
- Not trace the function call
Stops tracing function calls while profile data is acquired. This menu is used when acquiring profile data of the program in which functions are called in a special way, such as task switching in the OS.

To display the relation of function calls in the [Tree]sheet of the [Profile] window, acquire profile data without selecting this menu. In addition, do not select this menu when optimizing the program by the optimizing linkage editor using the acquired profile information file.

- Find...
Displays the [Find Text] dialog box to find a character string in the [Function] column. Search is started by inputting a character string to be found in the edit box and clicking [Find Next] or pressing the Enter key.
- Find Data...
Displays the [Find Data] dialog box.

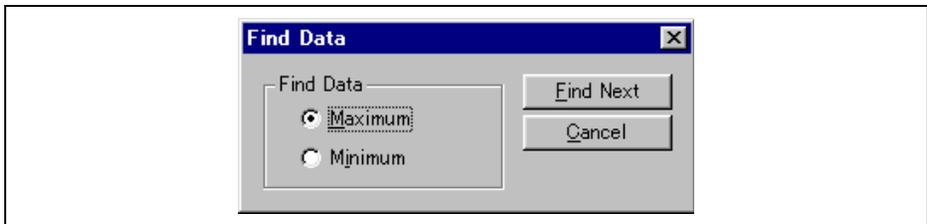


Figure 5.86 [Find Data] Dialog Box

By selecting the column to be searched in the [Column] combo box and the search type in the [Find Data] group and entering [Find Next] button or Enter key, search is started. If the [Find Next] button or the Enter key is input repeatedly, the second larger data (the second smaller data when the Minimum is specified) is searched for.

- Clear Data
Clears the number of times functions are called and profile data. Data in the [List] sheet of the [Profile] window and the data in the [Profile-Chart] window are also cleared.
- Output Profile Information Files...
Displays the [Save Profile Information Files] dialog box. Profiling results are saved in a profile information file (.pro extension). The optimizing linkage editor optimizes user programs according to the profile information in this file. For details of the optimization using the profile information, refer to the manual of the optimizing linkage editor.

Note: If profile information has been acquired by choosing the [Not trace the function call] menu, the program cannot be optimized by the optimizing linkage editor.

- Output Text File...
Displays the [Save Text of Profile Data] dialog box. Displayed contents are saved in a text file.

- Setting
 - This menu has the following submenus (the menus available only in the [List] sheet are also included).
 - Show Functions/Variables
 - Displays both functions and global variables in the [Function/Variable] column.
 - Show Functions
 - Displays only functions in the [Function/Variable] column.
 - Show Variables
 - Displays only global variables in the [Function/Variable] column.
 - Only Executed Functions
 - Only displays the executed functions. If a stack information file (.sni extension) output from the optimizing linkage editor does not exist in the directory where the load module is located, only the executed functions are displayed even if this check box is not checked.
 - Include Data of Child Functions
 - Sets whether or not to display information for a child function called in the function as profile data.
- Properties...
 - Sets the items to be measured.

5.19.9 [Profile-Chart] Window

The [Profile-Chart] window displays the relation of calls for a specific function. This window displays the specified function in the middle, with the callers of the function on the left and the callees of the function on the right. The numbers of times the function calls the called functions or is called by the calling functions are also displayed in this window.

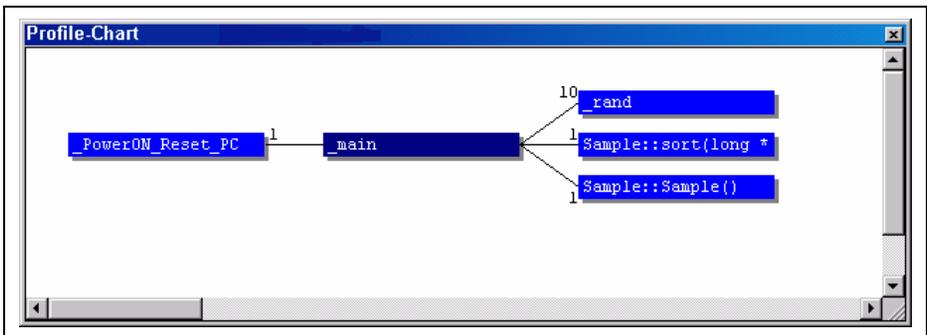


Figure 5.87 [Profile-Chart] Window

5.19.10 Types and Purposes of Displayed Data

The profile function is able to acquire the following information:

Address	You can see the locations in memory to which the functions are allocated. Sorting the list of functions and global variables in order of their addresses allows the user to view the way the items are allocated in the memory space.
Size	Sorting in order of size makes it easy to find small functions that are frequently called. Setting such functions as inline may reduce the overhead of function calls. If you are using a microcomputer which incorporates a cache memory, more of the cache memory will need to be updated when you execute larger functions. This information allows you to check if those functions that may cause cache misses are frequently called.
Stack Size	When there is deep nesting of function calls, pursue the route of the function calls and obtain the total stack size for all of the functions on that route to estimate the amount of stack being used.
Times	Sorting by the number of calls or accesses makes it easy to identify the frequently called functions and frequently accessed global variables.
Profile Data	Measurement of a variety of CPU-specific data is also available as well as items that can be measured with the performance measurement function. For details, refer to the online help.

5.19.11 Creating Profile Information Files

To create a profile information file, choose the [Output Profile Information Files...] menu option from the pop-up menu. The [Save Profile Information Files] dialog box is displayed. Pressing the [Save] button after selecting a file name will write the profile information to the selected file. Pressing the [Save All] button will write the profile information to all of the profile information files.

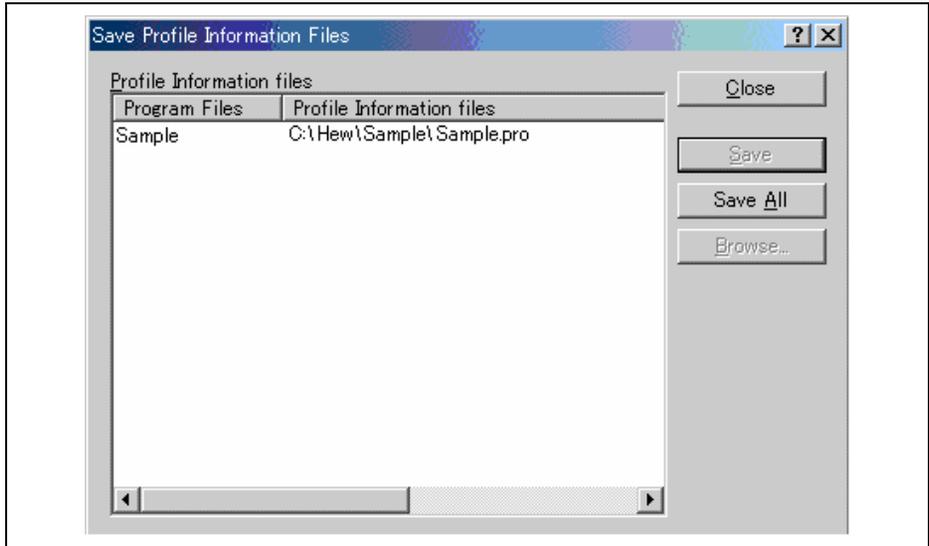


Figure 5.88 [Save Profile Information Files] Dialog Box

5.19.12 Notes

1. Errors

The profile function internally breaks user program execution, collects the measured data, and re-executes the user program.

Since the function also counts when the measured item is generated at break or re-execution, an error will be included in the measured profile value.

The measured value of this function should be the target.

2. Functions that cannot be used while the profile function is being used
 - (a) Performance measurement function

The profile function is implemented by using the performance measurement function described in section 2.3.6, Performance Measurement Function, in the Debugger Part. This function cannot be used when the profile function is enabled.
 - (b) Step function

When the profile function is enabled, do not use the step function. The profile data cannot be measured correctly.
 - (c) Memory access during user program execution

When the profile function is enabled, memory access is disabled during user program execution.
 - (d) When the profile function is used, a break occurs if a branch instruction is generated. Accordingly, the realtime emulation will not be performed. In addition, since the emulator firmware is controlled on generation of a break, the executed result of the branch instruction may be displayed on the [Trace] window when the execution is returned to the user program from the emulator firmware. In this case, ****EML**** is displayed.
 - (e) When the profile function is enabled, do not use the function of Break Condition 3.
3. Others
 - (a) When the profile function is used, the contents that have been set in the performance measurement function or data that has been measured will be deleted.
 - (b) Since the profile function is implemented with the internal break, it takes a long time to start and end the user program execution. The user program execution times under the following environment are shown below:

Environment:

 - Host computer: 930 MHz (Pentium® III)
 - Memory: 127 Mbytes
 - OS: Windows® Me
 - Execution program: 10,000 nested calls
 - (i) When the profile function is not used: 1 second or lower
 - (ii) When the profile function is used in the setting without including a child function: 96 seconds
 - (iii) When the profile function is used in the setting including a child function: 316 seconds

Section 6 Tutorial

6.1 Introduction

This section describes the main functions of the emulator 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 `main` function generates random data to be sorted.
- The `sort` function sorts the generated random data in ascending order.
- The `change` function then sorts the data in descending order.

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

- Notes:
1. Operation of `Tutorial.abs` is big endian. For little-endian operation, `Tutorial.abs` must be recompiled. After recompilation, the addresses may differ from those given in this section.
 2. This section describes general usage examples for the emulator. For the specifications of particular products, refer to the additional document or the online help.
 3. The operation address of `Tutorial.abs` attached to each product differs depending on the product. Replace the address used in this section with upper 16 bits of the actually loaded address.
Example: Although the PC address is `H'0000006c` in the manual, enter `H'0C00xxxx` when the loaded address of `Tutorial.abs` is `H'0C00006c` (upper bit `H'0000` is changed to `H'0C00`).

6.2 Running the HEW

To run the HEW, refer to section 3.5, System Check.

6.3 Setting up the Emulator

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

- **AUD clock**
A clock used in acquiring AUD traces.
If its frequency is set too low, complete data may not be acquired during realtime tracing.
Set the frequency not to exceed the upper limit for the MCU's AUD clock.
The AUD clock is only needed for using emulators that have an AUD trace function.
- **JTAG clock (TCK)**
A communication clock used except for acquiring AUD trace.
If its frequency is set too low, the speed of downloading will be lowered.
Set the frequency not to exceed the upper limit for the MCU's guaranteed TCK range.

For details of the limitations on both clocks, refer to section 2.2.4, Notes on Using the JTAG Clock (TCK) and AUD Clock (AUDCK), in the Specific Guide for the SHxxxx E10A Emulator. The following is a description of the procedure used to set the clocks.

6.4 Setting the [Configuration] Dialog Box

- Select [Emulator] then [Systems...] from the [Options] menu to set a communication clock. The [Configuration] dialog box is displayed.

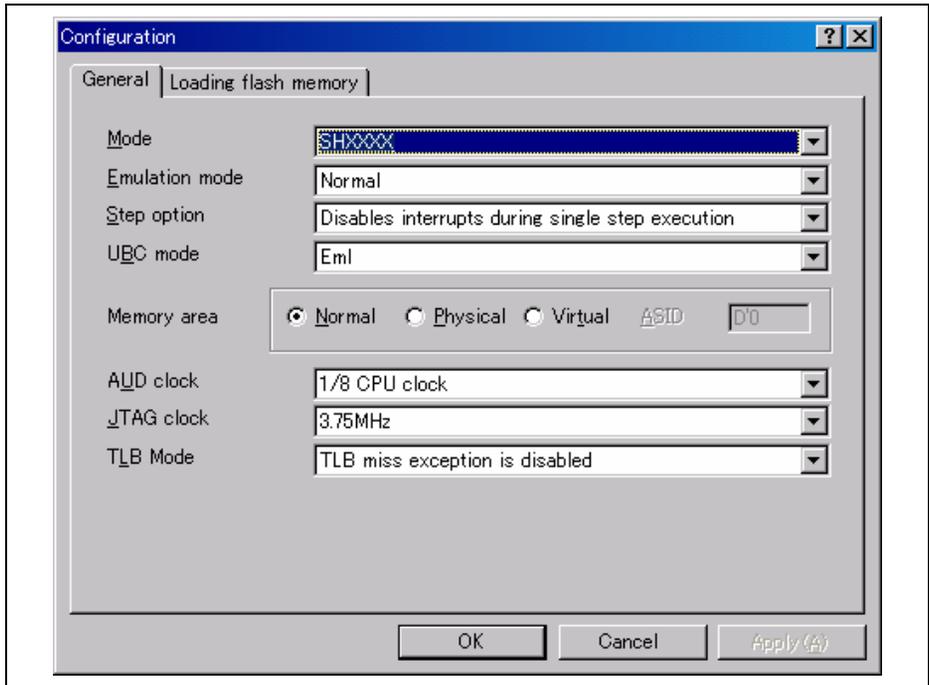


Figure 6.1 [Configuration] Dialog Box

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

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

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

6.5 Checking the Operation of the Target Memory for Downloading

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

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

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

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

- Select [Memory...] from the [CPU] submenu of the [View] menu, enter **H'00000000** and **H'0000FFFF** in the [Begin] and [End] edit boxes, respectively, and set the format in the [Format] combo box to **Byte**.

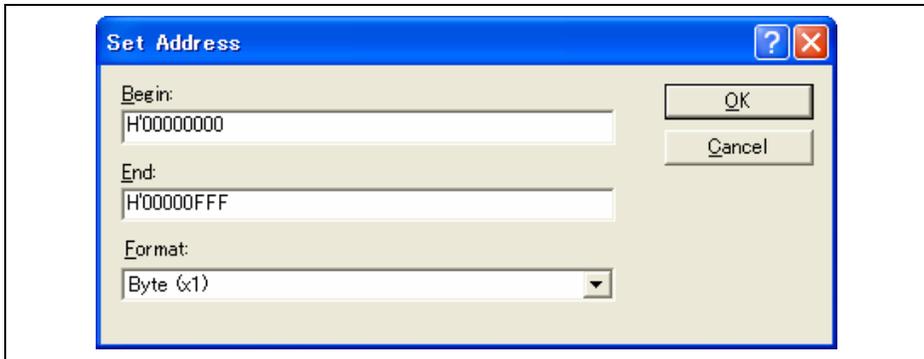


Figure 6.2 [Set Address] Dialog Box

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

Address	Data	Value
0x00000000	00 00 08 00
0x00000004	00 01 00 00
0x00000008	00 00 08 2C	...,
0x0000000C	00 01 00 00
0x00000010	00 00 08 44	...D
0x00000014	00 00 08 48	...H
0x00000018	06 8F 5E D2	..^.

Figure 6.3 [Memory] Window

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

6.6 Downloading the Tutorial Program

6.6.1 Downloading the Tutorial Program

Download the object program to be debugged.

In this emulator, it is enabled to download the program and set the PC breakpoint in the internal flash memory area. For the method to set the PC breakpoint, refer to section 6.17.1, PC Break Function.

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

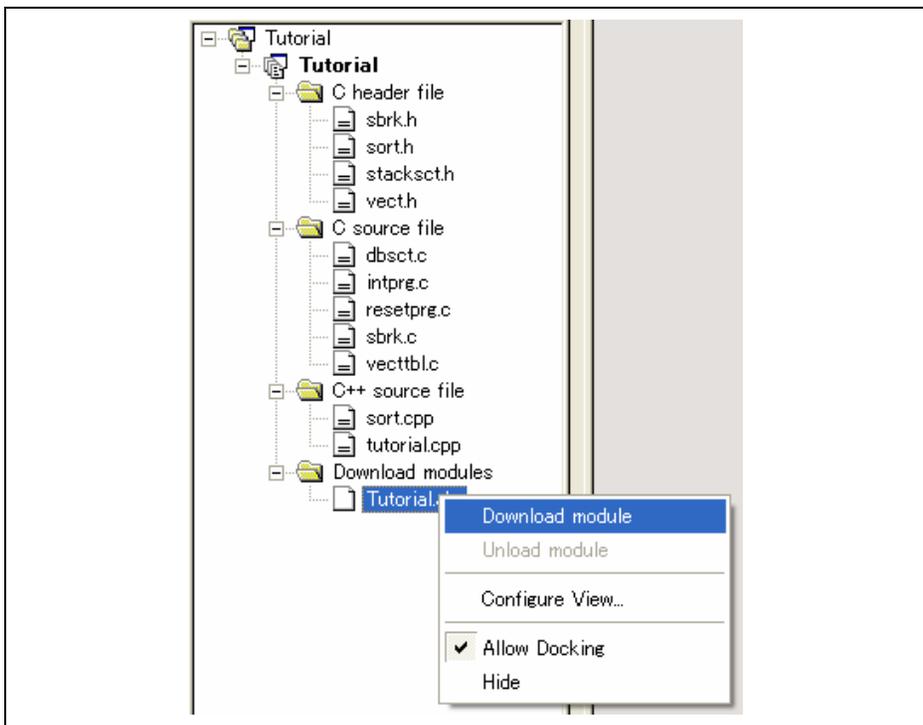


Figure 6.4 Downloading the Tutorial Program

6.6.2 Displaying the Source Program

The HEW allows the user to debug a user program at the source level.

- Double-click [tutorial.cpp] under [C++ source file].

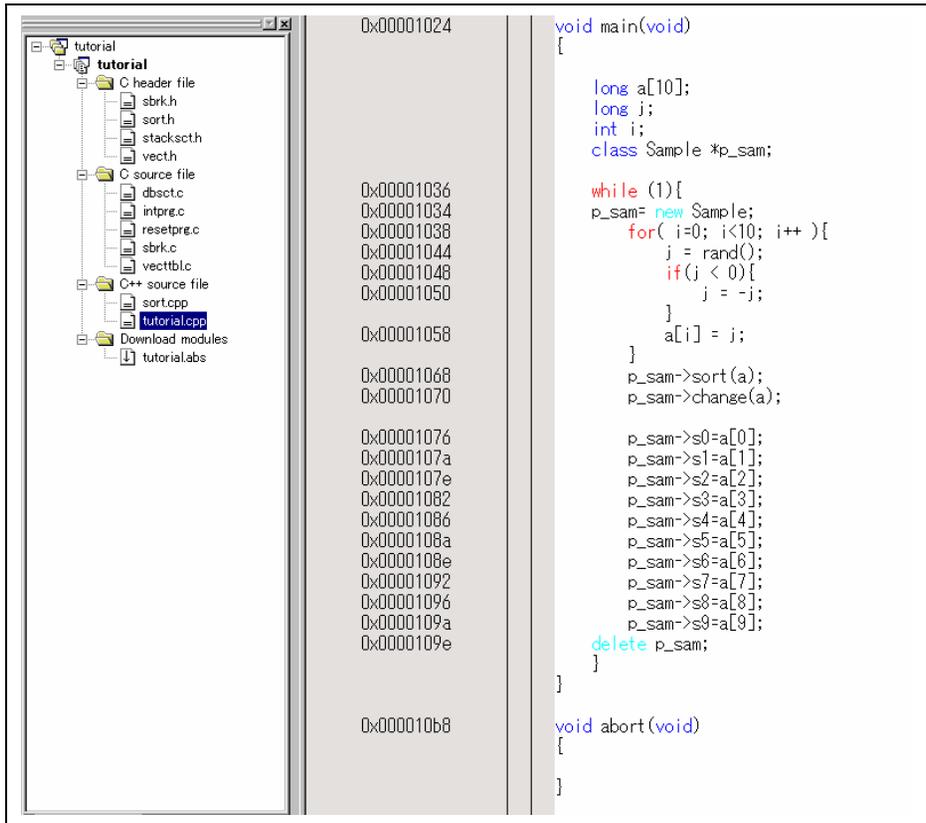


Figure 6.5 [Editor] Window (Displaying the Source Program)

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

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

6.7 Setting a PC Breakpoint

A PC breakpoint is a simple debugging function.

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

- Select by double-clicking the [Editor] column on the line containing the `sort` function call.

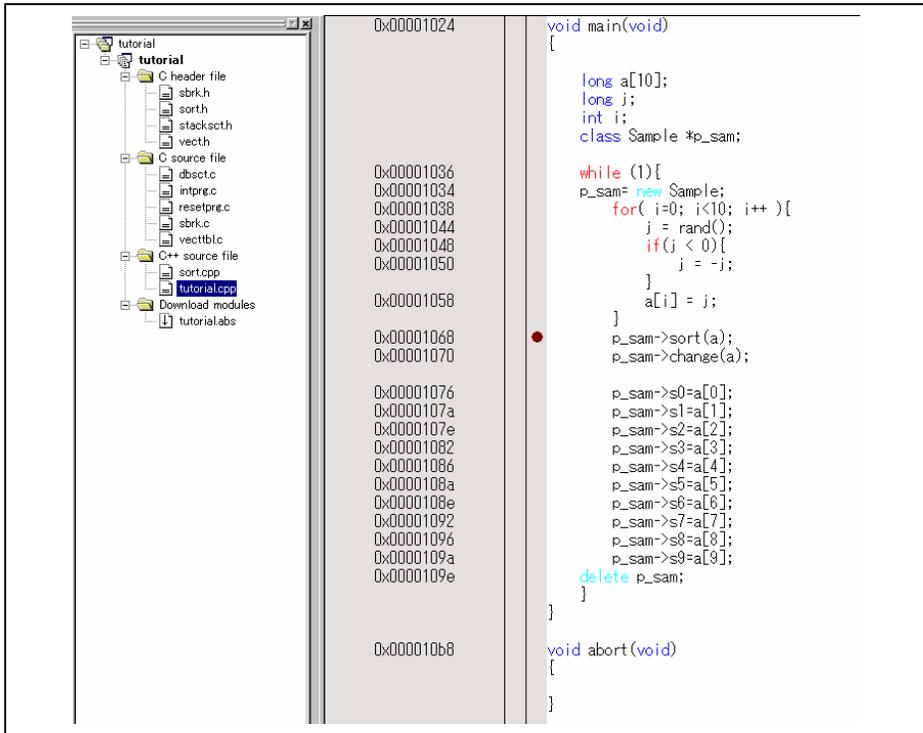


Figure 6.6 [Editor] Window (Setting a PC Breakpoint)

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

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

6.8 Setting Registers

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

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

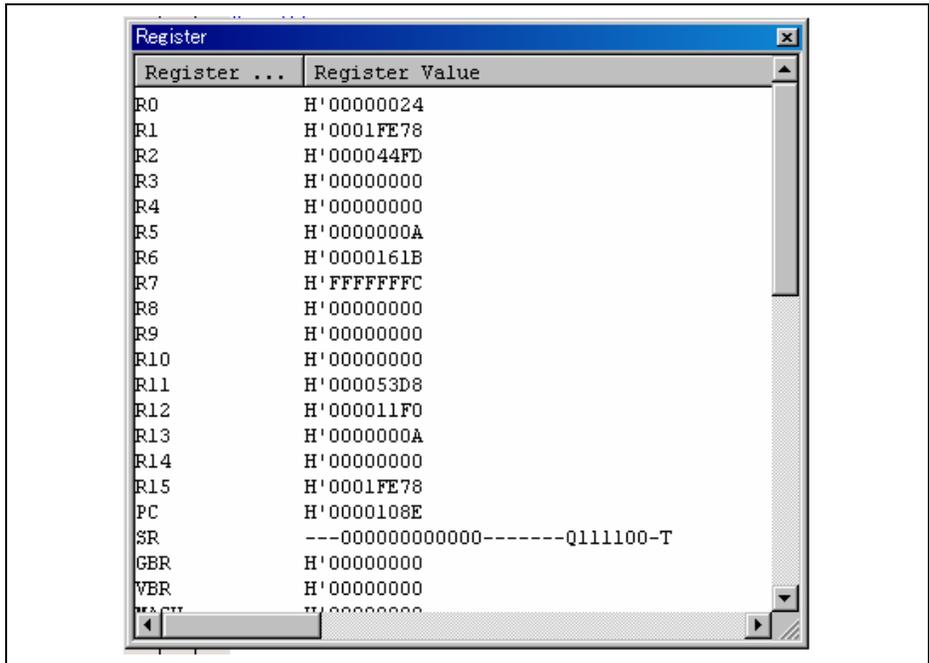


Figure 6.7 [Register] Window

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

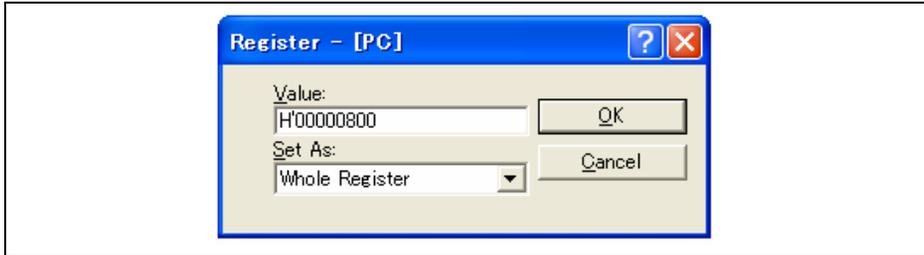


Figure 6.8 [Register] Dialog Box (PC)

- Change the value of the stack pointer (SP) in the same way. Set H'00010000 for the value of the stack pointer in this tutorial program. When using the MCU with flash memory, specify the end address of the internal RAM for the stack pointer (SP). The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.



Figure 6.9 [Register] Dialog Box (R15)

6.9 Executing the Program

Execute the program as described in the following:

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



Figure 6.10 [Go] Button

When the program execution is started, '***RUNNING' is displayed on the status bar, and then the executed PC address is displayed in the products that support the CPU status acquisition function. The program will be executed up to the breakpoint that has been set, and an arrow will be displayed in the [Editor] column to show the position that the program has halted, with the message [BREAKPOINT] in the status bar.

- Notes:
1. When the source file is displayed after a break, a path of the source file may be inquired. The location of the source file is as follows:
<HEW2 installation directory>\Tools\Renesas\DebugComp\Platform\E10A\xxxx
\Tutorial\source.
 2. If program execution is failed, select [Reset CPU] from the [Debug] menu, reset the device, and restart the procedure from figure 6.8.

0x00001024	<code>void main(void)</code>
	<code>{</code>
	<code> long a[10];</code>
	<code> long j;</code>
	<code> int i;</code>
	<code> class Sample *p_sam;</code>
	<code> while (1){</code>
0x00001036	<code> p_sam= new Sample;</code>
0x00001034	<code> for(i=0; i<10; i++){</code>
0x00001038	<code> j = rand();</code>
0x00001044	<code> if(j < 0){</code>
0x00001048	<code> j = -j;</code>
0x00001050	<code> }</code>
	<code> a[i] = j;</code>
0x00001058	<code> }</code>
	<code> p_sam->sort(a);</code>
0x00001068	<code> p_sam->change(a);</code>
0x00001070	
	<code> p_sam->s0=a[0];</code>
0x00001076	<code> p_sam->s1=a[1];</code>
0x0000107a	<code> p_sam->s2=a[2];</code>
0x0000107e	<code> p_sam->s3=a[3];</code>
0x00001082	<code> p_sam->s4=a[4];</code>
0x00001086	<code> p_sam->s5=a[5];</code>
0x0000108a	<code> p_sam->s6=a[6];</code>
0x0000108e	<code> p_sam->s7=a[7];</code>
0x00001092	<code> p_sam->s8=a[8];</code>
0x00001096	<code> p_sam->s9=a[9];</code>
0x0000109a	<code> delete p_sam;</code>
0x0000109e	<code> }</code>
	<code>}</code>
	<code>void abort(void)</code>
0x000010b8	<code>{</code>

Figure 6.11 [Editor] Window (Break State)

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

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

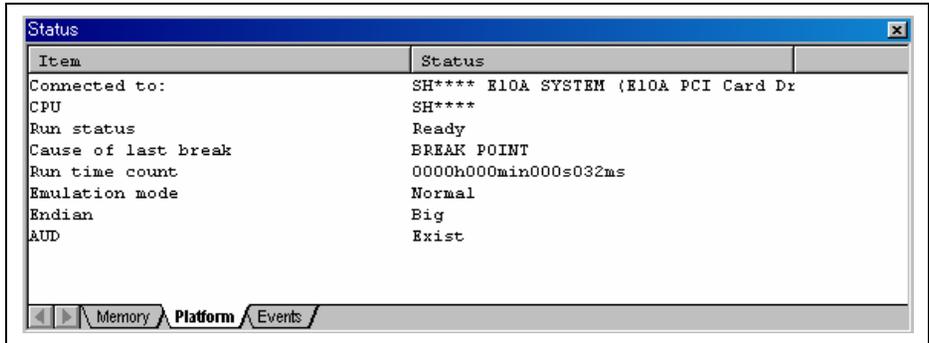


Figure 6.12 [Status] Window

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

6.10 Reviewing Breakpoints

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

- Select [Eventpoints] from the [Code] submenu of the [View] menu. The [Event] window is displayed. Select the [Breakpoint] sheet.

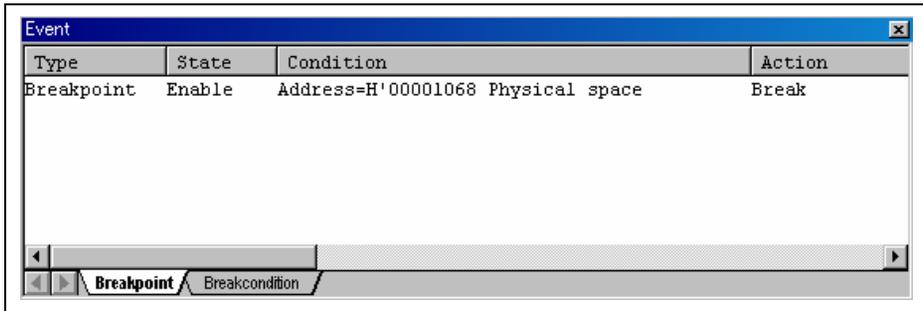


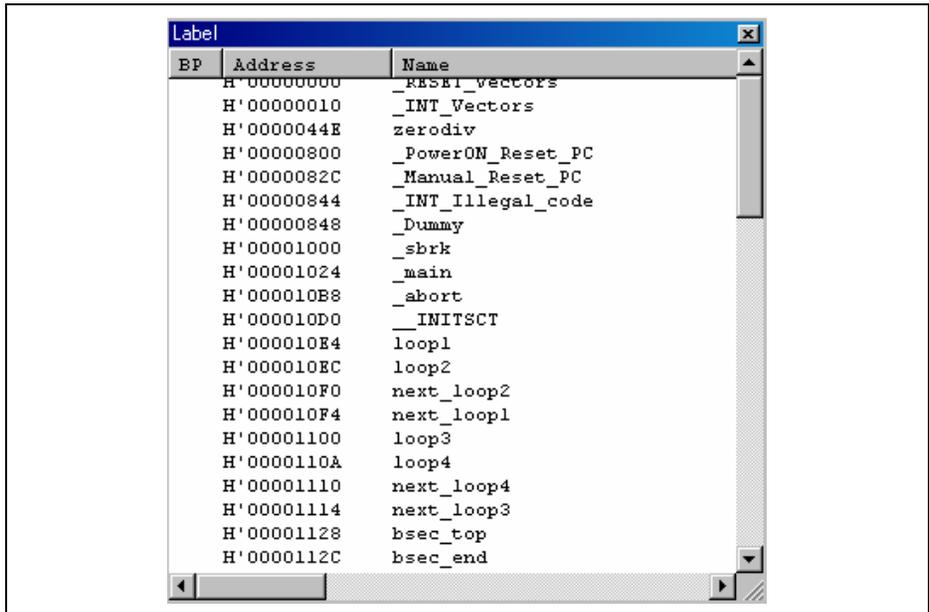
Figure 6.13 [Event] Window

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

6.11 Referring to Symbols

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

Select [Label] from the [Symbol] submenu of the [View] menu. The [Label] window is displayed so that the user can refer to the addresses of symbols in modules.



The screenshot shows a window titled "Label" with a table of symbols. The table has three columns: "BP", "Address", and "Name". The symbols listed include various vectors, reset points, and loop labels.

BP	Address	Name
H'00000000		_RESET_vectors
H'00000010		_INT_Vectors
H'0000044E		zerodiv
H'00000800		_PowerON_Reset_PC
H'0000082C		_Manual_Reset_PC
H'00000844		_INT_Illegal_code
H'00000848		_Dummy
H'00001000		_sbrk
H'00001024		_main
H'000010B8		_abort
H'000010D0		_INITSTCT
H'000010E4		loop1
H'000010EC		loop2
H'000010F0		next_loop2
H'000010F4		next_loop1
H'00001100		loop3
H'0000110A		loop4
H'00001110		next_loop4
H'00001114		next_loop3
H'00001128		bsec_top
H'0000112C		bsec_end

Figure 6.14 [Label] Window

6.12 Viewing Memory

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

- Select [Memory ...] from the [CPU] submenu of the [View] menu, enter `_main` in the [Begin] edit box, enter `+ff` in the [End] edit box, and set `Word` in the [Format] combo box.

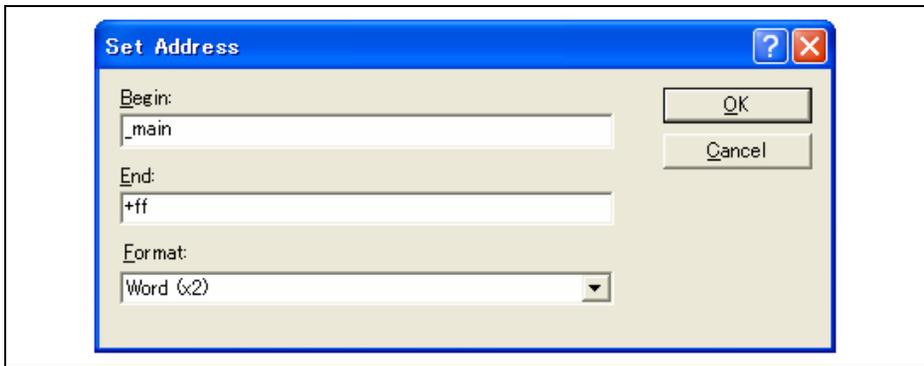
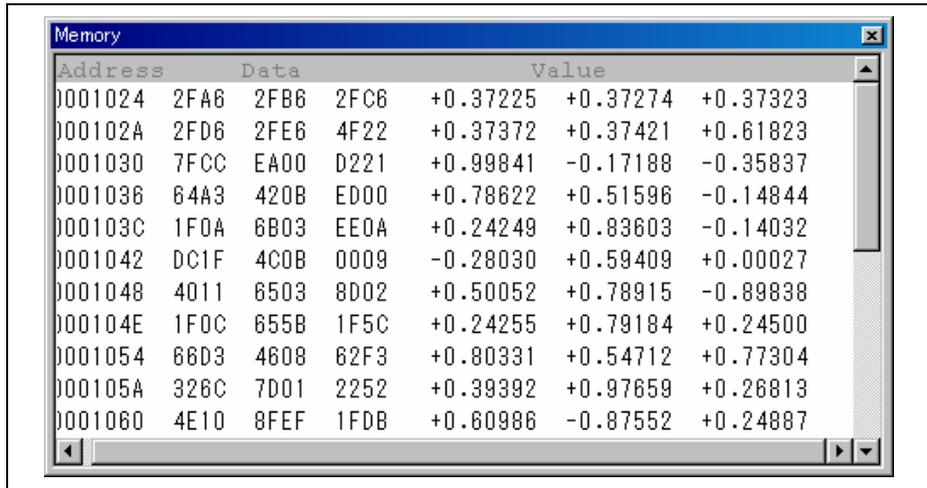


Figure 6.15 [Set Address] Dialog Box

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



Address	Data	Value
0001024	2FA6 2FB6 2FC6	+0.37225 +0.37274 +0.37323
000102A	2FD6 2FE6 4F22	+0.37372 +0.37421 +0.61823
0001030	7FCC EA00 D221	+0.99841 -0.17188 -0.35837
0001036	64A3 420B ED00	+0.78622 +0.51596 -0.14844
000103C	1FOA 6B03 EEOA	+0.24249 +0.83603 -0.14032
0001042	DC1F 4C0B 0009	-0.28030 +0.59409 +0.00027
0001048	4011 6503 8D02	+0.50052 +0.78915 -0.89838
000104E	1F0C 655B 1F5C	+0.24255 +0.79184 +0.24500
0001054	66D3 4608 62F3	+0.80331 +0.54712 +0.77304
000105A	326C 7D01 2252	+0.39392 +0.97659 +0.26813
0001060	4E10 8FEF 1FDB	+0.60986 -0.87552 +0.24887

Figure 6.16 [Memory] Window

6.13 Watching Variables

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

- Click the left of displayed array a in the [Editor] window to position the cursor.
- Select [Instant Watch...] with the right-hand mouse button.

The following dialog box will be displayed.

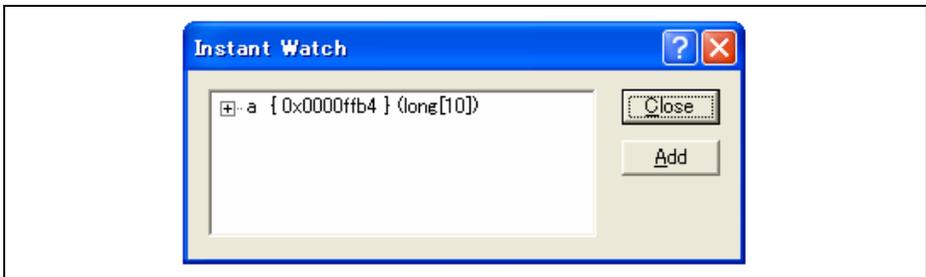


Figure 6.17 [Instant Watch] Dialog Box

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

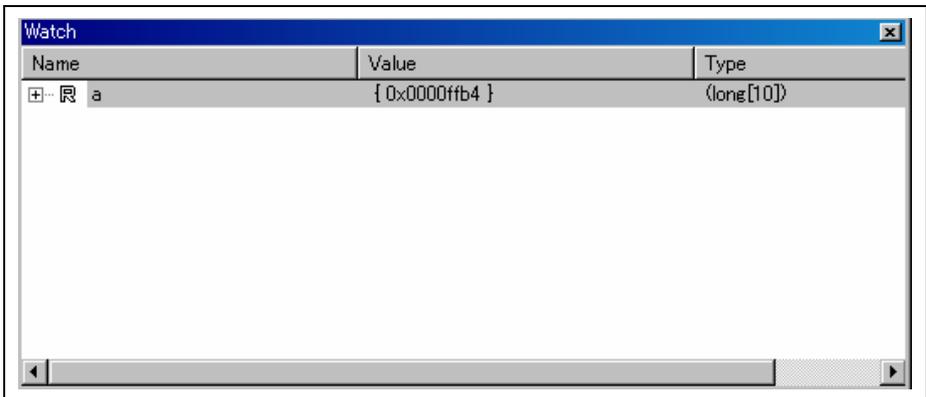


Figure 6.18 [Watch] Window (Displaying the Array)

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

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

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

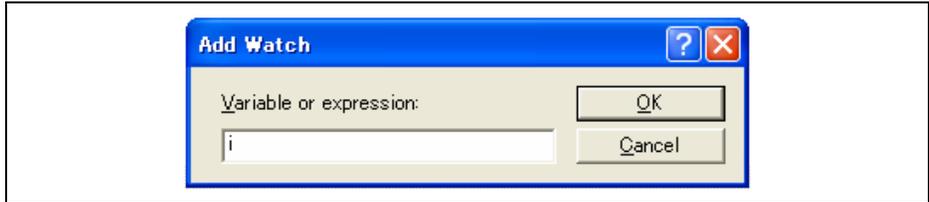


Figure 6.19 [Add Watch] Dialog Box

- Click the [OK] button.

The [Watch] window will now also show the int-type variable `i`.

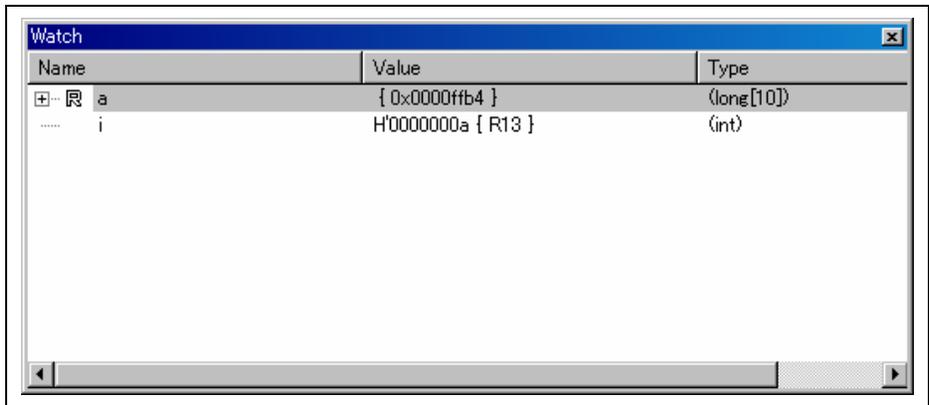


Figure 6.20 [Watch] Window (Displaying the Variable)

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

Name	Value	Type
a	{ 0x0000ffb8 }	(long[10])
R [0]	H'00000000 { 0x0000ffb8 }	(long)
R [1]	H'00000daa { 0x0000ffbc }	(long)
R [2]	H'000020da { 0x0000ffc0 }	(long)
R [3]	H'00002704 { 0x0000ffc4 }	(long)
R [4]	H'00002f5a { 0x0000ffc8 }	(long)
R [5]	H'00003ead { 0x0000ffcc }	(long)
R [6]	H'0000421f { 0x0000ffd0 }	(long)
R [7]	H'00004d1d { 0x0000ffd4 }	(long)
R [8]	H'000053dc { 0x0000ffd8 }	(long)
R [9]	H'00005665 { 0x0000ffdc }	(long)
i	H'0000000a { R13 }	(int)

Figure 6.21 [Watch] Window (Displaying Array Elements)

6.14 Displaying Local Variables

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

- Select [Locals] from the [Symbol] submenu of the [View] menu. The [Locals] window is displayed.

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

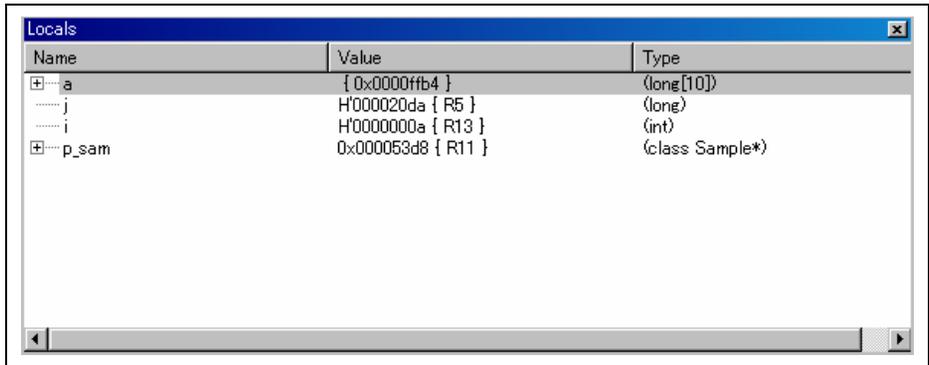


Figure 6.22 [Locals] Window

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

6.15 Stepping Through a Program

The HEW provides a range of step menu commands that allow efficient program debugging.

Table 6.1 Step Option

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

6.15.1 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.



Figure 6.23 [Step In] Button

0x00002000	Sample::Sample()
0x00002002	{
0x00002012	s0=0;
0x00002016	s1=0;
0x00002018	s2=0;
0x0000201a	s3=0;
0x0000201c	s4=0;
0x0000201e	s5=0;
0x00002020	s6=0;
0x00002022	s7=0;
0x00002024	s8=0;
0x00002026	s9=0;
	}
0x0000202e	→ void Sample::sort(long *a)
	{
	long t;
	int i, j, k, gap;
0x0000203e	gap = 5;
0x00002046	while(gap > 0){
0x00002054	for(k=0; k<gap; k++){
0x0000205c	for(i=k+gap; i<10; i=i+gap){
0x0000206c	for(j=i-gap; j>=k; j=j-gap){
	if(a[j]>a[j+gap]){
	t = a[j];
0x00002078	a[j] = a[j+gap];
0x0000207c	a[j+gap] = t;
	}
	else
	break;
	}
	}
	}
	}

Figure 6.24 [Editor] Window (Step In)

- The highlighted line moves to the first statement of the `sort` function in the [Editor] window.

6.15.2 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.

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



Figure 6.25 [Step Out] Button

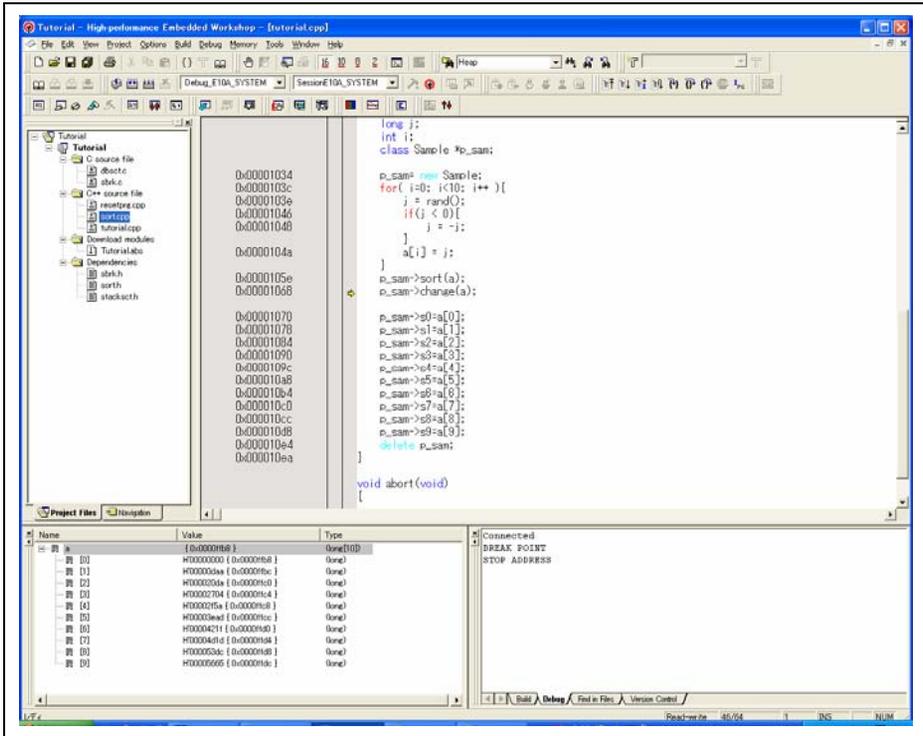


Figure 6.26 [HEW] Window (Step Out)

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

6.15.3 Executing [Step Over] Command

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

- Move to the change function following the procedures described in section 6.15.1, Executing [Step In] Command, in the Debugger Part.
- 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.



Figure 6.27 [Step Over] Button

High-performance Embedded Workshop User's Manual

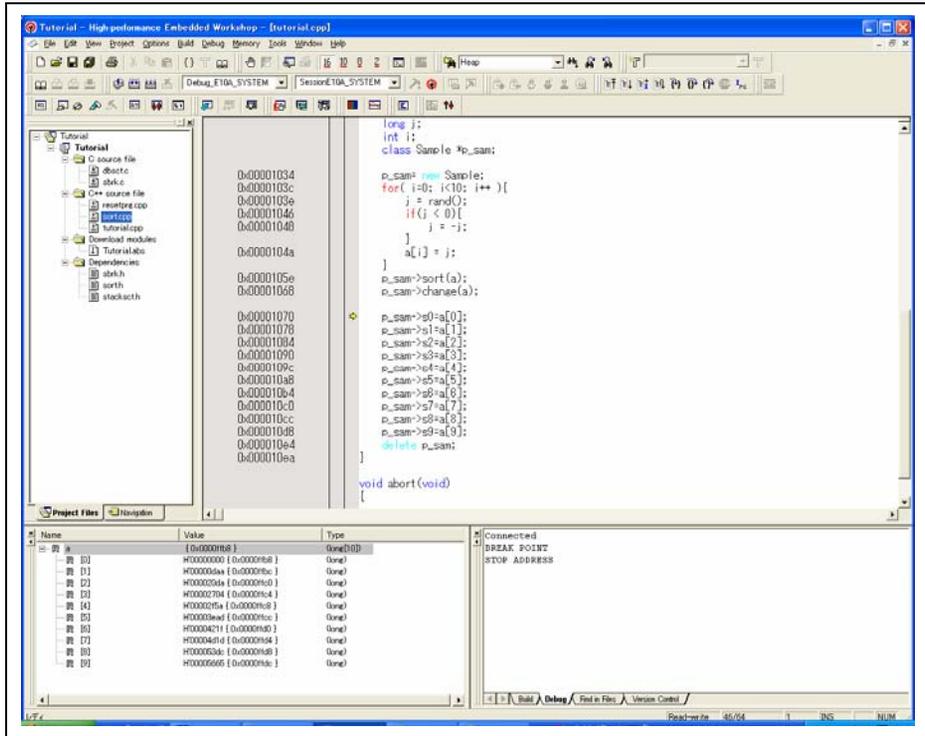


Figure 6.28 [HEW] Window (Step Over)

6.16 Forced Breaking of Program Executions

The HEW can force a break in the execution of a program.

- Cancel all breaks.
- To execute the remaining sections of the main function, select [Go] from the [Debug] menu or the [Go] button on the toolbar.



Figure 6.29 [Go] Button

- The program goes into an endless loop. To force a break in execution, select [Halt] from the [Debug] menu or the [Halt] button on the toolbar.



Figure 6.30 [Halt] Button

6.17 Break Function

The emulator has PC and hardware break functions. With the HEW, a PC breakpoint can be set using the [Event] window, and a hardware break condition can be set using the [Break Condition] dialog box.

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

6.17.1 PC Break Function

The emulator can set up to 255 PC breakpoints. Other methods for setting a PC breakpoint than in section 6.7, Setting a PC Breakpoint, are described below.

- Select [Eventpoints] from the [Code] submenu of the [View] menu. The [Event] window is displayed.
- Select the [Breakpoint] sheet.

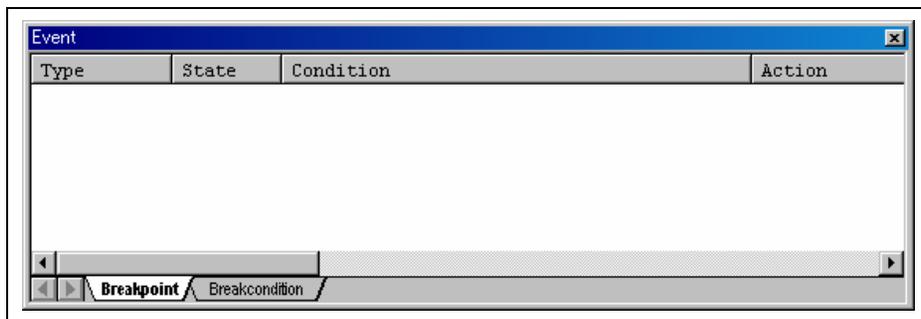


Figure 6.31 [Event] Window (Before PC Breakpoint Setting)

- Click the [Event] window with the right-hand mouse button and select [Add...] from the popup menu.
- Enter **H'00001076** in the [Address] edit box.

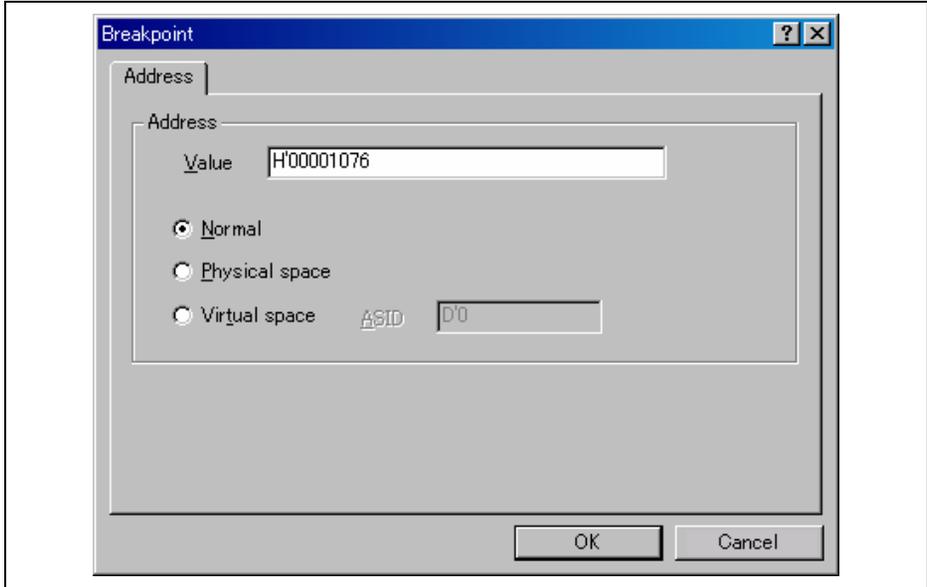


Figure 6.32 [Breakpoint] Dialog Box

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

- Click the [OK] button.

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

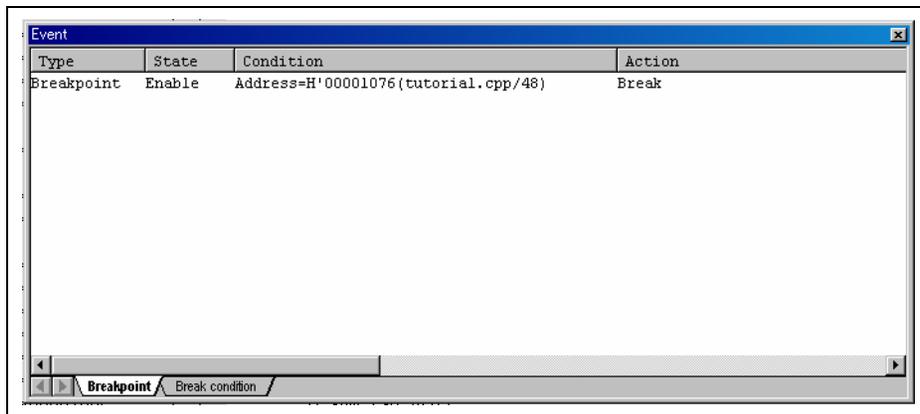


Figure 6.33 [Event] Window (PC Breakpoint Setting)

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

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

- Set the program counter and stack pointer values (PC = H'00000800 and R15 = H'00010000) that were set in section 6.8, Setting Registers, in the [Register] window. Click the [Go] button. When using the MCU with flash memory, specify the end address of the internal RAM for the stack pointer (SP). The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.
- If program execution is failed, reset the device and execute again the procedures above.

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

0x00001024	void main(void)
	{
	long a[10];
	long j;
	int i;
	class Sample *p_sam;
	while (1){
0x00001036	p_sam= new Sample;
0x00001034	for(i=0; i<10; i++){
0x00001038	j = rand();
0x00001044	if(j < 0){
0x00001048	j = -j;
0x00001050	}
	a[i] = j;
0x00001058	}
	p_sam->sort(a);
0x00001068	p_sam->change(a);
0x00001070	
	p_sam->s0=a[0];
0x00001076	p_sam->s1=a[1];
0x0000107a	p_sam->s2=a[2];
0x0000107e	p_sam->s3=a[3];
0x00001082	p_sam->s4=a[4];
0x00001086	p_sam->s5=a[5];
0x0000108a	p_sam->s6=a[6];
0x0000108e	p_sam->s7=a[7];
0x00001092	p_sam->s8=a[8];
0x00001096	p_sam->s9=a[9];
0x0000109a	delete p_sam;
0x0000109e	}

Figure 6.34 [Editor] Window at Execution Stop (PC Break)

The [Status] window displays the following contents.

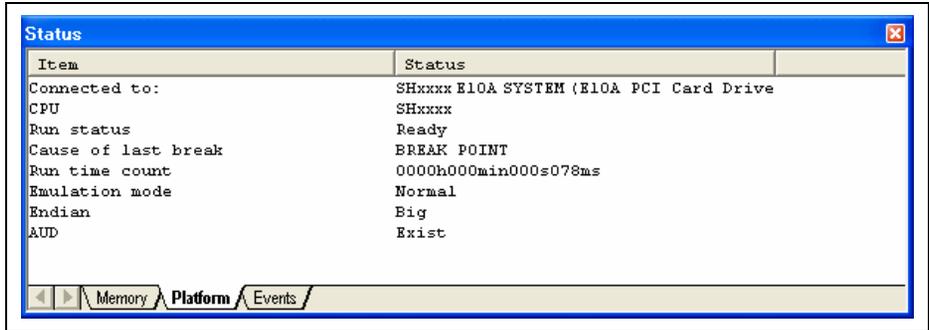


Figure 6.35 Displayed Contents of the [Status] Window (PC Break)

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

6.18 Hardware Break Function

A method is given below in which the address bus condition and the read cycles for the bus status condition are set under Break condition 1 as hardware break conditions.

- Select [Eventpoints] from the [Code] submenu of the [View] menu. The [Event] window is displayed.
- The PC breakpoint that has been previously set is deleted. Click the [Event] window with the right-hand mouse button and select [Delete All] from the popup menu to cancel all PC breakpoints that have been set.
- To set a Break condition 1, click the [Break condition] tab.

Up to three breakpoints can be set independently for the hardware break condition, in the Break conditions 1 to 3. In this example, set the hardware break condition for Break condition 1.

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

- Select a line of Break condition 1 in the [Event] window. When highlighted, double-click this line.

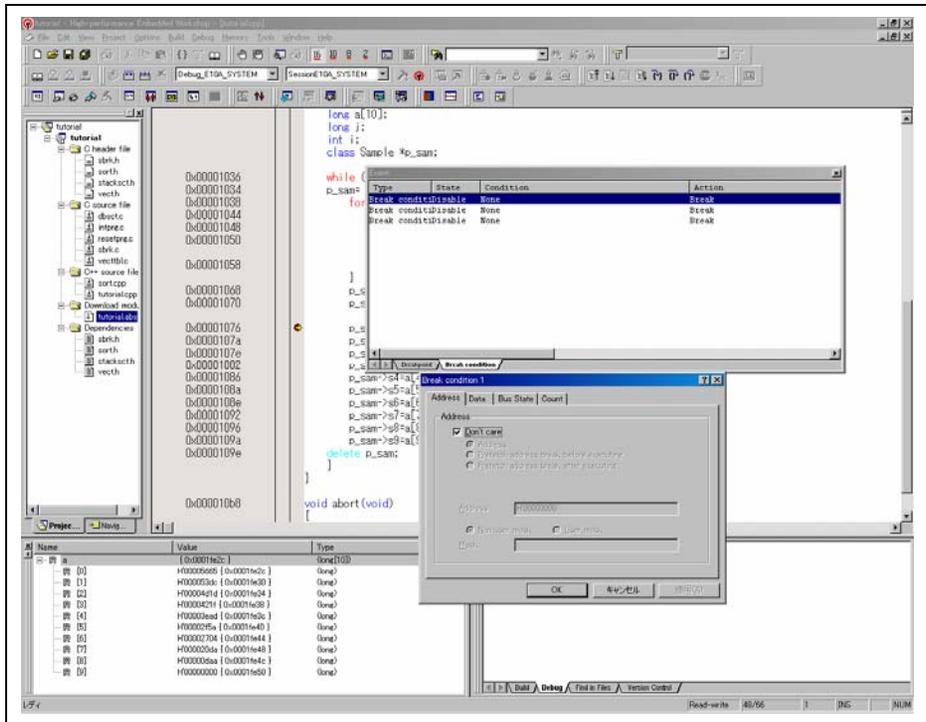


Figure 6.36 [HEW] Window ([Break condition 1])

- The [Break condition 1] dialog box is displayed.
- Clear the [Don't care] check box in the [Address] page.
- Select the [Address] radio button and enter **H'00001068** as the value in the [Address] edit box.
When using the MCU with flash memory, enter **H'00001062** in the [Address] edit box.

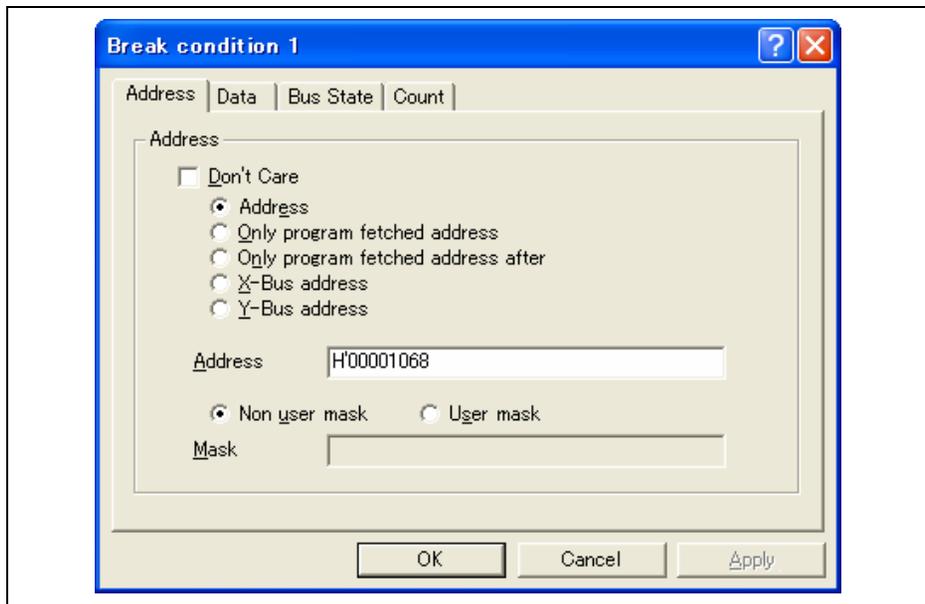


Figure 6.37 [Address] Page ([Break condition 1] Dialog Box)

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

- Select [Bus State] to display the [Bus State] page.
- Select the [Read] radio button in the [Read/Write] group box.

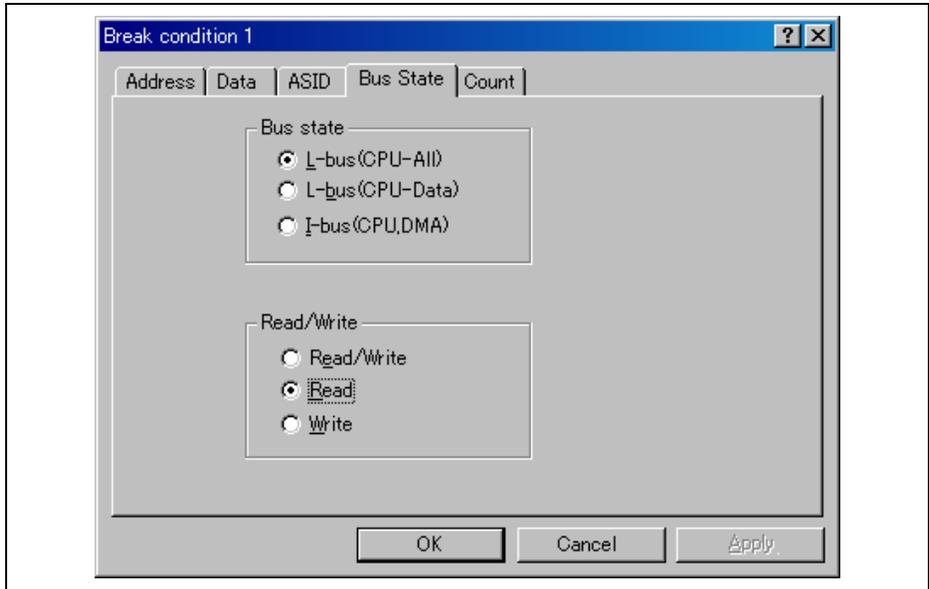


Figure 6.38 [Bus State] Page ([Break condition 1] Dialog Box)

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

- Click the [OK] button.
- The first point display in the State line changes from Disable to Enable.
- The first point display in the Condition line changes from None to Address = H'00001068 direction read.
- Set the program counter and stack pointer values (PC = H'00000800 and R15 = H'00010000) that were set in section 6.8, Setting Registers, in the Debugger Part, in the [Register] window. Click the [Go] button.
When using the MCU with flash memory, specify the end address of the internal RAM for the stack pointer (SP). The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.
- If program execution is failed, reset the device and execute again the procedures above.

The program runs and then stops at the condition specified under Break Condition 1.

```

0x00001024 void main(void)
{
    long a[10];
    long j;
    int i;
    class Sample *p_sam;

    while (1){
        p_sam= new Sample;
        for( i=0; i<10; i++ ){
            j = rand();
            if(j < 0){
                j = -j;
            }
            a[i] = j;
        }
        p_sam->sort(a);
        p_sam->change(a);

        p_sam->s0=a[0];
        p_sam->s1=a[1];
        p_sam->s2=a[2];
        p_sam->s3=a[3];
        p_sam->s4=a[4];
        p_sam->s5=a[5];
        p_sam->s6=a[6];
        p_sam->s7=a[7];
        p_sam->s8=a[8];
        p_sam->s9=a[9];
        delete p_sam;
    }
}

```

Figure 6.39 [Editor] Window at Execution Stop (Break Condition 1)

The [Status] window displays the following contents.

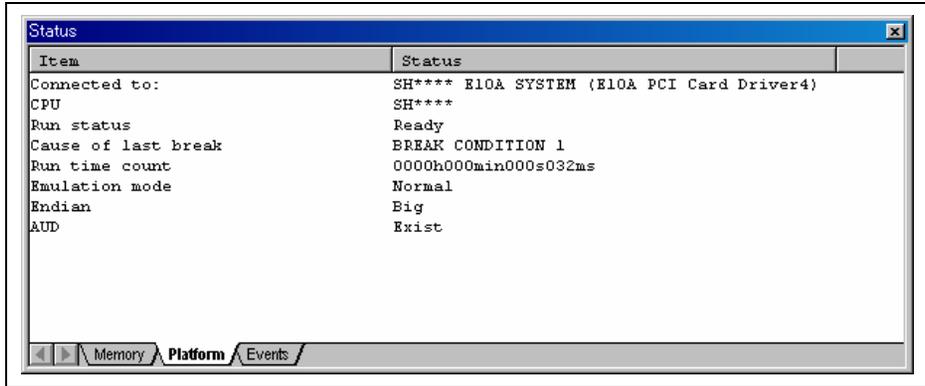


Figure 6.40 Displayed Contents of the [Status] Window (Break Condition 1)

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

6.18.1 Setting the Sequential Break Condition

The emulator has sequential break functions.

Set hardware break conditions as follows:

Break condition 1: When address H'00001068 is accessed in a read cycle, a break condition is satisfied. When using the MCU with flash memory, set the address to H'00001062.

Break condition 2: When address H'00001058 is accessed in a read cycle, a break condition is satisfied. When using the MCU with flash memory, set the address to H'00001050.

Follow the setting method described in the previous section.

To set these breakpoints as sequential:

- Select [Sequential Setting] from the popup menu by clicking the [Event] window with the right-hand mouse button. The [Sequential setting] dialog box will open.

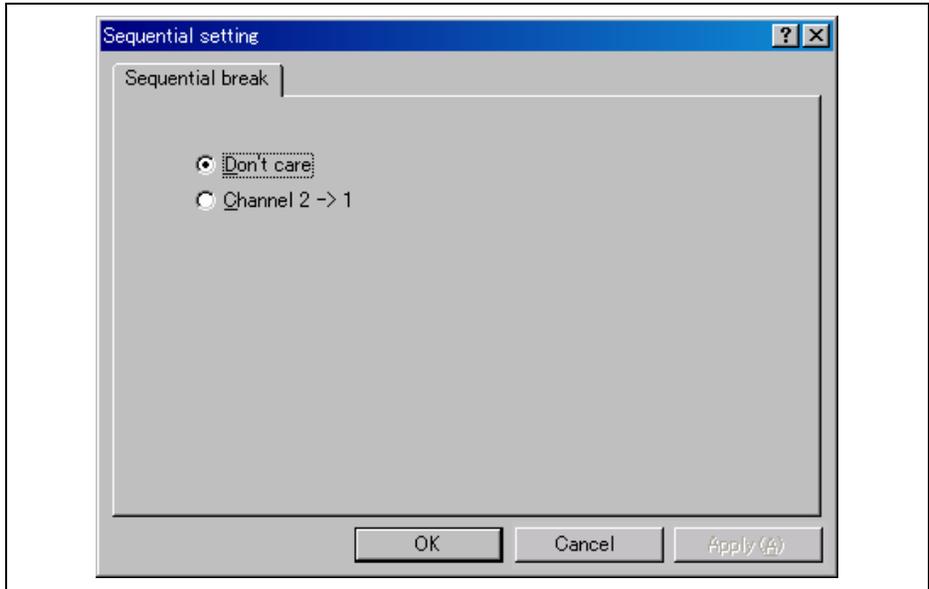


Figure 6.41 [Sequential setting] Dialog Box

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

- Select the [Channel 2 -> 1] radio button and click the [OK] button.

When the setting is completed, the [Event] window will be as shown in figure 6.42.

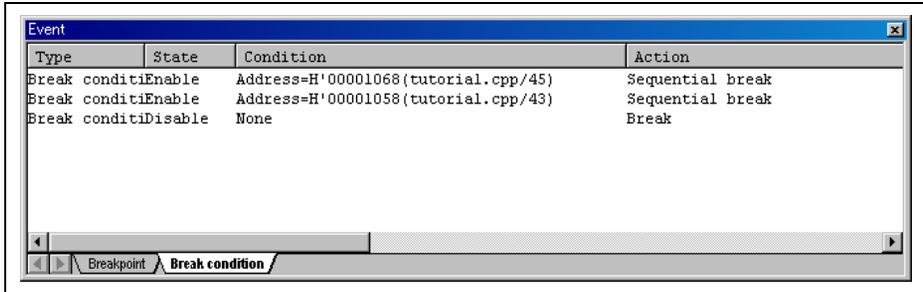


Figure 6.42 [Break condition] Page

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

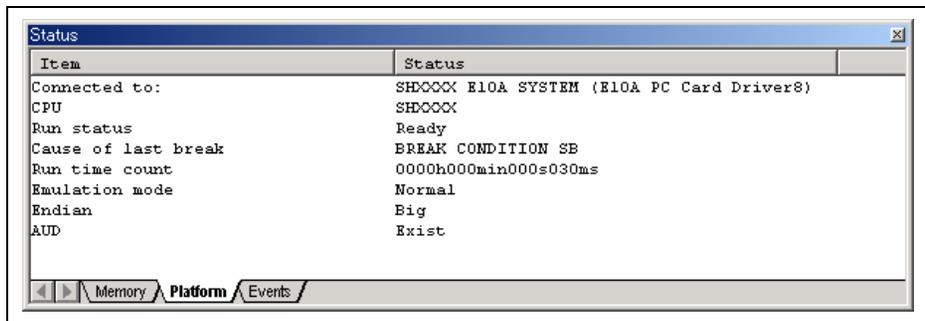
- Set the program counter and stack pointer values (PC = H'00000800 and R15 = H'00010000) that were set in section 6.8, Setting Registers, in the Debugger Part, in the [Register] window. Click the [Go] button.
When using the MCU with flash memory, specify the end address of the internal RAM for the stack pointer (SP). The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.
- If program execution is failed, reset the device and execute again the procedures above.

The program runs and then stops at the condition specified under Break Condition 1.

<pre> 0x00001024 0x00001036 0x00001034 0x00001038 0x00001044 0x00001048 0x00001050 0x00001058 0x00001068 0x00001070 0x00001076 0x0000107a 0x0000107e 0x00001082 0x00001086 0x0000108a 0x0000108e 0x00001092 0x00001096 0x0000109a 0x0000109e </pre>	<pre> void main(void) { long a[10]; long j; int i; class Sample *p_sam; while (1){ p_sam= new Sample; for(i=0; i<10; i++){ j = rand(); if(j < 0){ j = -j; } a[i] = j; } p_sam->sort(a); p_sam->change(a); p_sam->s0=a[0]; p_sam->s1=a[1]; p_sam->s2=a[2]; p_sam->s3=a[3]; p_sam->s4=a[4]; p_sam->s5=a[5]; p_sam->s6=a[6]; p_sam->s7=a[7]; p_sam->s8=a[8]; p_sam->s9=a[9]; delete p_sam; } } </pre>
---	--

Figure 6.43 [Editor] Window at Execution Stop (Sequential Break)

The [Status] window displays the following contents.



The screenshot shows a window titled "Status" with a table of system information. The table has two columns: "Item" and "Status". The items listed are: Connected to: SHEXXX E10A SYSTEM (E10A PC Card Driver8), CPU: SHEXXX, Run status: Ready, Cause of last break: BREAK CONDITION SB, Run time count: 0000h000min000s030ms, Emulation mode: Normal, Endian: Big, and AUD: Exist. At the bottom of the window, there are navigation buttons and tabs for "Memory", "Platform", and "Events".

Item	Status
Connected to:	SHEXXX E10A SYSTEM (E10A PC Card Driver8)
CPU	SHEXXX
Run status	Ready
Cause of last break	BREAK CONDITION SB
Run time count	0000h000min000s030ms
Emulation mode	Normal
Endian	Big
AUD	Exist

Figure 6.44 Displayed Contents of the [Status] Window (Sequential Break)

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

6.19 Trace Functions

The emulator has two branch-instruction trace functions.

- Internal Trace Function

The branch source and branch destination addresses, mnemonics, operands, and source lines are displayed. Since this function uses the trace buffer built into the MCU, a realtime trace can be acquired.

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

- AUD Trace Function

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

The following information can be acquired:

- Types of trace information: Branch information, memory access information from the CPU, and PC or Rn value during the Trace Rn instruction execution
- Trace acquisition address value
- Data value
- Mnemonic
- Operand
- Source line

- Notes:
1. The AUD trace function is not supported for all products. For the specifications of each product, refer to the online help.
 2. The types of trace information that can be acquired by an AUD trace function differ according to the product. For the specifications of each product or the number of acquired branches, refer to the online help.
 3. When multiple loops are performed to reduce the number of AUD trace displays, only the IP counts up according to the product.

6.19.1 Displaying the Trace Window

Select [Trace] from the [Code] submenu of the [View] menu. The result of the acquired trace is displayed.

6.19.2 Internal Trace Function

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

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

The type is specified as follows:

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

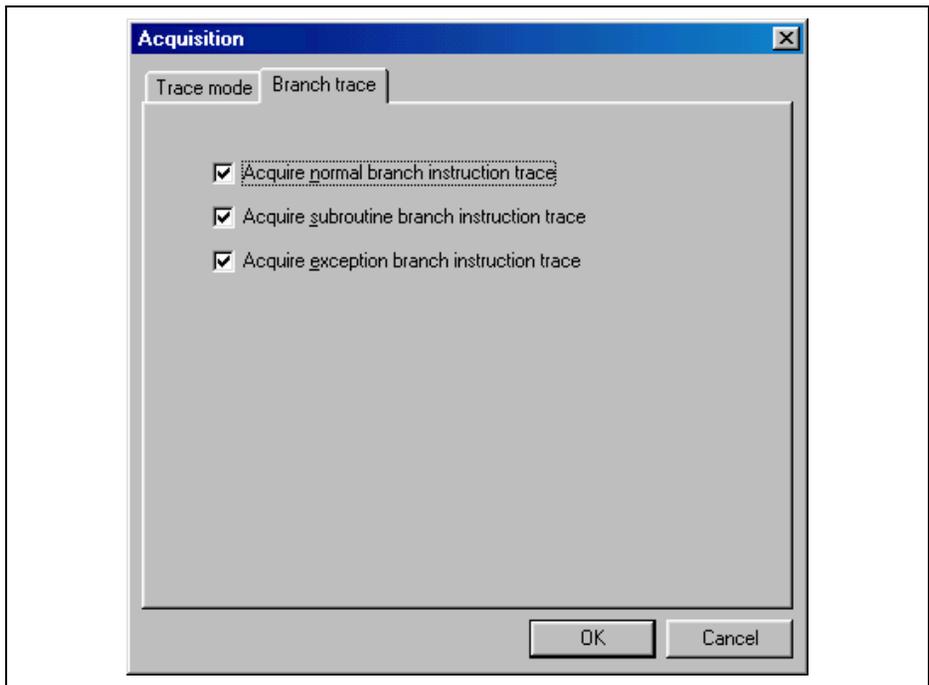


Figure 6.45 [Branch trace] Page

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

Run the program as shown in the example of section 6.17.1, PC Break Function. The trace results are displayed in the [Trace] window after the program execution is completed.

PTR	IP	Type	Bus	R/W	Address	Data	Source	Label
-000012		DESTINATION			00002054		for(j=1-gap; j>=k; j=j-gap) {	
-000011	-D'0005	BRANCH			0000205E			
-000010		DESTINATION			00002074			
-000009	-D'0004	BRANCH			00002076			
-000008		DESTINATION			00002062		if(a[j]>a[j+gap]) {	
-000007	-D'0003	BRANCH			00002076			
-000006		DESTINATION			00002062		if(a[j]>a[j+gap]) {	
-000005	-D'0002	BRANCH			00002076			
-000004		DESTINATION			00002062		if(a[j]>a[j+gap]) {	
-000003	-D'0001	BRANCH			0000206A			
-000002		DESTINATION			00002078			
-000001	-D'0000	BRANCH			0000209E			
+000000		DESTINATION			00001068			

Figure 6.46 [Trace] Window

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

Note: The number of branch instructions that can be acquired by a trace differs according to the product. For the number that can be specified for each product, refer to the online help.

6.19.3 AUD Trace Function

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

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

(1) Setting the trace acquisition mode

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

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

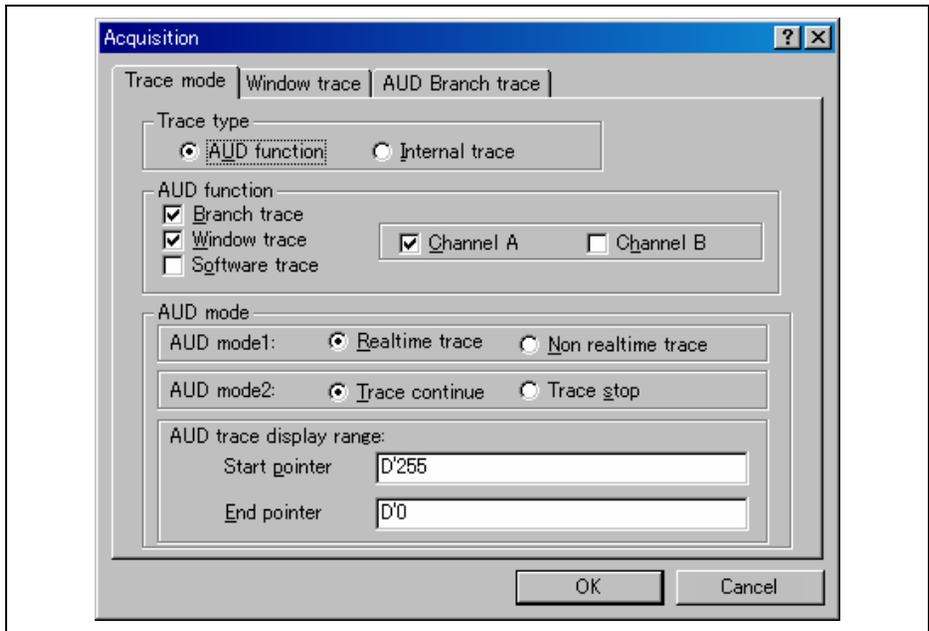


Figure 6.47 [Trace mode] Page

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

The following table shows the options.

AUD Trace Acquisition Mode

Type	Mode	Description
Continuous trace occurs	Realtime trace	When the next branch occurs while the trace information is being output, all the information may not be output. The user program can be executed in realtime, but some trace information may be lost.
	Non realtime trace	When the next branch occurs while the trace information is being output, the CPU stops operations until the information is output. The user program is not executed in realtime.
Trace buffer full	Trace continue	This function always overwrites the oldest trace information to acquire the latest trace information.
	Trace stop	After the trace buffer becomes full, the trace information is no longer acquired. The user program is continuously executed.

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

(2) Displaying the trace result

- Run the program as shown in the example of section 6.17.1, PC Break Function, in the Debugger Part. The trace results are displayed in the [Trace] window after the program execution is completed.

The trace results are displayed in the [Trace] window after the program execution is completed.

The following is an example of the trace display.

PTR	IP	Type	Bus	R/W	Address	Data	Source	Label
-000341	-D'000253	BRANCH			00002076			
-000340		DESTINATION			00002062			
-000339	-D'000252	LOST					if(a[j]>a[j+gap]){	
-000338	-D'000251	MEMORY	L-Bus	READ	0000FF38	00000DAA		
-000337	-D'000250	LOST						
-000336	-D'000249	MEMORY	L-Bus	READ	0000FF4C	000020DA		
-000335	-D'000248	BRANCH			0000206A			
-000334		DESTINATION			00002078			
-000333	-D'000247	BRANCH			00002092			
-000332		DESTINATION			00002042			
-000331	-D'000246	BRANCH			0000204A			
-000330		DESTINATION			00002080			
-000329	-D'000245	BRANCH			00002082			
-000328		DESTINATION			0000204E		for(i=k+gap; i<10; i=	
-000327	-D'000244	BRANCH			00002050			
-000326		DESTINATION			0000207A			
-000325	-D'000243	BRANCH			0000207C			
-000324		DESTINATION			00002054		for(j=i-gap; j>=k; j=j	
-000323	-D'000242	BRANCH			0000205E			
-000322		DESTINATION			00002074			
-000321	-D'000241	BRANCH			00002076			
-000320		DESTINATION			00002062		if(a[j]>a[j+gap]){	
-000319	-D'000240	LOST						
-000318	-D'000239	MEMORY	L-Bus	READ	0000FF28	00000000		
-000317	-D'000238	LOST						
-000316	-D'000237	MEMORY	L-Bus	READ	0000FF30	00002704		
-000315	-D'000236	BRANCH			0000206A			
-000314		DESTINATION			00002078			

Figure 6.48 [Trace] Window (Example)

6.19.4 MMU Support

This function can be used when the supported MCU has an MMU.

- TLB window

In the emulator, the contents of the TLB table can be easily displayed and edited by selecting [CPU -> TLB] from the [View] menu. For details, refer to the online help.

- VP_MAP translation function

The MCU, which has an MMU, translates internal addresses (virtual addresses) to actual memory addresses (physical addresses). Address translation is performed according to the address translation table (translation look-aside buffer: TLB) in the MCU. The MMU operates during command input wait state as well as during user program execution. When a command for memory access is executed while the MMU address translation function is enabled, the address translated by the MMU is accessed. If the specified address is not within the TLB, a TLB miss occurs, and the TLB must be updated by the user program.

The emulator has address translation functions according to the VP_MAP tables. The VP_MAP tables are the address translation tables for the emulator created with the VP_MAP_SET command.

The following shows an example of how to use the VP_MAP tables.

Example:

1. Create VP_MAP tables for translating virtual addresses H'10000 to H'10fff to physical addresses H'4000000 to H'4000fff and virtual addresses H'11000 to H'11fff to physical addresses H'0 to H'fff.

```
>vs 10000 10fff 4000000 (RET)
>vs 11000 11fff 0 (RET)
>vd (RET)
<VADDR_TOP> <VADDR_END> <PADDR_TOP>
00010000    00010fff    04000000
00011000    00011fff    00000000
DISABLE
```

- Then, enable the VP_MAP tables. (When the tables are disabled, addresses are not translated.)

```
>ve enable (RET)
>vd (RET)
<VADDR_TOP> <VADDR_END> <PADDR_TOP>
00010000    00010fff    04000000
00011000    00011fff    00000000
ENABLE
```

Here, virtual addresses correspond to physical addresses as shown in figure 6.49.

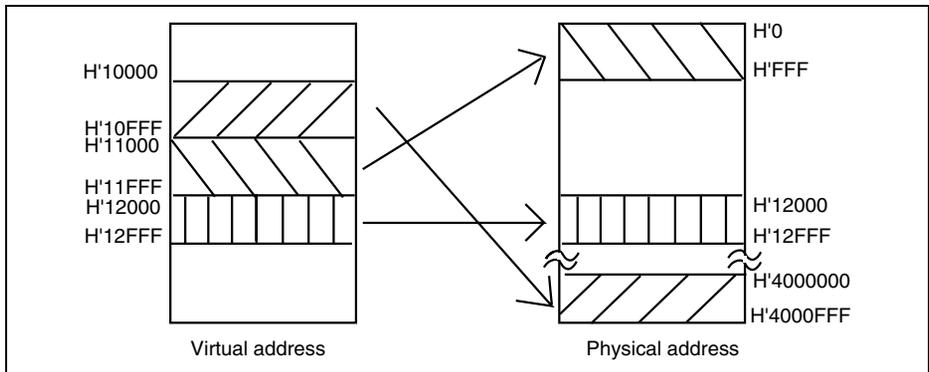


Figure 6.49 Address Translation according to VP_MAP Tables

How to translate addresses depends on the settings of the radio buttons of the [Memory area] group in the [Configuration] dialog box. The following shows how to translate addresses in each setting state.

- When the **Normal** radio button is selected:
The VP_MAP table has a priority over the TLB. When the VP_MAP table is enabled and the specified address is within the VP_MAP table settings, the emulator translates the address according to the VP_MAP table. If the specified address is outside the VP_MAP table settings even when the VP_MAP table is enabled, or when the VP_MAP table is disabled, the emulator translates the address according to the MMU state.
- When the **Physical** radio button is selected:
The address is not translated.
- When the **Virtual** radio button is selected:
The address is translated according to the TLB. If the specified address is outside the TLB table settings, a TLB error will occur.

Table 6.2 Address Translation Tables

Radio Button*	VP_MAP		MMU		Table Used for Translation	
	Enabled/ Disabled	Within/ Outside the Range	Enabled/ Disabled	Within/Outside the TLB Range		
Normal	Enabled	Within the range	Enabled	Within the range	Translated according to the VP_MAP table	
				Outside the range	Translated according to the VP_MAP table	
			Disabled	Within/outside the range	Translated according to the VP_MAP table	
		Outside the range	Enabled	Within the range	Translated according to the TLB table	
				Outside the range	TLB error	
			Disabled	Within/outside the range	Not translated	
	Disabled	Within/ outside the range	Enabled	Within the range	Translated according to the TLB table	
				Outside the range	TLB error	
			Disabled	Within/outside the range	Not translated	
				Outside the range	Not translated	
Virtual	Enabled/ disabled	Within/ outside the range	Enabled	Within the range	Translated according to the TLB table	
				Outside the range	TLB error	
				Disabled	Within the range	Translated according to the TLB table
			Disabled	Outside the range	Within the range	Translated according to the TLB table
					Outside the range	TLB error
					Outside the range	TLB error
Physical	Enabled/ disabled	Within/ outside the range	Enabled/ disabled	Within/outside the range	Not translated	
			Enabled/ disabled	Within/outside the range	Not translated	

Note: Specified by the [Memory area] group box in the [Configuration] dialog box.

6.20 Stack Trace Function

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

Note: This function can be used only when the load module that has the Dwarf2-type debugging information is loaded. Such load modules are supported in SHC/C++ compiler V6.0 or later.

- Double-click the [Editor] column in the `sort` function and set a PC breakpoint.

0x0000202e	void Sample::sort(long *a)
	{
	long t;
	int i, j, k, gap;
0x0000203e	gap = 5;
0x00002046	while(gap > 0){
0x00002054	for(k=0; k<gap; k++){
0x0000205c	for(i=k+gap; i<10; i=i+gap){
0x0000206c	for(j=i-gap; j>=k; j=j-gap){
	if(a[j]>a[j+gap]){
	t = a[j];
	a[j] = a[j+gap];
	a[j+gap] = t;
0x00002078	}
0x0000207c	} else break;
	}
	}
	} gap = gap/2;
0x00002090	}
0x000020ae	void Sample::change(long *a)
	{

Figure 6.50 [Editor] Window (PC Breakpoint Setting)

- Set the same program counter and stack pointer values (PC = H'00000800 and R15 = H'00010000) as were set in section 6.8, Setting Registers, in the Debugger Part (again, use the [Register] window). Click the [Go] button.

When using the MCU with flash memory, specify the end address of the internal RAM for the stack pointer (SP). The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.

- If program execution is failed, reset the device and execute again the procedures above.
- After the break in program execution, select [Stack Trace] from the [Code] submenu of the [View] menu to open the [Stack Trace] window.

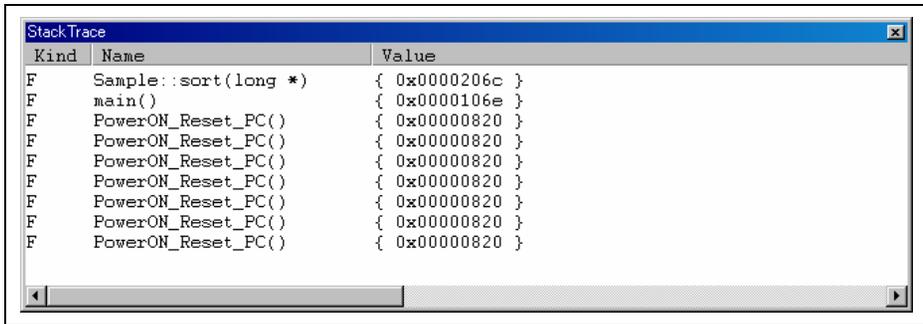


Figure 6.51 [Stack Trace] Window

Figure 6.51 shows that the position of the program counter is currently at the selected line of the `sort()` function, and that the `sort()` function is called from the `main()` function.

To remove the PC breakpoint, double-click the [Editor] column in the `sort()` function again.

Note: For details on this function, refer to the online help.

6.21 Performance Measurement Function

The emulator has performance measurement functions.

- Performance measurement function

This function applies a counter in the MCU to measure the number of times various events have occurred and cycle count. A start and end condition for counting can be set.

Various items that can be measured differ according to the supported MCU.

- Profiling function

The profiling function is used to measure the performance of each function.

When execution of the user program is ended, the number of times each function is called and the measured data are displayed. The measured items are the same as those for the number of times various events have occurred, and cycle count.

6.21.1 Performance Measurement Function

The following is an example of the use of a counter in the MCU to measure the number of times various events have occurred and cycle count.

This function cannot be used with the profiling function that is described later.

(1) Setting method

Select [Performance Analysis] from the [Performance] submenu of the [View] menu. When the [Select Performance Analysis Type] dialog box will open, click the [OK] button.

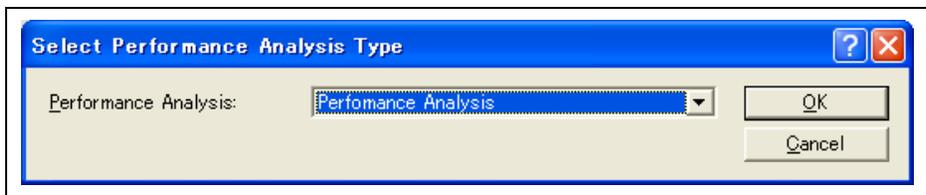
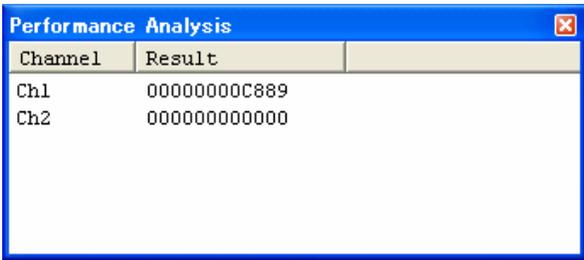


Figure 6.52 [Select Performance Analysis Type] Dialog Box

- The [Performance Analysis] window will be displayed.
- Place the mouse cursor anywhere within this window, click the right-hand mouse button, and then select [Set] from the popup menu. The [Performance Analysis] dialog box will open. The events to be measured and measuring conditions can be set in this dialog box.

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

After the conditions have been set, clicking the [OK] button and executing the user program will display the result of measurement in the [Performance Analysis] window.



Channel	Result
Ch1	00000000C889
Ch2	000000000000

Figure 6.53 [Performance Analysis] Window

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

6.21.2 Profiling Function

The profiling function can measure performance for each function.

- Notes:
1. Realtime operation is not possible while this function is in operation, since internal breaks are generated during program execution. Measuring the profile itself affects the measurements.
 2. Performance profile measurement is not supported for all products. On those products for which it is supported, its characteristics differ according to the product. For specifications for each product, refer to the additional document, Specific Guide for the SHxxx E10A Emulator.
 3. This function cannot be used with the performance measurement function that has been described before. Use one of these functions.

- First, download the tutorial program by referring to section 6.6, Downloading the Tutorial Program.

- Select [Profile] from the [Performance] submenu of the [View] menu. The [Profile] window will be displayed.

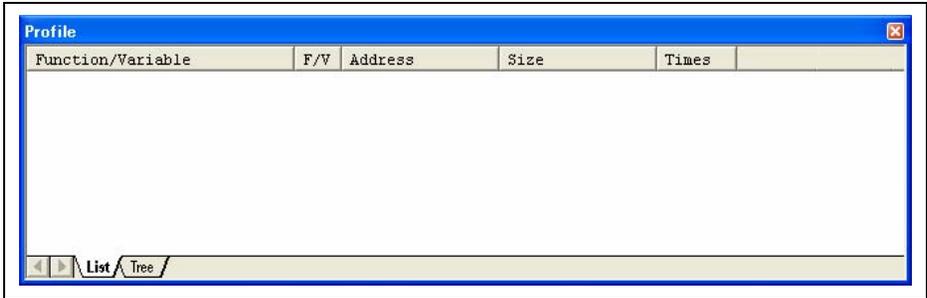


Figure 6.54 [Profile] Window

- To enable the profiling function, place the mouse cursor on an entry in the [List] sheet of the [Profile] window, click the right-hand mouse button, and then select [Enable Profiler] from the popup menu.

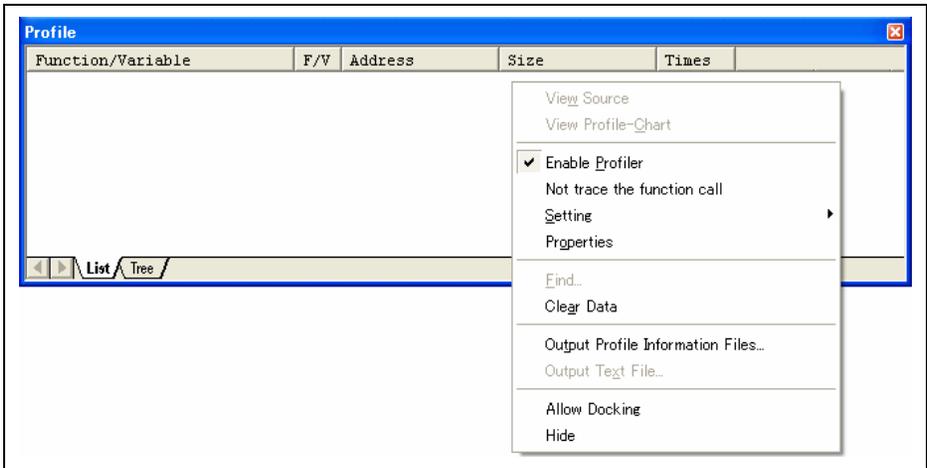


Figure 6.55 Selection of [Enable Profiler]

- To set the data to be measured for the selected function, place the mouse cursor on an entry in the [List] sheet of the [Profile] window, click the right-hand mouse button, and then select [Properties] from the popup menu. The [Select Data] dialog box is displayed.

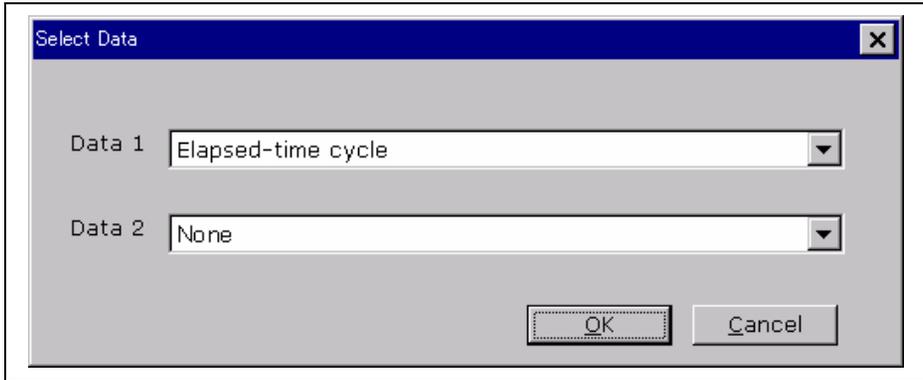


Figure 6.56 [Select Data] Dialog Box

- Use the [Select Data] dialog box to select the data to be measured. Elapsed-time cycle is selected as a first item to be measured.
- After the data has been selected, press the [OK] button.
- Double-click the [Edit] column for the while statement of the main function to set a PC breakpoint.

0x00001024	void main(void)
	{
	long a[10];
	long j;
	int i;
	class Sample *p_sam;
0x00001036	while (1){
0x00001034	p_sam= new Sample;
0x00001038	for(i=0; i<10; i++){
0x00001044	j = rand();
0x00001048	if(j < 0){
0x00001050	j = -j;
	}
0x00001058	a[i] = j;
	}
0x00001068	p_sam->sort(a);
0x00001070	p_sam->change(a);
0x00001076	p_sam->s0=a[0];
0x0000107a	p_sam->s1=a[1];
0x0000107e	p_sam->s2=a[2];
0x00001082	p_sam->s3=a[3];
0x00001086	p_sam->s4=a[4];
0x0000108a	p_sam->s5=a[5];
0x0000108e	p_sam->s6=a[6];
0x00001092	p_sam->s7=a[7];
0x00001096	p_sam->s8=a[8];
0x0000109a	p_sam->s9=a[9];
0x0000109e	delete p_sam;
	}
	}
0x000010b8	void abort(void)

Figure 6.57 [Editor] Window (PC Breakpoint Setting)

- Set the same program counter and stack pointer values (PC = H'00000800 and R15 = H'00010000) as were set in section 6.8, Setting Registers, in the Debugger Part (again, use the [Register] window). Click the [Go] button.

When using the MCU with flash memory, specify the end address of the internal RAM for the stack pointer (SP). The internal RAM area differs depending on the MCU. Refer to the hardware manual of the MCU used.

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

After the break in program execution, the results of the measurements are displayed in the [List] sheet of the [Profile] window.

Function/Variable	F/V	Address	Size	Times	Data1	Data2
PowerON_Reset_PC	F	H'00000800	H'0000002C	1	433	
Sample::change(long *)	F	H'000020B6	H'00000042	1	4281	
Sample::sort(long *)	F	H'0000202E	H'00000088	1	12920	
Sample::Sample()	F	H'00002000	H'0000002E	1	589	
_divlu	F	H'00001460	H'00000000	1	333	
0x00001368	F	H'00001368	H'00000000	1	912	
_malloc	F	H'000012B8	H'00000000	1	1736	
_free	F	H'00001214	H'00000000	1	372	
_rand	F	H'000011E8	H'0000002C	10	3660	
operator new(unsigned long)	F	H'00001184	H'00000064	1	792	
_CALL_INIT	F	H'00001130	H'0000002C	1	468	
_INITISCT	F	H'000010C8	H'00000000	1	38334	
_main	F	H'00001024	H'0000008A	1	4785	
_sbrk	F	H'00001000	H'00000024	1	236	

Figure 6.58 [List] Sheet of the [Profile] Window

Figure 6.59 shows the [Tree] sheet of the [Profile] window.

Function	Address	Size	Stack Size	Times	Data1	Data2
[-] C:\Hew2\Tools\Renesas\Debug...						
[-] PowerON_Reset_PC	H'00000800	H'0000002C	H'00000000	1	433	
[-] _INITISCT	H'000010C8	H'00000000	H'00000000	1	38334	
[-] _CALL_INIT	H'00001130	H'0000002C	H'00000000	1	468	
[-] _main	H'00001024	H'0000008A	H'00000000	1	4785	
[-] Sample:Sample0	H'00002000	H'0000002E	H'00000000	1	589	
[-] operator new(un...	H'00001184	H'00000064	H'00000000	1	792	
[-] _malloc	H'000012B8	H'00000000	H'00000000	1	1736	
[-] Sample:change(long...	H'000020B6	H'00000042	H'00000000	1	4281	
[-] Sample:sort(long *)	H'0000202E	H'00000088	H'00000000	1	12920	
[-] _rand	H'000011E8	H'0000002C	H'00000000	10	3660	

Figure 6.59 [Tree] Sheet of the [Profile] Window

When [View Profile-Chart] is selected from the popup menu, a chart diagram is shown.

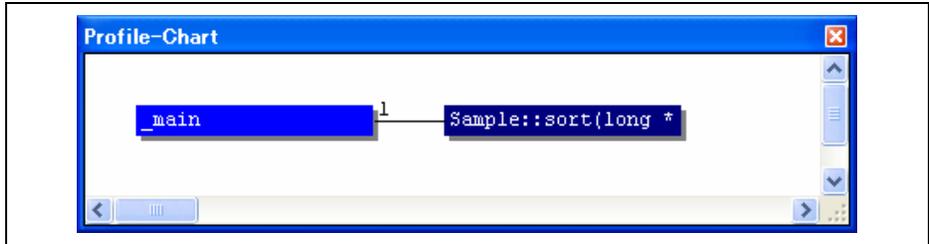


Figure 6.60 [Profile-Chart] Window

6.22 Download Function to the Flash Memory Area

The emulator enables downloading to the flash memory area. This function requires a program for writing the flash memory (hereinafter referred to as a write module), a program for erasing the flash memory (hereinafter referred to as an erase module), and the RAM area for downloading and executing these modules.

Notes: 1. The write and erase modules must be prepared by the user.

2. This function is not available for the MCU with flash memory.

- Interface with write and erase modules and emulator firmware

The write and erase modules must be branched from the emulator firmware. To branch from the emulator firmware to the write and erase modules, or to return from the write and erase module to the emulator firmware, the following conditions must be observed:

- Describe all the write and erase modules with the assembly language.
- Guarantee all the general register values and control register values before and after calling the write or erase module.
- Return the write or erase module to the calling source after processing.
- The write and erase module must be a Motorola-type file.

The module interface must be as follows to pass correctly the information that is required for flash memory accessing.

Table 6.3 Module Interface

Module Name	Argument	Return Value
Write module	R4(L): Write address R7(L): Verify option 0 = no verify, 1 = verify R5(L): Access size 0x4220 = byte, 0x5720 = word, 0x4C20 = longword R6(L): Write data	R0(L): End code Normal end = 0, Abnormal end = other than 0, Verify error = BT
Erase module	R4(L): Access size 0x4220 = byte, 0x5720 = word, 0x4C20 = longword	None

Note: The (L) means the longword size.

Note: Write module: The write data for the access size is set to the R6 register. When the access size is word or byte, 0 is set to the upper bits of the R6 register.

- Flash memory download method

For downloading to the flash memory, set the items on the [Loading flash memory] page in the [Configuration] dialog box, which is opened from [System...], then [Emulator] from the [Options] menu.

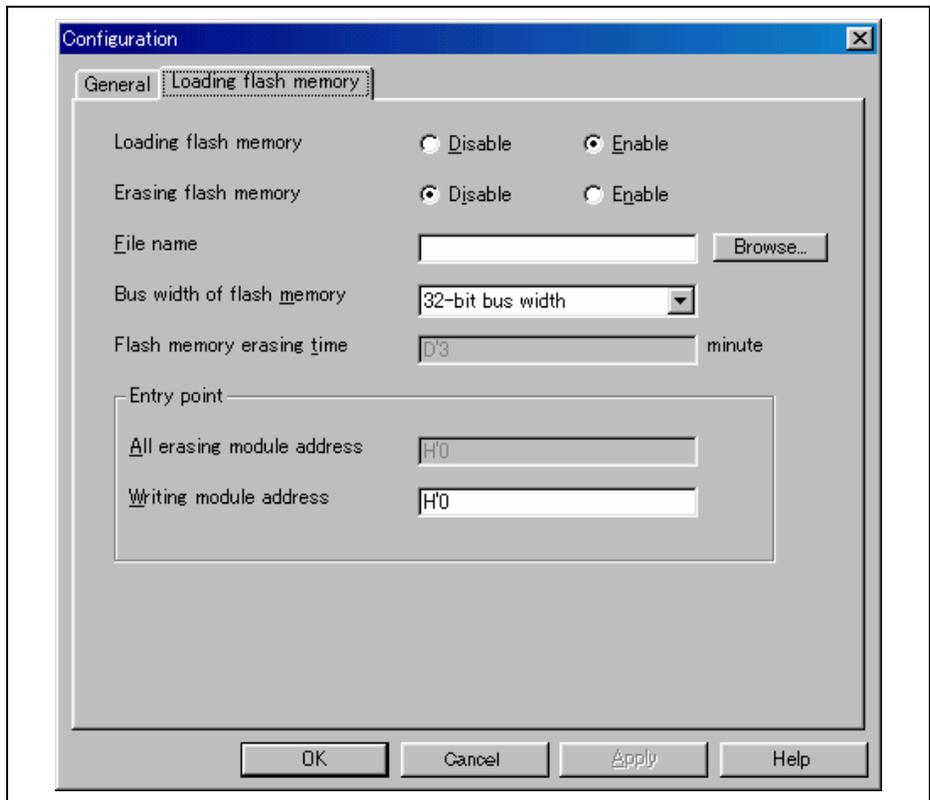


Figure 6.61 [Loading flash memory] Page

Table 6.4 shows the options for the [Loading flash memory] page.

Table 6.4 [Loading flash memory] Page Options

Option	Description
[Loading flash memory] radio button	Sets Enable for flash memory downloading. When Enable is selected, and [File load] is selected from the [File] menu for downloading, the write module is always called. Enable: Download to the flash memory Disable: Not download to the flash memory
[Erasing flash memory] radio button	Sets Enable for erasing before the flash memory is written. When Enable is selected, the erase module is called before calling the write module. Enable: Erase the flash memory Disable: Not erase the flash memory
[File name] edit box	Sets the file name of the S-type load module including the write and erase modules. The file that has been set is loaded to the RAM area before loading to the flash memory. A maximum of 128 characters can be input for the file name.
[Bus width of flash memory] list box	Sets the bus width of the flash memory.
[Flash memory erasing time] edit box*	Sets the TIMEOUT value for erasing the flash memory. Set a larger value if erasing requires much time; the default time is three minutes. The radix for the input value is decimal. It becomes hexadecimal by adding H'.
[Entry point] group box	Sets the calling destination address of the write and erase modules. [All erasing module address] edit box: Inputs the calling destination address of the erase module. [Writing module address] edit box: Inputs the calling destination address of the write module.

Note: Although the values that can be set are D'1 to D'65535, the TIMEOUT hours may be extended according to the set value. Therefore, it is recommended to input the minimum value by considering the erasing time of the flash memory in use.

- Notes on using the flash memory download function

The following are notes on downloading to the flash memory.

- When the flash memory download is enabled, downloading to areas other than the flash memory area is disabled.
- Downloading is only enabled to the flash memory area. Perform memory write or PC break only to the RAM area.
- When the flash memory erase is enabled, the [Stop] button cannot stop erasing.
- The area for the write and erase modules must be set in an MMU-disabled space.

- An example of downloading to the flash memory

The following is an example of downloading to the flash memory manufactured by Intel Corporation (type number: G28F640J5-150) that has been mounted on Renesas's SH7751 CPU board (type number: HS7751STC01H). A sample is provided in the \Fmtool folder in the installation destination folder. Create a program that suits the user specifications by referring to this sample.

Table 6.5 Board Specifications

Item	Contents	
SDRAM address	H'0C000000 to H'0FFFFFFF	
Flash memory address	H'00000000 to H'01FFFFFF	
Bus width of flash memory	32 bits	
Operating environment	CPU internal frequency	167 MHz
	Bus frequency	55.7 MHz
	CPU internal module frequency	27.84 MHz
	Endian	Big endian

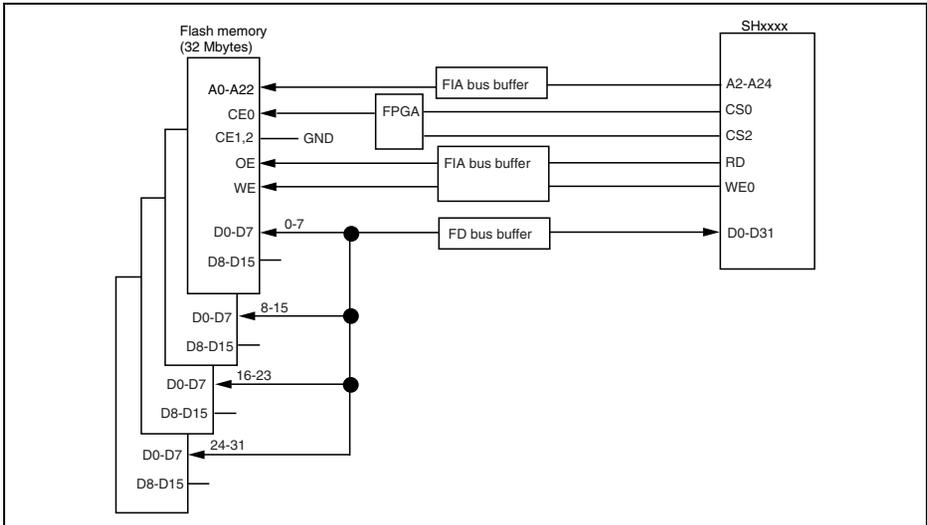
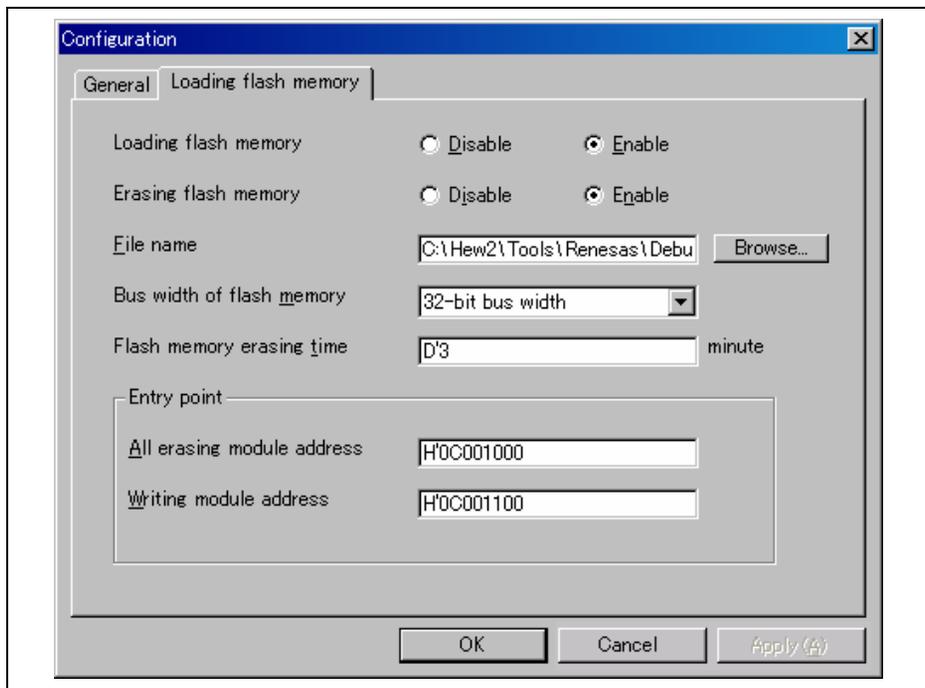


Figure 6.62 Flash Memory Wiring

Table 6.6 Sample Program Specifications

Item	Contents
RAM area to be used	H'0C001000 to H'0C0015BF
Write module start address	H'0C001100
Erase module start address	H'0C001000

- Since the SDRAM is used, the bus controller must be set.
- Set the options on the [Loading flash memory] page in the [Configuration] dialog box as follows:

**Figure 6.63 [Loading flash memory] Page**

- Notes:
1. When the data has already been written in the flash memory, be sure to select [Enable] for [Erasing flash memory]. If [Disable] is selected, a verify error occurs.
 2. When [Erasing flash memory] is selected, it takes about one minute to erase the flash memory (in this example).
- Select the object for downloading to the flash memory area.

6.23 What Next?

This tutorial has described the major features of the emulator and the use of the HEW.

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.

Appendix A Troubleshooting

1. *I have a text file open in the editor but syntactic color-coding is not being displayed.*

Ensure that you have named the file (i.e. saved it) and that the “Syntax coloring” check box is set on the “Editor” tab of the “Options” dialog box, which is launched via [**Tools -> Options...**]. HEW checks a file group to which the file’s extension belongs and decides whether the file is colored. To view the currently defined extensions and their file groups, use the “File Extensions” dialog box which is launched via [**Project -> File Extensions...**]. To view coloring information, select [**Tools -> Format**] to display the “Color” tab on the “Format” dialog box. (See the “Syntax Coloring” section in chapter 4, “Using the Editor” of the HEW Part for details.)

2. *I want to change the settings of a tool but the [Tools->Administration...] menu option is not selectable.*

[**Tools -> Administration...**] cannot be selected while a workspace is open. To enter the “Tool Administration” dialog box, close the current workspace.

3. *I opened a workspace from my PC, and one of my colleagues opened the same workspace simultaneously from another PC. I changed the settings of the workspace and saved it. My colleague saved the workspace after me. I opened the workspace again and found that the settings of the workspace differed from those I had made.*

The settings saved last are enforced. Once HEW opens a workspace, it is updated inside the memory. HEW does not save the settings into a file until a user saves the workspace intentionally.

Appendix B Regular Expressions

The HEW editor allows you to include special characters in search strings when performing a find or replace operation. These characters are listed in table B.1 and are detailed in the following pages.

Table B.1: Regular Expression Characters

Character	Function
?	Matches any single character (except a newline)
*	Matches any number of occurrences (0 or more) of any character except a newline
\n	Matches a newline character
\t	Matches a tab character
[]	Matches any one character or range listed within the brackets
\	Overrides any following regular expression character

- Symbol:** ?

Meaning: This character matches any single character, except the newline character.

Example: t?p matches “top”, “tip” but not “trap”.
- Symbol:** *

Meaning: This character matches any number of occurrences (0 or more) of any character except a newline. Thus, this character will not match across new lines. The * character will match as few occurrences as are necessary to make the rest of the pattern match.

Example 1: t*o matches the “to” of “too”, the “tro” of “trowel” and the “ty o” of “sporty orange” but not “smart orange” because the * character does not match across a new line.
- Symbol:** \n

Meaning: This character matches the newline character.

\n would be used to search for line endings or in patterns that cross line boundaries.

Example 1: ;\n matches every occurrence of a newline following a semicolon

Example 2: ;\nif searches for a semicolon, a new line and a line beginning with “if”.

- **Symbol:** `\t`
Meaning: This character matches the tab character.
Example 1: `\t8`
Finds every occurrence of a tab character followed by an 8.
Example 2: `init\t`
Finds every occurrence of a tab character following “init”.
- **Symbol:** `[]`
Meaning: This matches any one character or a range of single characters listed within the brackets. Brackets cannot be nested.
`[-]` specifies a range of characters e.g. `[a-z]` or `[0-9]`. The beginning character in the range must have a lower ASCII value than the ending character of the range.
`[~]` matches a single character if it is not any one of the characters between `[~` and `]`. This pattern also matches newline characters, unless the newline character is included within the brackets.
Example 1: `[AEIOU]`
Finds every uppercase vowel.
Example 2: `[<>?]`
Finds a literal `<`, `>`, or `?`.
Example 3: `[A-Za-z0-9_]`
Matches an upper or lowercase letter, a digit or an underscore.
Example 4: `[~0-9]`
Matches any character except a digit.
Example 5: `[\t\n]`
Matches a space, a tab or newline.
Example 6: `[\]]`
Matches a literal `]` if `]` is placed after `\`.
- **Symbol:** `\`
Meaning: This is the regular expression override character. If the character following the backslash is a regular expression character, it is treated as a normal character. The backslash is ignored if it is followed by a normal (non-regular expression) character.
Example 1: `*`
Searches for every occurrence of an asterisk.
Example 2: `\\`
Searches for every occurrence of a backslash.

Appendix C Placeholders

This appendix describes how to use the placeholders, a feature provided by several of the HEW components.

C.1 What is a Placeholder?

A placeholder is a special string, inserted into text, which is replaced at some subsequent time for the actual value. For example, one of the HEW placeholders is \$(FULLFILE) which represents a file with a full path. Suppose that you have an editor in `c:\myedit\myeditor.exe`, which can take the file to edit as a parameter. When invoking the editor the following shortcut could be made, e.g.:

```
c:\myedit\myeditor.exe c:\files\file1.c
```

if you wanted to open FILE1.C from the directory `c:\files`. However, what happens if you want the HEW to open any file through this editor? The problem is that the command above is specific to “`c:\files\file1.c`”. What we want to be able to do is to tell the HEW to use the editor specified but to open the file that you have chosen at that time. To do this, you can replace the specific name of the file with a general placeholder, i.e.:

```
c:\myedit\myeditor.exe $(FULLFILE)
```

Now whenever the HEW launches the editor with a file, it knows that it has to replace \$(FULLFILE) with the file you have selected.

C.2 Inserting a Placeholder

Placeholders can only be entered into three specific edit fields within the HEW (figures C.1, C.2 and C.3). There are four ways a placeholder can be entered:

In the first example, place the insertion cursor at the point you would like to insert the placeholder and then select the required placeholder from the pop-up menu to the right of the edit field.

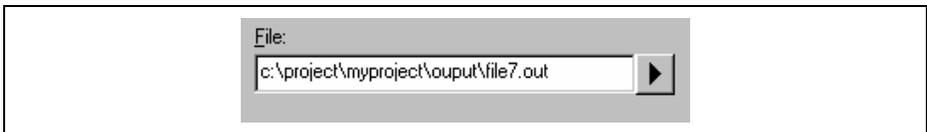


Figure C.1 Placeholder Pop-up Menu

In the second example, select the required placeholder other than “Custom directory” from the combo box and specify a sub-directory relative to the directory shown by the placeholder. If you select “Custom directory”, specify an absolute directory path in the “Sub-Directory” field.

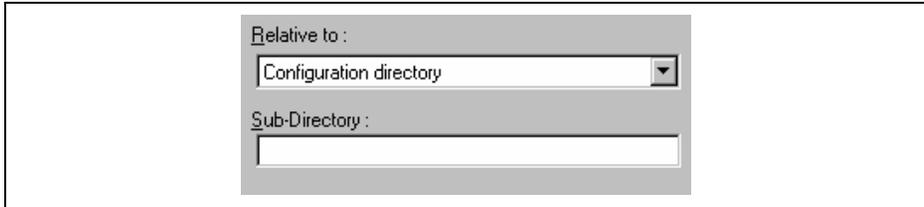


Figure C.2 Placeholder Combo Box and Sub-Directory Field

In the third example, place the insertion cursor at the point you would like to insert the placeholder, select the required placeholder from the combo box and then click the “Insert” button.



Figure C.3 Placeholder Combo Box

In the fourth example, type the placeholder into the field directly. Ensure that you type the placeholder name in uppercase and that it is preceded by \$(and followed by), i.e.

This is correct:

\$(FILEDIR)

These are incorrect:

\$(filedir)

\$(FILEDIR)

\$FILEDIR

C.3 Available Placeholders

Table C.1 lists the placeholders and their meanings.

Table C.1: Placeholders

Placeholder	Meaning
\$(FULLFILE)	Filename (including full path)
\$(FILEDIR)	File directory
\$(FILENAME)	Filename (excluding path and including extension)
\$(FILELEAF)	Filename (excluding path and extension)
\$(EXTENSION)	File extension
\$(WORKSPDIR)	Workspace directory
\$(WORKSPNAME)	Workspace name
\$(PROJDIR)	Project directory
\$(PROJECTNAME)	Project name
\$(CONFIGDIR)	Configuration directory
\$(CONFIGNAME)	Configuration name
\$(HEWDIR)	HEW installation directory
\$(TCINSTALL)	Toolchain install directory (on option dialog)
\$(TOOLDIR)	Tool installation directory (on Tools Administration)
\$(TEMPDIR)	Temp directory
\$(WINDIR)	Windows® directory
\$(WINSYSDIR)	Windows® system directory
\$(EXEDIR)	Command directory
\$(USERNAME)	User login (version control)
\$(PASSWORD)	User password (version control)
\$(VCDIR)	“Virtual” version control directory
\$(COMMENT)	Comment (version control)
\$(LINE)	Line number of an error/warning

For example, the placeholders will be expanded as shown in table C.2.

Table C.2: Placeholder Expansions (Example)

Placeholder	Expanded placeholder (example)
\$(FULLFILE)	c:\hew\workspace\project\file.src
\$(FILEDIR)	c:\hew\workspace\project
\$(FILENAME)	file.src
\$(FILELEAF)	file
\$(EXTENSION)	src
\$(WORKSPDIR)	c:\hew\workspace
\$(WORKSPNAME)	workspace
\$(PROJDIR)	c:\hew\workspace\project
\$(PROJECTNAME)	project
\$(CONFIGDIR)	c:\hew\workspace\project\debug
\$(CONFIGNAME)	debug
\$(HEWDIR)	c:\hew
\$(TCINSTALL)	c:\hew\toolchains\Hitachi\sh\511
\$(TOOLDIR)	c:\hew\toolchains\Hitachi\sh\511
\$(TEMPDIR)	c:\Temp
\$(WINDIR)	c:\Windows
\$(WINSYSDIR)	c:\Windows\System
\$(EXEDIR)	v:\vc\win32
\$(USERNAME)	JHARK
\$(PASSWORD)	214436
\$(VCDIR)	"c:\project" is mapped to "x:\vc\project"
\$(COMMENT)	"Please Enter Comment" dialog is invoked
\$(LINE)	12

In table C.2, we are assuming that

- a file path is "c:\hew\workspace\project\file.src".
- a workspace named "workspace" is located at "c:\hew\workspace".
- a project named "project" is located at "c:\hew\workspace\project".
- a configuration named "debug" has a configuration directory located at "c:\hew\workspace\project\debug".
- HEW.EXE is installed in "c:\hew".

- a *.HRF file of a toolchain (i.e. compiler, assembler, linker) is located at “c:\hew\toolchain\Hitachi\sh\511”. This is referred to as \$(TCINSTALL) on the option setting dialogs of the **[Options]** menu and as \$(TOOLDIR) on the “Tools Administration” dialog.
- the Windows® operating system is installed in “c:\Windows” and the Windows® system directory is “c:\Windows\System”.
- a version control executable path is “v:\vc\win32\ss.exe”, a user name and its password to login the version control system are “JHARK” and “214436” respectively, \$(COMMENT) is specified in a command line to the version control executable, and “c:\project” is mapped to “x:\vc\project” on the “Projects” tab of the “Version Control Setup” dialog, which is invoked via **[Tools->Version Control->Configure...]**.
- an error of compiler or assembler occurred at line 12.

Note: Not all of the placeholders are relevant in every field. For example, the \$(LINE) placeholder has no meaning when specifying a dependent files location. \$(USERNAME), \$(PASSWORD), \$(VCDIR), and \$(COMMENT) placeholders are acceptable only in version control. If you enter a placeholder into an edit field where it is not acceptable, you might be informed.

C.4 Placeholder Tips

Placeholders are there to allow you to create flexible paths to the various files used by the system.

- If there is a placeholder pop-up menu (▶) next to an edit field into which you are about to enter a path or file, you should consider how you can use a placeholder to make that path or file definition flexible.
- If you use several configurations, then the \$(CONFIGDIR) placeholder is very useful to ensure that files can be written to and from the current configuration's directory.
- Wherever possible, use a placeholder. They can always be removed or added later so don't be afraid to experiment.

Appendix D I/O File Format

HEW formats the [IO] window based on information it finds in an I/O Register definition file. When you select a debugging platform, HEW will look for a “<device>.IO” file corresponding to the selected device and load it if it exists. This file is a formatted text file that describes the I/O modules and the address and size of their registers. You can edit this file, with a text editor, to add support for memory mapped registers or peripherals you may have specific to your application (e.g. registers in an ASIC device mapped into the microcomputer's address space).

The following describes two formats of the “<device>.IO” file that supports or not the bit field.

D.1 File Format (Bit Field Not Supported)

Each module name must be defined in the [Modules] definition section and the numbering of each module must be sequential. Each module corresponds to a register definition section and within the section each entry defines an I/O register.

The [BaseAddress] definition is for devices where the location of I/O registers moves in the address space depending on the CPU mode. In this case, the [BaseAddress] value is the base address of the I/O registers in one specific mode and the addresses used in the register definitions are the address locations of the registers in the same mode. When the I/O register file is actually used, the [BaseAddress] value is subtracted from the defined register address and the resultant offset added to the relevant base address for the selected mode.

The [Register] definition entry is entered in the format <name> = <address> [<size> [<absolute>]].

1. <name> register name to be displayed.
2. <address> address of the register.
3. <size> which may be B, W or L for byte, word, or longword (default is byte).
4. <absolute> which can be set to A if the register is at an absolute address. This is only relevant if the I/O area address range moves about on the CPU in different modes. In this case, if a register is defined as absolute the base address offset calculation is not performed and the specified address is used directly.

Comment lines are allowed and must start with a “;” character.

An example is shown below.

Comment	Example:
Module definition	; H8S/2655 Series I/O Register Definitions File [Modules] BaseAddress=0 Module1=Power_Down_Mode_Registers Module2=DMA_Channel_Common Module3=DMA_Channel_0 ... Module42=Bus_Controller Module43=System_Control Module44=Interrupt_Controller ...
Register definition	[DMA_Channel_Common] DMAWER=0xffff00 B A DMATCR=0xffff01 B A DMACR0A=0xffff02 B A DMACR0B=0xffff03 B A DMACR1A=0xffff04 B A DMACR1B=0xffff05 B A DMABCRH=0xffff06 B A DMABCRL=0xffff07 B A ...
Register name	[DMA_Channel_0] MAR0AH=0xfffee0 W A MAR0AL=0xfffee2 W A IOAR0A=0xfffee4 W A ETCR0A=0xfffee6 W A MAR0BH=0xfffee8 W A MAR0BL=0xfffeea W A IOAR0B=0xfffeec W A ETCR0B=0xfffeee W A
Address	
Size	
Absolute address flag	

D.2 File Format (Bit Field Supported)

Each module name must be defined in the [Modules] definition section and the numbering of each module must be sequential. Each module corresponds to a register definition section and within the section each entry defines an I/O register.

The user must define “FileVersion=2” at the start of the section. It means that this I/O register file is described with the version that supports the bit field.

The [BaseAddress] definition is for devices where the location of I/O registers moves in the address space depending on the CPU mode. In this case, the [BaseAddress] value is the base address of the I/O registers in one specific mode and the addresses used in the register definitions are the address locations of the registers in the same mode. When the I/O register file is actually used, the [BaseAddress] value is subtracted from the defined register address and the resultant offset added to the relevant base address for the selected mode.

Each module has a section that defines the registers forming it along with an optional dependency. The dependency is checked to see if the module is enabled or not. Each register name must be defined in the section and the numbering of each register must be sequential. The dependency is entered in the section as dep=<reg> <bit> <value>.

1. <reg> is the register id of the dependency.
2. <bit> is the bit position within the register.
3. <value> is the value that the bit must be for the module to be enabled.

The [Register] definition entry is entered in the format id=<name> <address> [<size> [<absolute>[<format>[<bitfields>]]]].

1. <name> register name to be displayed.
2. <address> address of the register.
3. <size> which may be B, W or L for byte, word, or longword (default is byte).
4. <absolute> which can be set to A if the register is at an absolute address. This is only relevant if the I/O area address range moves about on the CPU in different modes. In this case, if a register is defined as absolute the base address offset calculation is not performed and the specified address is used directly.
5. <format> format for register output. Valid values are H for Hexadecimal, D for decimal, and B for binary.

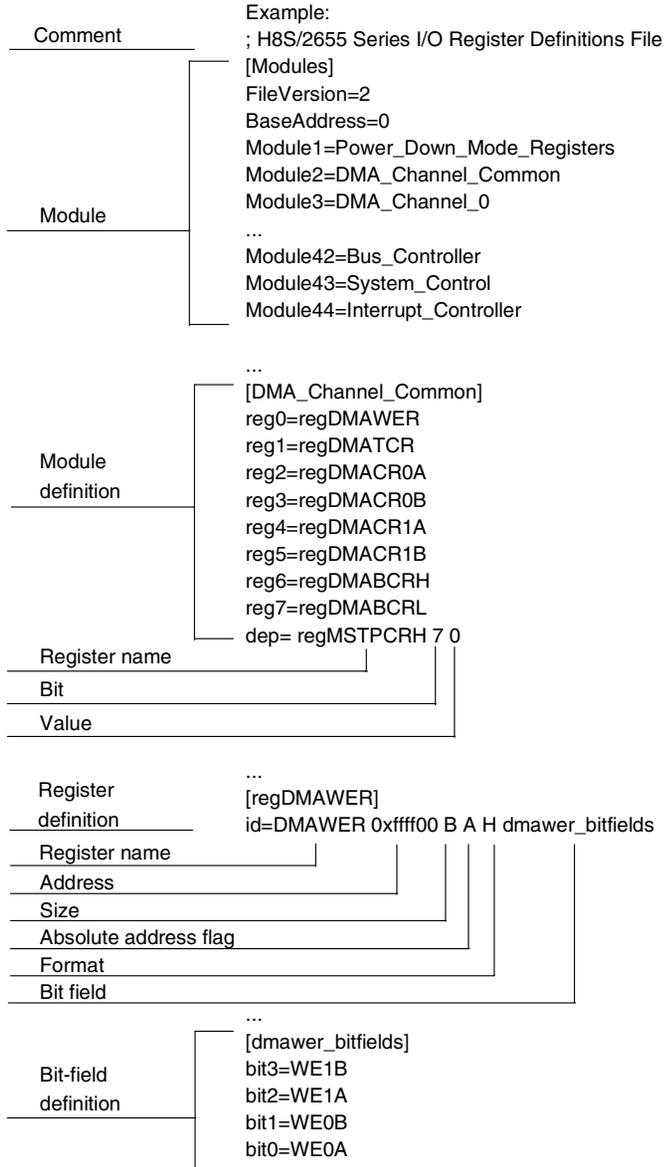
6. <bitfields> section defining the bits within the register.

Bitfield sections define the bits within a register each entry is of the type bit<no>=<name>.

1. <no> is the bit number.
2. <name> is a symbolic name of the bit.

Comment lines are allowed and must start with a “;” character.

An example is shown below.



Appendix E Symbol File Format

To decode the symbol file correctly, the file must be formatted as a Pentica-B file:

1. The file must be a plain ASCII text file.
2. The file must start with the word "BEGIN".
3. Each symbol must be on a separate line with the value first, in hexadecimal terminated by an "H", followed by a space then the symbol text.
4. The file must end with the word "END".

Example:

```
BEGIN
11FAH Symbol_name_1
11FCH Symbol_name_2
11FEH Symbol_name_3
1200H Symbol_name_4
END
```


Appendix F Menus

Table F.1 shows GUI menus.

Table F.1 GUI Menus

Menu	Option	Shortcut	Toolbar Button	Remarks
View	Disassembly	Ctrl + D		Opens the [Disassembly] window.
	Command Line	Ctrl + L		Opens the [Command Line] window.
	Workspace	Alt + K		Opens the [Workspace] window.
	Output	Alt + U		Opens the [Output] window.
CPU	Registers	Ctrl + R		Opens the [Register] window.
	Memory...	Ctrl + M		Opens the [Memory] window.
	IO	Ctrl + I		Opens the [IO] window.
	Status	Ctrl + U		Opens the [Status] window.
	Cache	Shift + Ctrl + C		Opens the [Cache] window.
	TLB	Shift + Ctrl + X		Opens the [TLB] window.
Sym- bol	Labels	Shift + Ctrl + A		Opens the [Labels] window.
	Watch	Ctrl + W		Opens the [Watch] window.
	Locals	Shift + Ctrl + W		Opens the [Locals] window.
Code	Eventpoints	Ctrl + E		Opens the [Event] window.
	Trace	Ctrl + T		Opens the [Trace] window.
	Stack Trace	Ctrl + K		Opens the [Stack Trace] window.

Table F.1 GUI Menus (cont)

Menu	Option	Shortcut	Toolbar Button	Remarks
View (cont)	Graphic	Image...	Shift + Ctrl + G	 Opens the [Image] window.
		Waveform...	Shift + Ctrl + V	 Opens the [Waveform] window.
	Performance	Performance Analysis	Shift + Ctrl + P	 Opens the [Performance Analysis] window.
		Profile	Shift + Ctrl + F	
Options	Debug Sessions...			Opens the [Debug Sessions] dialog box to list, add, or remove the debug session.
	Debug Settings...			Opens the [Debug Settings] dialog box to set the debugging conditions or download modules.
	Radix	Hexadecimal		 Uses a hexadecimal for displaying a radix in which the numerical values will be displayed and entered by default.
		Decimal		 Uses a decimal for displaying a radix in which the numerical values will be displayed and entered by default.
		Octal		 Uses an octal for displaying a radix in which the numerical values will be displayed and entered by default.
		Binary		 Uses a binary for displaying a radix in which the numerical values will be displayed and entered by default.
	Emulator	System...		 Opens the [Configuration] dialog box allowing the user to modify the debugging platform settings.

Table F.1 GUI Menus (cont)

Menu	Option	Shortcut	Toolbar Button	Remarks
Debug	Reset CPU			Resets the target hardware and sets the PC to the reset vector address.
	Go	F5		Starts executing the user program at the current PC.
	Reset Go	Shift + F5		Resets the target microcomputer and executes the user program from the reset vector address.
	Go To Cursor			Starts executing the user program at the current PC until the PC reaches the address indicated by the current text cursor position.
	Set PC To Cursor			Sets the PC to the address at the row of the text cursor.
	Run...			Launches the [Run Program] dialog box allowing the user to enter the PC or PC breakpoint during executing the user program.
	Step In	F11		Executes a block of user program before breaking.
	Step Over	F10		Executes a block of user program before breaking. If a subroutine call is reached, then the subroutine will not be entered.
	Step Out	Shift + F11		Executes the user program to reach the end of the current function.
	Step...			Launches the [Step Program] dialog box allowing the user to modify the settings for stepping.

Table F.1 GUI Menus (cont)

Menu	Option	Shortcut	Toolbar Button	Remarks
Debug (cont)	Step Mode	Auto		Steps only one source line when the [Editor] window is active. When the [Disassembly] window is active, stepping is executed in a unit of assembly instructions.
		Assembly		Executes stepping in a unit of assembly instructions.
		Source		Steps only one source line.
	Halt Program	Esc		Stops the execution of the user program.
	Initialize			Disconnects the debugging platform and connects it again.
	Disconnect			Disconnects the debugging platform.
	Download Modules			Downloads the object program.
	Unload Modules			Unloads the object program.
Memory	Search...			Searches for the specified value from the specified memory area.
	Copy...			Copies the specified memory area to the specified address.
	Compare...			Compares the specified two memory areas.
	Fill...			Fills the specified value in the specified memory area.
	Refresh			Forces a manual update of the contents of all the [Memory] windows open.
	Configure Overlay...			Selects the target section group when the overlay function is used.

Appendix G Command-Line Functions

The emulator supports the commands that can be used in the command-line window.

For details, refer to the online help.

Appendix H Notes on HEW

1. Note on Moving Source File Position after Creating Load Module

When the source file is moved after creating the load module, the [Open] dialog box may be displayed to specify the source file during the debugging of the created load module. Select the corresponding source file and click the [Open] button.

2. Source-Level Execution

— Source file

Do not display source files that do not correspond to the load module in the program window. For a file having the same name as the source file that corresponds to the load module, only its addresses are displayed in the program window. The file cannot be operated in the program window.

— Step

Even standard C libraries are executed. To return to a higher-level function, enter Step Out. In a for statement or a while statement, executing a single step does not move execution to the next line. To move to the next line, execute two steps.

3. Operation During Accessing Files

Do not perform other operations during downloading the load module or saving in the [Verify Memory], [Save Memory], or [Trace] window because this will not allow correct file accessing to be performed.

4. Watch

— Local variables at optimization

Depending on the generated object code, local variables in a C source file that is compiled with the optimization option enabled will not be displayed correctly. Check the generated object code by displaying the [Disassembly] window.

If the allocation area of the specified local variable does not exist, displays as follows.

Example: The variable name is asc.

asc = ? - target error 2010 (xxxx)

— Variable name specification

When a name other than a variable name, such as a symbol name or function name, is specified, no data is displayed.

Example: The function name is main.

main =

5. Line Assembly

— Input radix

Regardless of the Radix setting, the default for line assembly input is decimal. Specify H' or 0x as the radix for a hexadecimal input.

6. Command Line Interface

— Batch file

To display the message “Not currently available” while executing a batch file, enter the sleep command. Adjust the sleep time length which differs according to the operating environment.

Example: To display “Not currently available” during memory_fill execution:

```
sleep d'3000
```

```
memory_fill 0 ffff 0
```

— File specification by commands

The current directory may be altered by file specifications in commands. It is recommended to use absolute paths are recommended to be used to specify the files in a command file so that the current directory alteration is not affected.

Example: FILE_LOAD C:\\Hew2\\Tools\\Renesas\\DebugComp\\Platform\\E10A\\Tutorial\\Tutorial\\Debug_SHxxx_E10A_SYSTEM\\tutorial.abs

7. Memory Save During User Program Execution

Do not execute memory save or verifying during user program execution.

8. Load of Motorola S-type Files

This HEW does not support Motorola S-type files with only the CR code (H'0D) at the end of each record. Load Motorola S-type files with the CR and LF codes (H'0D0A) at the end of each record.

9. Note on [Register] Window Operation During Program Execution

The register value cannot be changed in the [Register] window during program execution. Even if the changed value is displayed, the register contents are not changed actually.

10. Break Functions

— BREAKPOINT cancellation

When the contents of the BREAKPOINT address is modified during user program execution, the following message is displayed when the user program stops.

BREAKPOINT IS DELETED A=xxxxxxx

If the above message is displayed, cancel all BREAKPOINT settings with the [Delete All] or [Disable] button in the [Event] window.

11. Number of BREAKPOINT and [Stop At] Settings in the [Run...] Menu

The maximum number of BREAKPOINTS and [Stop At] settings allowed in the [Run...] menu is 255. Therefore, when 255 BREAKPOINTS are set, specification by [Stop At] in the [Run...] menu becomes invalid. Use the BREAKPOINTS and [Stop At] in the [Run...] menu with 255 or less total settings.

12. Note on RUN-TIME Display

The execution time of the user program displayed in the [Status] window is not a correct value since the timer in the host computer has been used.

13. Note on Displaying Timeout error

If Timeout error is displayed, the emulator cannot communicate with the target microcomputer. Turn off the emulator and the user system and connect the emulator again by using the HEW.

14. Note on Using the [Run Program] Dialog Box

When [Run...] is selected from the [Debug] menu to specify the stop address, there is the following note:

- When the breakpoint that has been set as Disable is specified as the stop address, note that the breakpoint becomes Enable when the user program stops.

15. Memory Access during User Program Execution

When a memory is accessed from the memory window, etc. during user program execution, the user program is resumed after it has stopped in the E10A emulator to access the memory. Therefore, realtime emulation cannot be performed.

The stopping time of the user program is as follows:

Environment:

Host computer: 800 MHz (Pentium® III)

SH7710: 66.67 MHz (system clock frequency)

When a one-byte memory is read from the command-line window, the stopping time will be about 32 ms.

16. Support of Double Float Format

In the following memory operations, the double float format is not supported:

- [Fill Memory] dialog box
- [Search Memory] dialog box
- MEMORY_FILL command

The [Format] specification in the [Copy Memory] dialog box is ignored. Memory is copied in a byte unit.

17. BREAKPOINT Setting for SLEEP Instruction

When a break is set for the SLEEP instruction, use the Break Condition not the BREAKPOINT.

18. Note on Session Save in the [Configuration] Dialog Box

The following settings are not saved as a session:

- JTAG clock in the [General] page
- Loading flash memory in the [Loading flash memory] page

19. Scrolling Window During User Program Execution

Do not scroll the [Memory] and [Disassembly] windows by dragging the scroll box during user program execution. This generates many memory reads causing the user program to stop execution until the memory reads have been completed.

20. Memory Test Function

This product does not support the memory test function, which is used by selecting [Test...] from the [Memory] menu.

**Renesas Microcomputer Development Environment System
User's Manual
SuperH™ Family E10A Emulator**

Publication Date: Rev.1.00, May 26, 2003

Rev.2.00, October 6, 2005

Published by: Sales Strategic Planning Div.

Renesas Technology Corp.

Edited by: Customer Support Department

Global Strategic Communication Div.

Renesas Solutions Corp.

Renesas Technology Corp. Sales Strategic Planning Div. Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan



RENESAS SALES OFFICES

<http://www.renesas.com>

Refer to "<http://www.renesas.com/en/network>" for the latest and detailed information.

Renesas Technology America, Inc.

450 Holger Way, San Jose, CA 95134-1368, U.S.A
Tel: <1> (408) 382-7500, Fax: <1> (408) 382-7501

Renesas Technology Europe Limited

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, United Kingdom
Tel: <44> (1628) 585-100, Fax: <44> (1628) 585-900

Renesas Technology Hong Kong Ltd.

7th Floor, North Tower, World Finance Centre, Harbour City, 1 Canton Road, Tsimshatsui, Kowloon, Hong Kong
Tel: <852> 2265-6688, Fax: <852> 2730-6071

Renesas Technology Taiwan Co., Ltd.

10th Floor, No.99, Fushing North Road, Taipei, Taiwan
Tel: <886> (2) 2715-2888, Fax: <886> (2) 2713-2999

Renesas Technology (Shanghai) Co., Ltd.

Unit2607 Ruijing Building, No.205 Maoming Road (S), Shanghai 200020, China
Tel: <86> (21) 6472-1001, Fax: <86> (21) 6415-2952

Renesas Technology Singapore Pte. Ltd.

1 Harbour Front Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: <65> 6213-0200, Fax: <65> 6278-8001

Renesas Technology Korea Co., Ltd.

Kukje Center Bldg. 18th Fl., 191, 2-ka, Hangang-ro, Yongsan-ku, Seoul 140-702, Korea
Tel: <82> 2-796-3115, Fax: <82> 2-796-2145

Renesas Technology Malaysia Sdn. Bhd.

Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No.18, Jalan Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: <603> 7955-9390, Fax: <603> 7955-9510

SuperH™ Family E10A Emulator User's Manual



Renesas Electronics Corporation

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

REJ10J1015-0200