To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

To all our customers

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

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

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

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

Cautions

Keep safety first in your circuit designs!

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

Notes regarding these materials

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

User's Manual

High-performance Embedded Workshop 2 (for Windows® 98/ME, Windows NT® 4.0, Windows® 2000 and Windows® XP) HEW Debugger

User's Manual

Renesas Microcomputer Development Environment System

Renesas Electronics
www.renesas.com

Rev.2.0 2002.06

Cautions

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

Trademarks

Microsoft, MS-DOS, Windows, Windows NT 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 Information

Product Code: S32HEWM
Version: 2.1

Copyright © Hitachi Micro Systems Europe Ltd. 2001. All rights reserved.
Copyright © Hitachi, Ltd. 2001. All rights reserved.

About This Manual

This manual describes how to use the Hitachi Embedded Workshop and the debugger functionality. It covers all aspects of the debug process from creating your debug workspace, loading the download modules and running your program. For information on the "look and feel" of the Hitachi Embedded Workshop or customizing the HEW environment please refer to the main HEW user manual.

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

Cautions ... iii

1. Overview ... 1
1.1 Workspaces, Projects and Files ... 1
1.2 Launching the HEW ... 2
1.3 Creating a New Workspace ... 3
1.4 Opening a Workspace ... 4
1.5 Saving a Workspace ... 5
1.6 Closing a Workspace ... 5
1.7 Using Old Workspaces ... 5
1.8 Exiting the HEW ... 5
1.9 Debugger Sessions ... 6
1.10 Selecting a Session ... 7
1.10.1 Adding and Deleting Sessions ... 8
1.10.2 Saving session information ... 10

2. Preparing to Debug ... 11
2.1 Compiling for Debug ... 11
2.2 Selecting a Debugging Platform ... 11
2.3 Configuring the Debugging Platform ... 13
2.3.1 Mapping ... 14
2.3.2 Memory Resource ... 15
2.3.3 Downloading a Program ... 16
2.3.4 Manual Download of Modules ... 18
2.3.5 Automatic Download of Modules ... 18
2.3.6 Unload modules ... 19
2.3.7 Looking at Registers ... 20
2.3.8 Expanding a Bit Register ... 20
2.3.9 Modifying Register Contents ... 21
2.3.10 Using Register Contents ... 22

3. Looking at Your Program ... 23
3.1 Viewing the Code ... 23
3.2 Viewing Assembly-Language Code ... 23
3.3 Modifying Assembly-Language Code ... 24
3.4 Looking at Labels ... 25
3.5 Listing Labels ... 25
3.6 Looking at a Specific Address ... 26
3.7 Looking at the Current Program Counter Address ... 26
3.8 Source address column ... 26
3.9 Debugger columns ... 28

4. Working with Memory ... 29
4.1 Looking at an Area of Memory ... 29
4.2 Displaying Data in Different Formats ... 30
4.3 Looking at a Different Area of Memory ... 30
4.4 Modifying Memory Contents ... 31
4.4.1 Selecting a Memory Range ... 31
4.4.2 Finding a Value in Memory ... 31
4.5 Filling an Area of Memory with a Value ... 32
4.5.1 Filling a Range ... 32

4.6 Copying an Area of Memory 33
4.7 Testing an Area of Memory 33
4.8 Saving and Verifying an Area of Memory 34
4.9 Looking at I/O Memory 34
4.9.1 Opening an IO Window 35
4.9.2 Expanding an I/O Register Display 35
4.9.3 Modifying I/O Register Contents 35

5. Executing Your Program 37
5.1 Running from Reset 37
5.2 Continuing Run 37
5.3 Running to the Cursor 37
5.4 Single Step 37
5.4.1 Stepping Into a Function 38
5.4.2 Stepping Over a Function Call 38
5.4.3 Stepping Out of a Function 38
5.5 Multiple Steps 39

6. Stopping Your Program 41
6.1 Halting Execution 41
6.2 Standard Breakpoints (PC Breakpoints) 41
6.2.1 Using the Breakpoint Dialog Box 42
6.2.2 Toggling PC Breakpoints 42

7. Elf/Dwarf2 Support 43
7.1 C/C++ Operators 43
7.2 C/C++ Expressions 43
7.3 Supporting Duplicate Labels 44
7.4 Selecting a Function 44
7.5 Deselecting a Function 44
7.6 Setting a Function 45
7.7 Overlay Function 45
7.8 Displaying Section Group 45
7.9 Tooltip Watch 46
7.10 Local Variables 47
7.11 Watch Items 48
7.11.1 Adding a Watch 48
7.11.2 Expanding a Watch 48
7.11.3 Editing a Watch Item's Value 49

8. Viewing the Profile Information 51
8.1 Stack Information Files 51
8.2 Profile Information Files 52
8.3 Loading Stack Information Files 53
8.4 Enabling the Profile 54
8.5 Specifying the measurement for profiling 54
8.6 Executing the Program and Checking the Results 54
8.6.1 List Tab 54
8.6.2 Tree Tab 54
8.6.3 Profile-Chart Window 55
8.7 Types and Purposes of Displayed Data 55
8.8 Creating Profile Information Files 57
8.9 Notes 57

Appendix A - I/O File Format 59
A.1 File format 59

Appendix B - Symbol File Format 61

1. Overview

This chapter describes the fundamental concepts of the Hitachi 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 Hitachi 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 though 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.

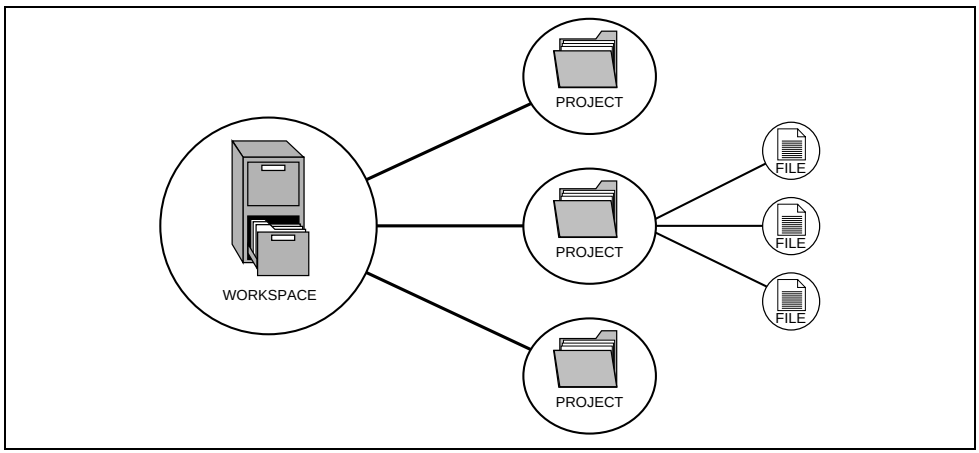


Figure 1.1: Workspaces, Projects and Files

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 Launching the HEW

To run the HEW, open the “Start” menu of Windows®, select “Programs”, select “Hitachi Embedded Workshop 2” and then select the shortcut of the Hitachi Embedded Workshop. By default, the Welcome! dialog shown in figure 1.2 will be displayed.

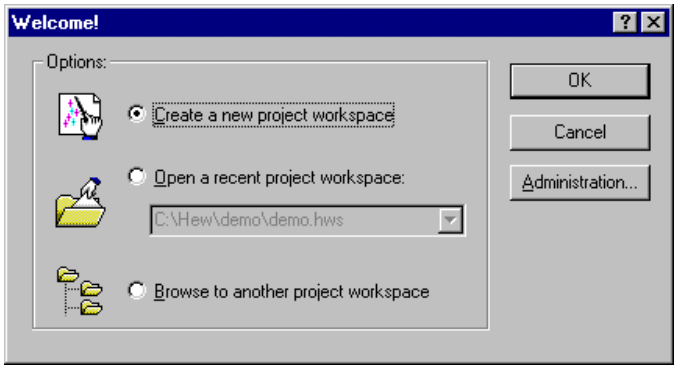


Figure 1.2: Welcome! Dialog

To create a new workspace, select the “Create a new project workspace” button, and click the “OK” button. To open one of recent project workspaces, select the “Open a recent project workspace” button, select a workspace from the drop-down list, and click the “OK” button. 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 the “Browse to another project workspace” button, and click the “OK” button. To register a tool to or unregister a tool from the HEW, click the “Administration…” button. Click the “Cancel” button to use the HEW without opening a workspace.

1.3 Creating a New Workspace

➲ To create a new workspace:

1. Select the “Create a new project workspace” option from the Welcome! dialog (figure 1.2) and press the “OK” button or select **[File->New Workspace...]**. The New Project Workspace dialog will be displayed (figure 1.3).

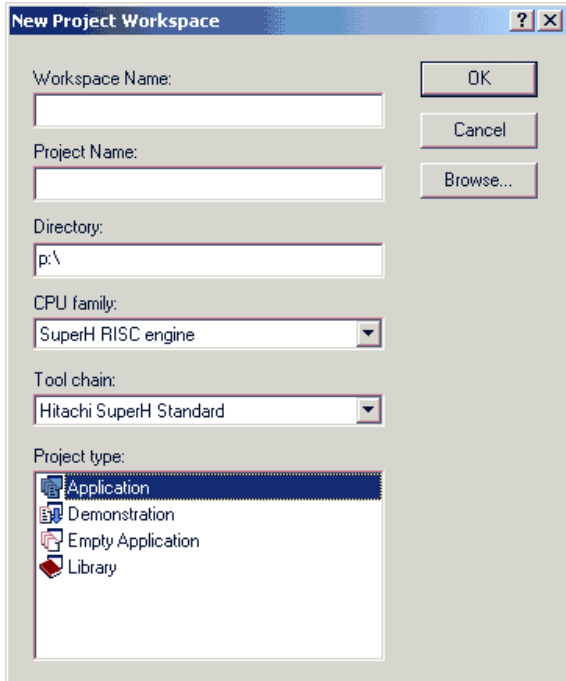


Figure 1.3: New Project Workspace Dialog

2. Enter the name of the new workspace into the “Workspace Name” field. This can be up to 32 characters in length and contain letters, numbers and the underscore character. As you enter the workspace name the HEW will add a subdirectory and project name for you automatically. This can be changed if desired. Use the “Browse…” button to graphically select the directory in which you would like to create the workspace. Alternatively, you can type the directory into the “Directory” field manually. This allows the workspace and project name to be different.
3. Select the CPU family and tool chain upon which you would like to base the workspace. Note that this cannot be changed once the workspace has been created.
4. When a new workspace is created, HEW will also create a project with the name specified in the project name field and place it inside the new workspace automatically. The “Project types” 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. The project types displayed will be all valid types for the current CPU family and tool chain pair. The project types are classified in three classes: tool chains only, debugger only or full project generator that configures both the debugger and tool chain aspect of HEW.
5. Click the “OK” button to create the new workspace and project.

Note: It is not possible to create a workspace if one already exists in the same directory.

1.4 Opening a Workspace

➲ To open a workspace:

1. Select “Browse to another project workspace” option from the Welcome! dialog (figure 1.2) and press the “OK” button or select **[File->Open Workspace…]**. The Open Project Workspace dialog will be invoked.
2. Select the workspace file that you want to open (.HWS files only).
3. Click the “Open” button to open the workspace.

   If the HEW is set-up to display information when a workspace is opened then a workspace properties dialog will be invoked (figure 1.4). Otherwise, the workspace will be opened.

   Note that whether the workspace properties dialog is shown depends on the setting of either the “Show workspace information on workspace open” check box on the workspace properties dialog or the “Display workspace information dialog on opening workspace” check box on the “Workspace” tab of the Tools Options dialog. Click the “OK” button to open the workspace. Click the “Cancel” button to stop opening the workspace.

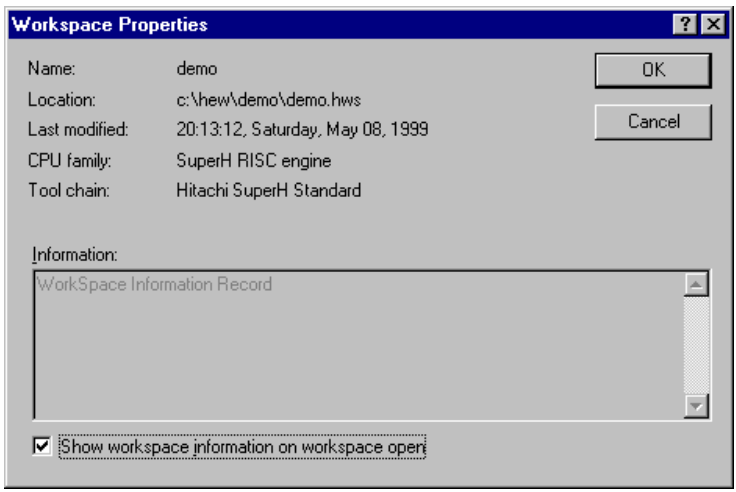


Figure 1.4: Workspace Properties Dialog

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

➲ To open a recently used workspace:

1. Select “Open a recent project workspace” from the Welcome! dialog, select the name of the workspace from the drop down list and then click the “OK” button.

or:

2. Select **[File->Recent Workspaces]** and from this sub-menu select the name of the workspace.

Note: The Hitachi Embedded Workshop only permits one workspace to be open at a time. Consequently, if you attempt to open a second workspace, the first will be closed before the new one is opened.

1.5 Saving a Workspace

Selecting **[File-> Save Workspace]** can save a HEW workspace.

1.6 Closing a Workspace

Selecting **[File-> Close Workspace]** can close a HEW workspace. If there are any outstanding changes to the workspace or any of its projects you will be requested whether or not you wish to save them.

Selecting **[File-> Save Workspace]** can save a HEW workspace.

1.7 Using Old Workspaces

The HEW can open any workspace that was created on a previous version of the HEW. This should not cause any problems and any differences in the workspace file details should be upgraded on the open. A back-up version of the initial workspace or project file should have been saved in the current directory of the file that is being upgraded. No upgrade path is available for session files used in the Hitachi Debugging Interface.

1.8 Exiting the HEW

The HEW can be exited by selecting **[File->Exit]**, pressing the **ALT+F4** accelerator 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.

1.9 Debugger Sessions

The HEW allows you to store all of your builder options into a configuration. This means that you can “freeze” all of the options and give them a name. In a similar way HEW allows you to store your debugger options in a session. Later on, you can select the session and all of the debugger options will be restored. These sessions allow the user to specify target, download modules and debug options. This means that potentially each session can be targeted at a different end platform.

This facility can allow you to have many different sessions each with different debugger options defined. For example it is possible to have each session using the same target but with slight variations in the session options. This can mean it is very easy for you to switch session and modify such things as register values or target settings such as clock speed. For example 2 sessions named Session1 and Session 2 both share the same target but the sessions can be subtlety different with regard to the options defined. This means that both sessions can share the same download module and avoid unnecessary code rebuilds. This is because sessions are not directly related to the build configuration data.

Each session’s data is stored in a separate file to the HEW project. You can then manipulate the data to share or modify as is required in the project. This is described further below:

1.10 Selecting a Session

The current session can be set in two ways.

➲ To select a session:

Either:

1. Select it from the drop down list box (figure 1.5) in the toolbar.

Figure 1.5: Toolbar Selection

or:

1. Select **[Options->Debug Sessions…]**. This will invoke the "Debug Sessions" dialog box (figure 1.6).

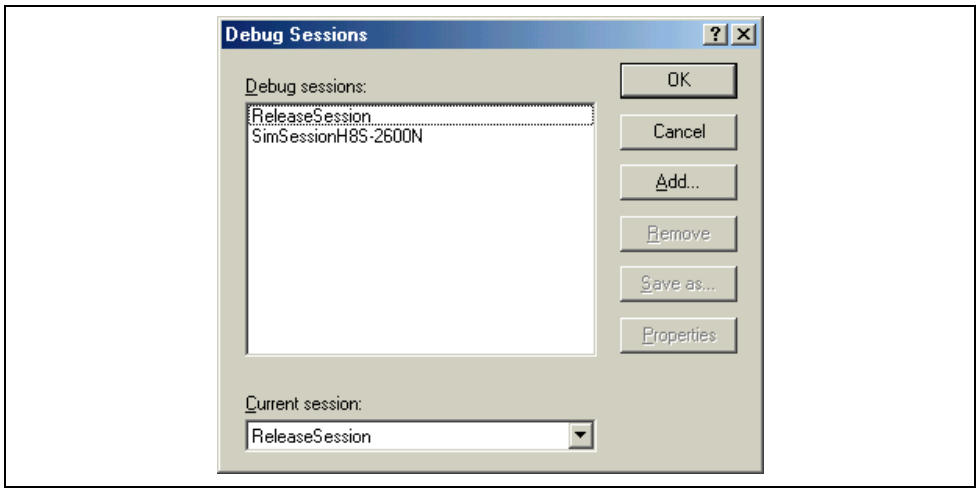


Figure 1.6: Debug Sessions Dialog

2. Select the session that you want to use from the "Current session" drop down list.
3. Click "OK" to set the session.

1.10.1 Adding and Deleting Sessions

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

➲ To add a new empty session:

1. Select **[Options->Debug Sessions…]** to display the “Debug Sessions” dialog box (figure 1.6).
2. Click the “Add…” button. The “Add Session” dialog box will be invoked (figure 1.7).
3. Click the “Add new session” radio button.
4. Enter a name for the session.
5. Click “OK” to close the “Debug Sessions” dialog box.
6. This operation creates a new empty session in the project directory. It uses the session name as its new file name. If the file name already exists an error is displayed.

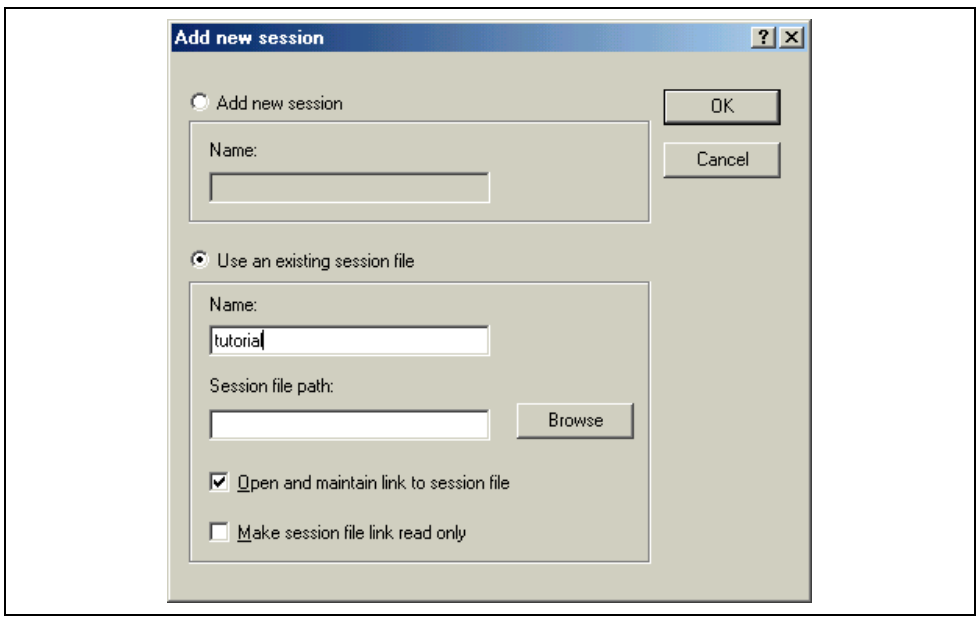


Figure 1.7: Add new sessions Dialog

➲ To import a link to an existing session file:

1. Select **[Options->Debug Sessions…]** to display the “Debug Sessions” dialog box (figure 1.6).
2. Click the “Add…” button. The “Add new session” dialog box will be invoked (figure 1.7).
3. Click the “Use an existing session file” radio button.
4. Enter a name for the session.
5. Browse to an existing session file location, which you would like to import into the current project.
6. Click the “Open and maintain link to session file” check box. This means the session will not be imported into the project directory but instead the HEW will link to the session location
7. Click “OK” to close the “Debug Sessions” dialog box.

➲ To remove a session:

1. Select **[Options->Debug Sessions…]** to display the “Debug Sessions” dialog box (figure 1.6).
2. Select the session you would like to modify.
3. Click the “Remove” button.
4. Note it is not possible to remove the current session.
5. Click “OK” 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 1.6).

2.Select the session you would like to view the properties for.

3.Click the properties button. The properties dialog is displayed. (Figure 1.8).

4.Click “OK” to close the “Debug Sessions” dialog box.

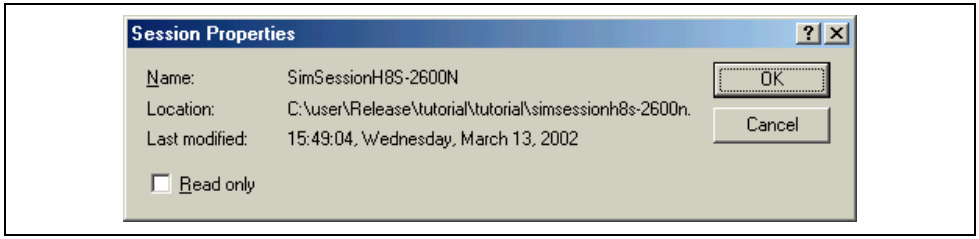


Figure 1.8: Session Properties Dialog

➲ To make a session read only:

1.Select [Options->Debug Sessions…] to display the “Debug Sessions” dialog box (figure 1.6).

2.Select the session you would like to view the properties for.

3.Click the properties button. The properties dialog is displayed. (Figure 1.8).

4.Click the read only check box. This makes 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 “OK”.

➲ To save a session as a different name:

1.Select [Options->Debug Sessions…] to display the “Debug Sessions” dialog box (figure 1.7).

2.Select the session you would like to save.

3.Click the Save as button. The Save as dialog is displayed. (Figure 1.9).

4.Browse to the new file location.

5.If you only want to export the session file to another location then leave the “Maintain link” checkbox unchecked. If you would like HEW to use this location instead of the current session location then check the “Maintain link” check box.

6.Click “OK”.

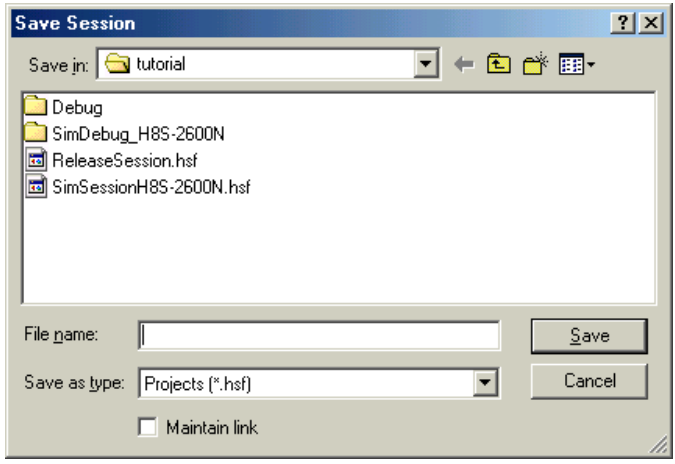


Figure 1.10: Save Session Dialog

1.10.2 Saving session information

➲ To save a session:

1.Select the **[File->Save Session]**.

2. Preparing to Debug

This section of the manual describes all the facilities that are available in the debugger for setting up the target platform to start debugging your program. You will learn how to select and configure a debugging platform with which to debug, and how to load your debug object file.

2.1 Compiling for Debug

In order to be able to debug your program at C/C++ source level, your C/C++ program must be compiled and linked with the debug option enabled. When this option is enabled, the compiler puts all the information necessary for debugging your C/C++ code into the absolute file or management information file, which are then usually called *debug object files*. When you create your project the initial setup will normally configure for debug.

Note Make sure you have the debug option enabled on your compiler and linker, when you generate an object file for debugging.

If your debug object file does not contain any debugging information (for example, the S-Record format), then you can still load it into the debugging platform, but you will only be able to debug at the assembly-language level.

2.2 Selecting a Debugging Platform

Selecting the debugging platform is very dependent on the installation of the HEW. If the HEW has a toolchain installed then the application project generator will be able to setup both the toolchain and the debugger targets simultaneously. This allows the options for targets and toolchain to be matched closely so no inconsistencies occur. If there is no toolchain installed you will be only be able to select debug only project types. By default HEW will display a debug only project generation type for each CPU family in the new workspace dialog. This project type will provide similar behavior to the previous HDI session dialog box.

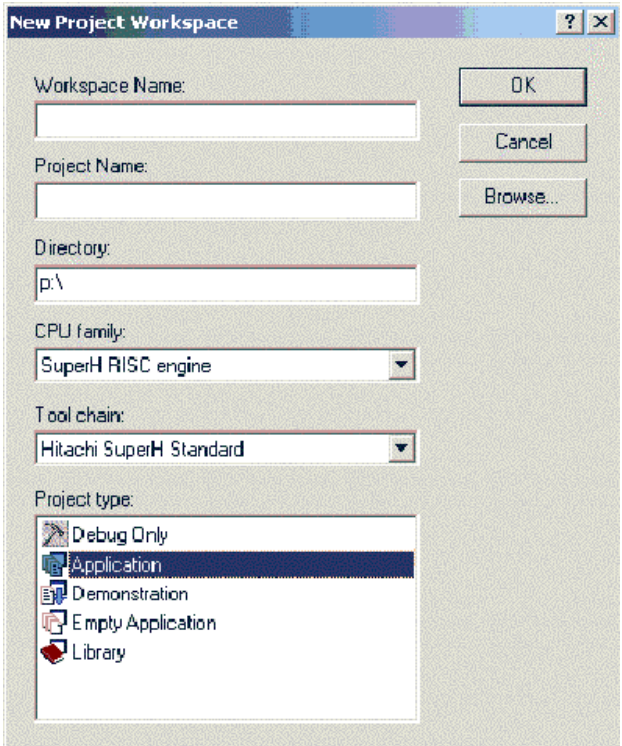


Figure 2.1: New Workspace Dialog

The dialog in figure 2.1 allows you to select a project type for generation, which matches your CPU target.

➲ To create a debug only project using the standard debug project type:

1. Select **[File->New Workspace…]**. The dialog shown in figure2.1 is displayed.
2. Select the “Debug Only” project type and click the “OK” button.
3. The “Debug Only” project wizard is displayed. The dialog shown in figure 2.2 is displayed.
4. It allows the user to select the specific target for his debug project. This is dependent on the selection made in the new workspace dialog. This page displays all valid debugger targets that are currently installed. The user can choose which platforms will be used for debugging in this new debug project. It is possible to select one or more targets and a debugger session will be created for each target.
5. Click the “Finish” button to compete the project creation process. A summary dialog will be displayed. Once completed the debugger project generator should create a project, which sets up the debug options correctly for each debugger session. The generator also creates a default configuration.

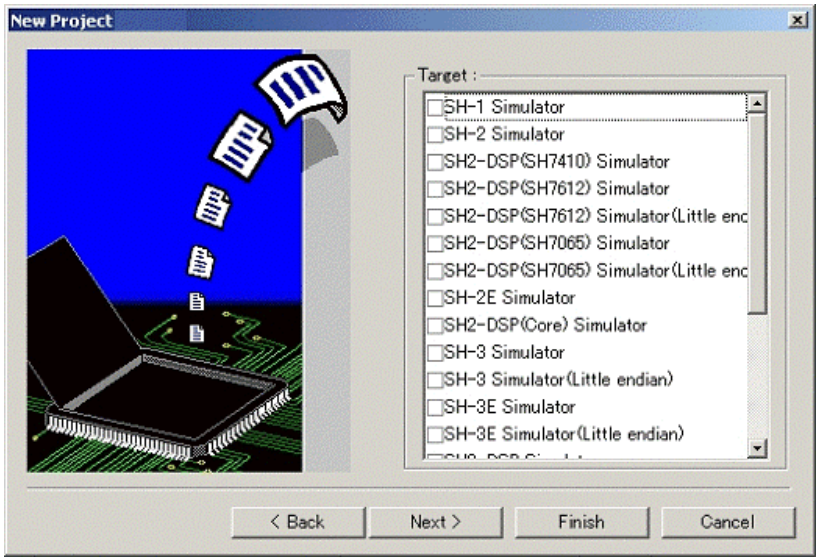


Figure 2.2: New Project Dialog

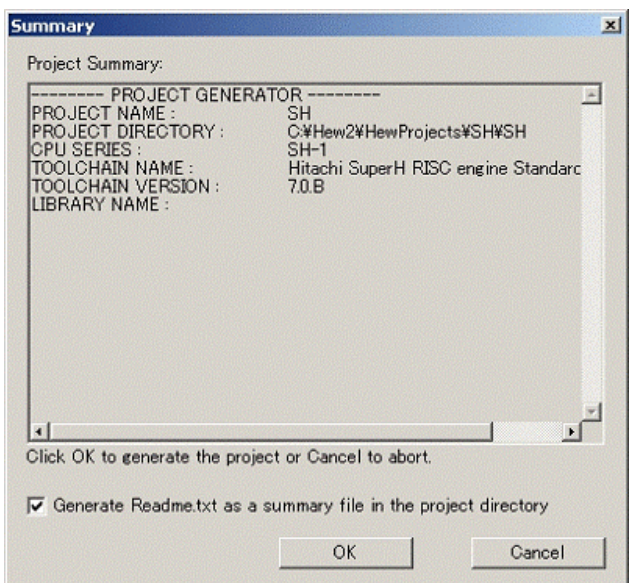


Figure 2.3: Summary Dialog

2.3 Configuring the Debugging Platform

Before you can load a program into your debugging platform you must set it up to match your application's system. The items that must be set-up are typically device type, operating mode, clock speed and the memory map. It is particularly important to set-up the memory map. In the Hitachi Embedded Workshop the project

generation process will have completed much of this work. However if you are using a different configuration of board from the standard types then some customization will be essential.

2.3.1 Mapping

For the debugger to correctly represent your user system, the memory map must be set up. It needs to know which areas in the device's address space are RAM, ROM, on-chip registers or areas where there is no memory.

When you select the device type and mode in the project generator, the HEW will automatically set up the map for that device and the mode in which the processor is operating. For example in a device with internal ROM and RAM, the areas where these are located in the device's memory map will be set by default.

If you are using a device that does not have internal memory, or a device with external memory instead of, or in addition to, the internal memory, then you must tell the debugging platform that you have memory there.

When using the simulator, the memory map settings can be modified in the Simulator System dialog box. For details, refer to the manual for the debugging platform.

In this case, the settings can be referred to in the Simulator pane of the Standard Toolchain dialog box. Clicking the “View” button of the Standard Toolchain dialog box (figure 2.4) displays the CPU hardware map dialog box (figure 2.5).

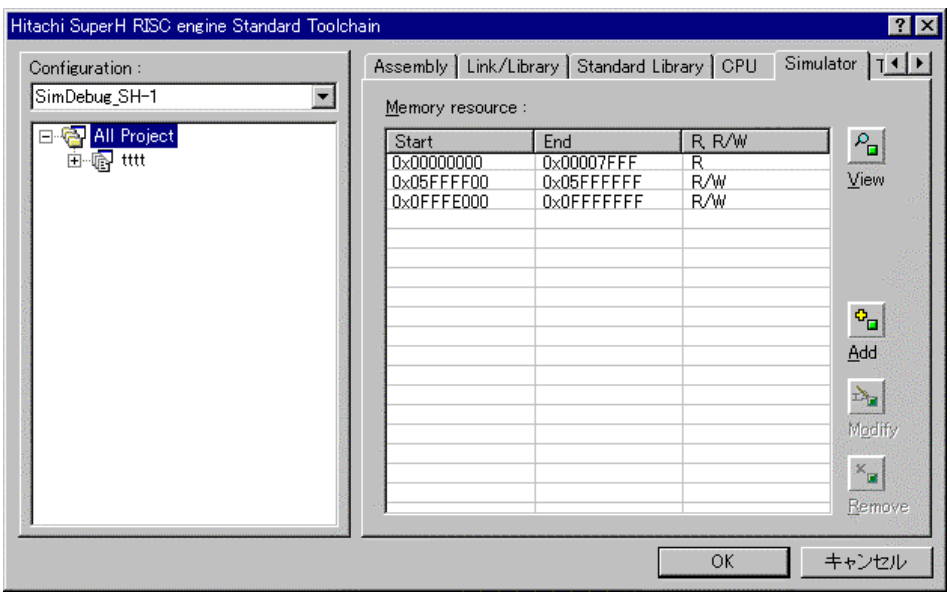


Figure 2.4: Standard Toolchain Dialog

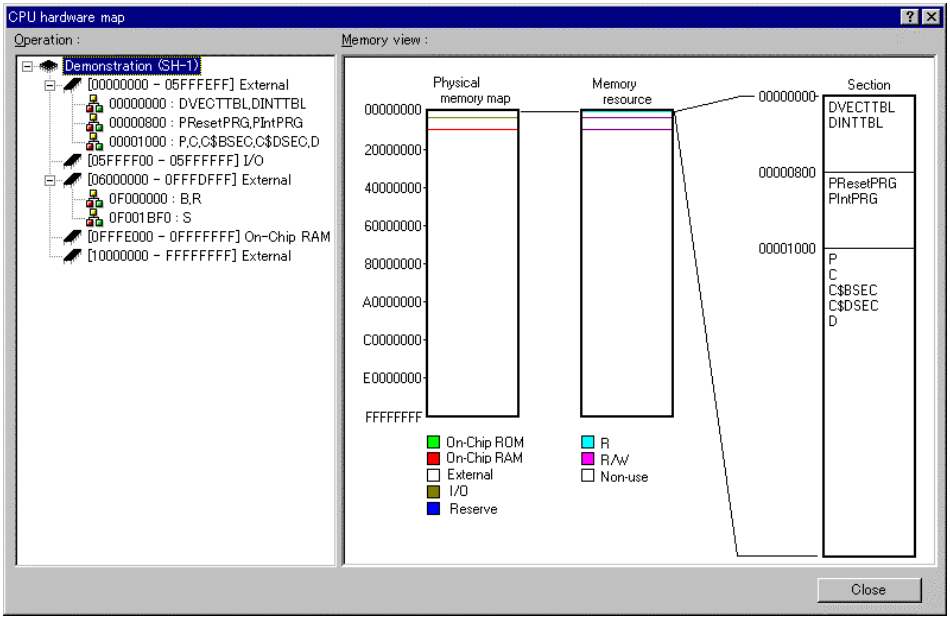


Figure 2.5: CPU hardware map Dialog

The following information is displayed:

[Operation] Displays the address range

[Memory View] Displays memory map and memory resource information

2.3.2 Memory Resource

To use the simulator, the memory resource of the address range being used must be saved. The memory resource settings can be modified in the Simulator Memory Resource dialog box. For details, refer to the manual for the debugging platform.

In this case, the settings can be referred to in the Simulator pane of the Standard Toolchain dialog box (figure 2.6). The Simulator pane contains the following buttons:

[View] Displays the memory map information

[Add] Adds memory resource

[Modify] Modifies the selected memory resource

[Remove] Removes the selected memory resource

Clicking the “Add” button displays the Add memory resource dialog box (figure 2.7).

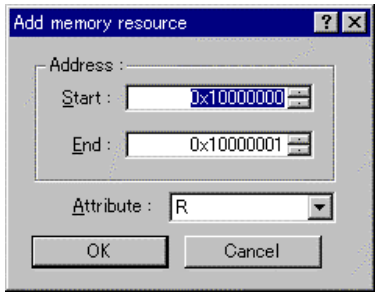


Figure 2.7: Add memory resource Dialog

Specify the following three items in the Add memory resource dialog box:

[Start] Start address

[End] End address

[Attribute] Attributes (R: Read only, R/W: Readable/Writable)

Modification of the memory resource information in the Simulator Memory Resource dialog box will be reflected in the contents of the Standard Toolchain dialog box, and modification in the Standard Toolchain dialog box will be reflected in the contents of the Simulator Memory Resource dialog box as well.

2.3.3 Downloading a Program

Once you have made sure that there is memory in your system in which to download your code, you can then proceed to download a program to debug. The initial selection of download module is automatic with regard to an application generator, as it is the output from the linker. However with regard to the debug only project generator it is possible for you to select the module which you wish to download.

It is also possible to manually choose download modules after the project creation. This is achieved via the debug settings dialog shown in figure 2.9. This dialog allows you to control the debug settings throughout your workspace. The tree on the left of the dialog contains all of the current projects. Selecting a project in this tree will then show you the settings for that project and the session selection in the session drop list. It is possible to select multiple sessions in this list box or all sessions. If you select multiple sessions you can choose to modify the settings for one or more sessions at once. The debug settings dialog displays the following debug options:

- Current debug target for the current project and session selection.
- Download modules for the current project and session selection.

The download module list displays the order in which the files will be downloaded to the target. It is possible to add, remove, modify, move up and move down modules in this list.

➲ To add a new download module:

1. Select **[Options->Debug Settings…]**. The dialog shown in figure 2.9 will be displayed.
2. Select the project and configurations you are interested in adding a download module to in the tree control.
3. Click the “Add” button. The dialog displayed in figure 2.8 is displayed.
4. The “Format” list box contains a list of supported object format readers. This allows you to choose the correct reader for the download module you wish to add to the project.

5. The “File name” field allows you to browse to the download module on your disk, or simply type it into the edit control if you have not yet built the module.
6. Enter an offset address to the “Offset address” field (suitable only for some object formats).
7. When the user clicks Add the debug download module is added to the bottom of the list. If you wish to position the module in a different position in relation to the other modules, select the module and then use the move up and move down buttons to position the module correctly.

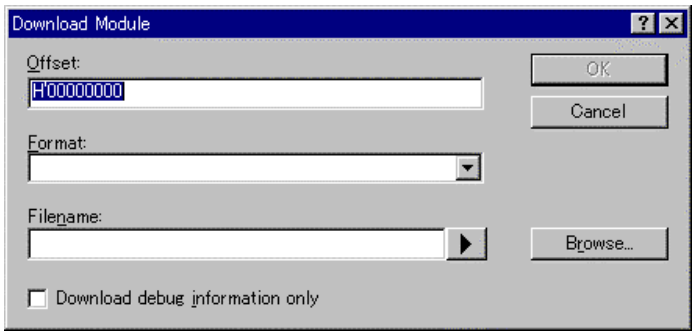


Figure 2.8: Download Module Dialog

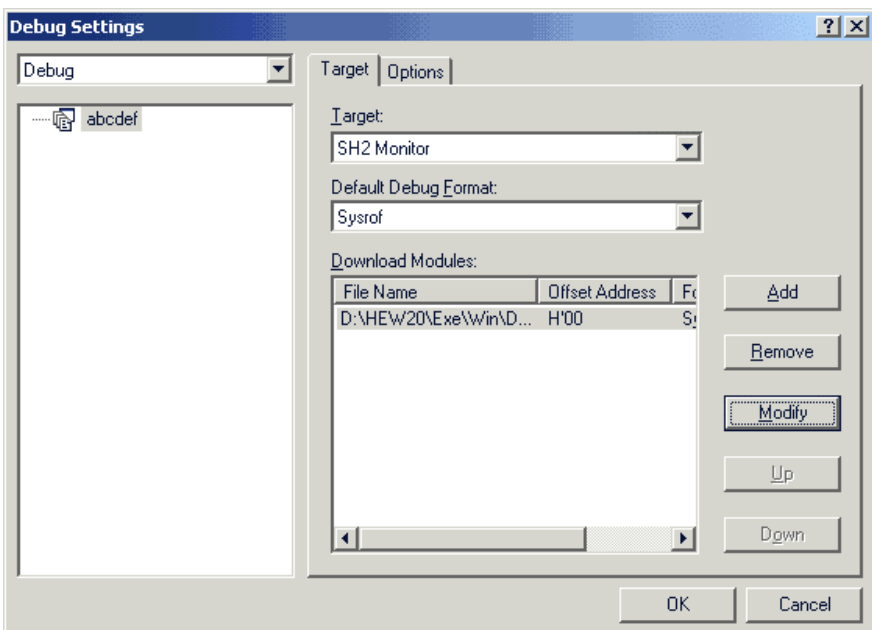


Figure 2.9: Debug Settings Dialog (“Target” Pane)

Any changes made in the debug settings dialog are only changed when you click the “OK” button. If the download modules are changed for the current target then the modules are flagged for download next time the program is run for debug. The default debug format is set to the first download module in the list by default. It is only possible to specify one default debug object format per session. All currently installed debugger formats are listed here.

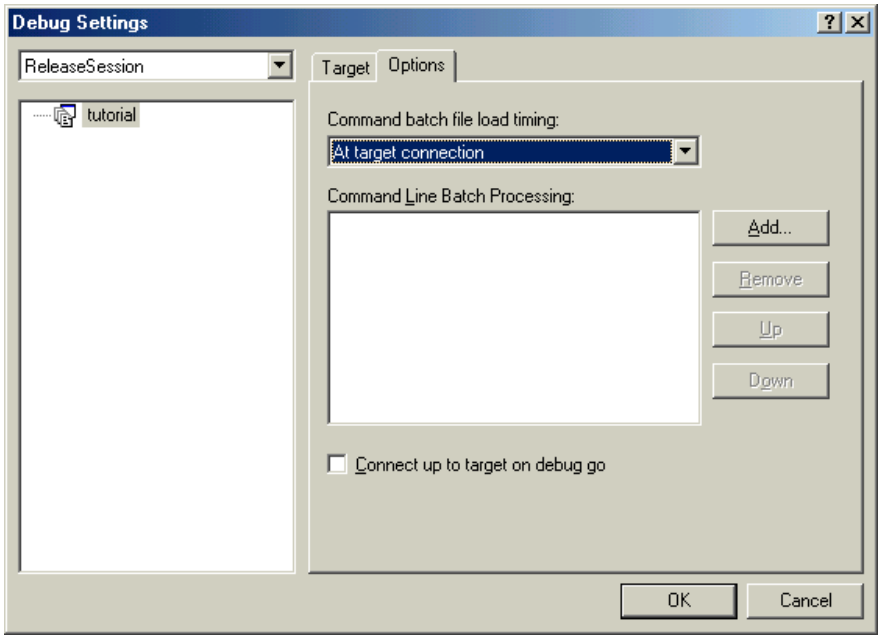


Figure 2.10: Debug Settings Dialog ("Options" Pane)

The options page of the debug settings dialog contains the option which determines when a target is connected. If the options are checked then the target is not connected until you select **[Build->Debug->Go]**. If the option is unchecked then the target is connected whenever a configuration is opened. This happens when a new workspace or project is opened and also when you switch configuration using the toolbar or configurations dialog.

The command line batch-processing list contains a list of command line batch commands. These batch commands can perform debug operations via the command line. It is possible to choose when these are executed in the timing drop list. Each drop list selection has a different list of batch files. So choosing a selection in the timing drop list alters which command line batch files you can see. The order in the list is the order the command line batch files will be executed when the timing event occurs.

2.3.4 Manual Download of Modules

Once you have decided which download modules are to be download to the target it is possible to manually update the modules on the connected target. This is achieved by selecting **[Debug -> Download Modules]**. This allows a single module or all of the modules to be downloaded. This can also be achieved by right clicking on a module in the workspace window under the download module folder.

2.3.5 Automatic Download of Modules

When you select **[Debug->Run]** the HEW calculates whether any of the file times or debugger settings, which affect the modules, have changed since the last download. If the HEW detects a change then it asks the user whether a download needs to take place for each module. If the user selects "Yes" then the modules are downloaded to the target.

If you have banked multiple modules at the same start address then by default only the first module at that address is downloaded. It is then possible to manually load and unload the other modules in the list whenever you want from the **[Debug -> Download Modules]** and the **[Debug -> Unload Modules]** menu items.

2.3.6 Unload modules

It is then possible to manually unload downloaded modules in the list whenever you want from the **[Debug -> Unload Modules]** menu item. This feature can also be accessed from the download module pop-up in the workspace window.

When a module is unloaded its symbols are lost from the HEW debugging system. Although the target remains unmodified. It will no longer be possible to debug this module once it is unloaded.

2.3.7 Looking at Registers

If you are debugging at assembly-language level, then you will probably find it useful to see the contents of the CPU's general registers. You can do this using the Registers window.

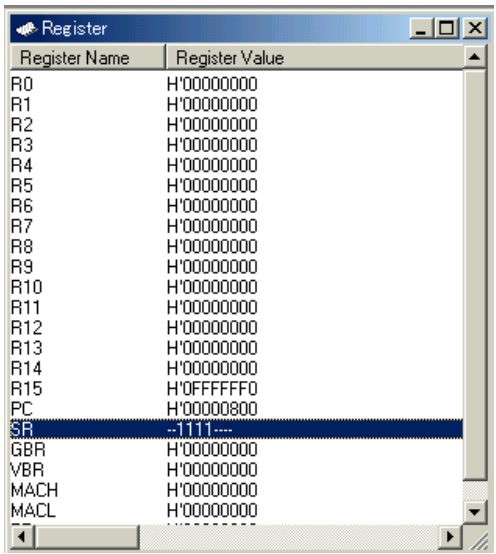


Figure 2.11: Registers Window

To open a Registers window choose **[View->Registers]** or click the Register Window toolbar button [ ]. A Registers window opens showing all of the CPU's general registers and the values, displayed in hexadecimal.

2.3.8 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 name 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 checkbox for any bit specifies it as holding a 1, while removing the check specifies it as a 0.

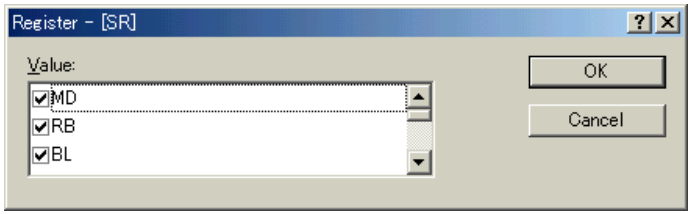


Figure 2.12: Expanding a Flag Register

To collapse an expanded flag register, double click on the minus sign. The flags collapse back to the single item and the minus sign changes back to a plus sign.

2.3.9 Modifying Register Contents

To change a register's contents open the Edit Register dialog box in one of these methods:

1. Double-click the register you want to change.
2. Select the register you want to change, and press **ENTER**.
3. Select the register you want to change, and choose **[Edit…]** from the popup menu.

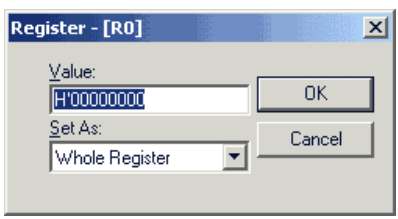


Figure 2.13: Register Dialog Box

As in any other data entry field in HEW 2.1, you can enter a formatted number or C/C++ expression. You can choose whether to modify the whole register contents, a masked area, floating or flag bits by selecting an option from the drop list 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.

2.3.10 Using Register Contents

Use the value contained in a CPU register by specifying the register name prefixed by the “#” character, e.g.: #R1, #PC, #R6L, or #ER3 when you are entering a value elsewhere in the HEW, for example when displaying a specified address in the Disassembly or Memory windows.

3. Looking at Your Program

This section describes how to look at your program as source code and assembly language mnemonics. The HEW has various facilities for dealing with code and symbol information which are explained in this section and you will be shown how to look at text files in the user interface.

Note: After a break occurs the HEW displays the location of the program counter (PC). In some cases, for example if a SYSROF based project is moved from its original path, then the source files may not be automatically found. In this case the HEW will open a source file browser dialog to allow you to manually locate the file - this path will then be used to update any other source files in this debug project.

3.1 Viewing the Code

Select your source file and click the **"Open"** button, the HEW opens the file in the integrated editor. It is also possible to view your source files by double clicking on them in the workspace window.

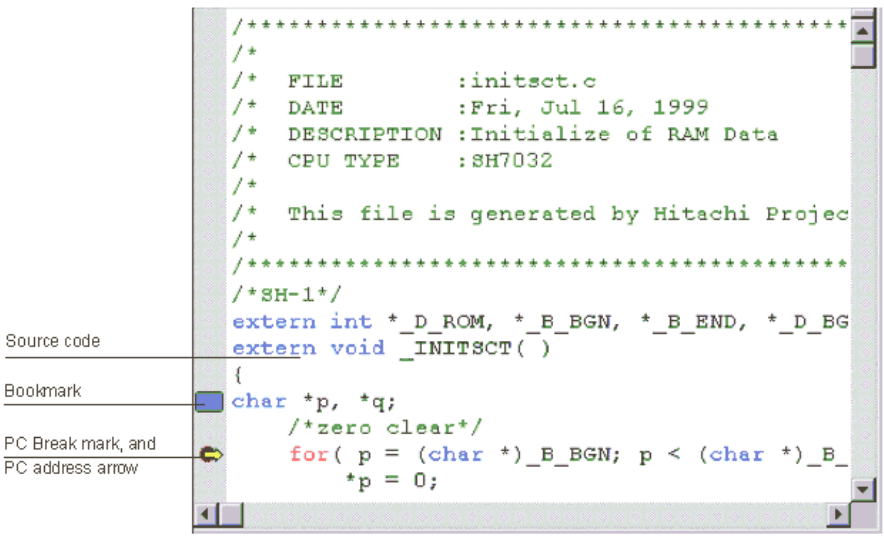


Figure 3.1: Source Window

The editor is divided into two areas: the gutter area (containing markers for breakpoints, PC location, etc.) and the text area (containing color syntax highlighted code). This is shown above in figure 3.1.

3.2 Viewing Assembly-Language Code

If you have a source file open, right-click to open the popup menu and select [View disassembly] to open a Disassembly window at the same address as the current Source window.

If you do not have a source file, but wish to view code at assembly-language level, either choose **[View->Disassembly]**; use the **Ctrl+D** accelerator; or click on the Disassembly window toolbar button [ ].

The Disassembly window opens at the current PC location, and shows Address and Code (optional) - showing the disassembled mnemonics (with labels when available). Optionally, any source line starting at that address may be shown, thus providing mixed mode display.

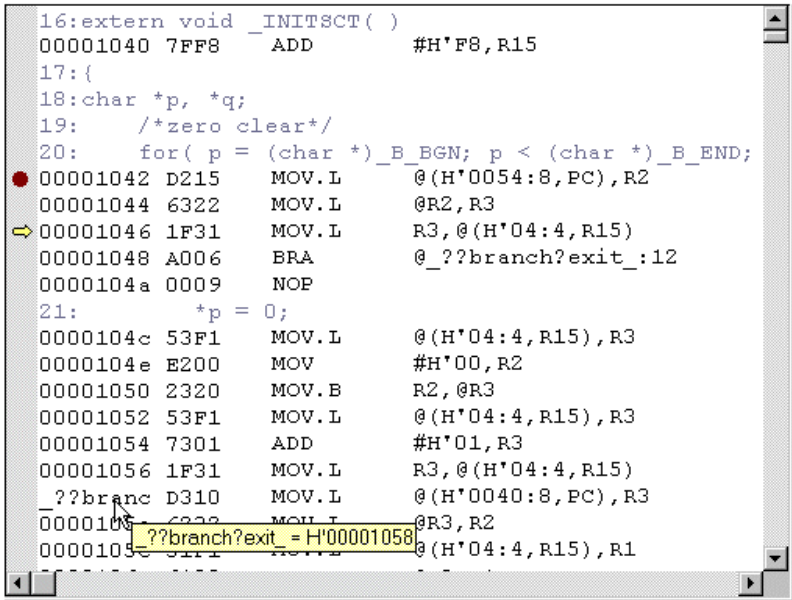


Figure 3.2: Disassembly Window

3.3 Modifying Assembly-Language Code

You can modify the assembly-language code by double clicking on the instruction that you wish to change. The Assembler dialog box will open:

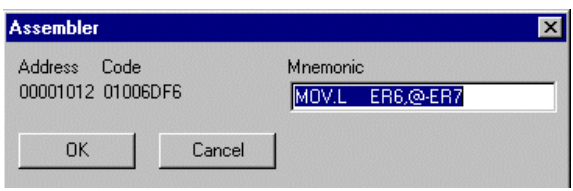


Figure 3.3: Assembler Dialog Box

The address, machine code and disassembled instruction are displayed. Type the new instruction or edit the old instruction in the “Mnemonics” field. Pressing the **ENTER** key will assemble the instruction into memory and move on to the next instruction. Clicking **“OK”** 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 Disassembly window) will show the new assembly-language code, but the source window will be unchanged. This is true even if the source file contains assembler.

3.4 Looking at Labels

In addition to the debugging information that the HEW uses to link your program's source code to the actual code in memory, the *debug object file* also contains symbolic information. This is a table of text names that represent an address in the program and is referred to as labels in HEW. In the Disassembly window, you will see the first 8 characters of the label in place of the corresponding address, and as part of an instruction's operand.

Note: An instruction's operand is replaced with a label name if the operand and label value match. If two or more labels have the same value, then the label that comes first alphabetically will be displayed.

Tip: Wherever you can enter an address or values in an edit control you can use a label instead.

3.5 Listing Labels

To see a list of all the labels defined in the current debugger session select **[View->Labels]**.

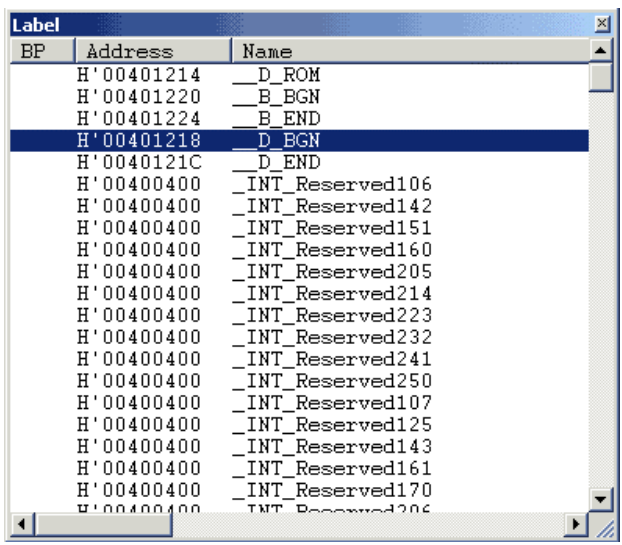


Figure 3.4: Labels Window

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

You can quickly toggle a software break at the entry point of a function by double-clicking in the BP column. Alternatively, right-click to show the popup menu and select **[Break]**.

3.6 Looking at a Specific Address

When you are looking at your program in a Disassembly window, you may want to look at another area of your program's code. Rather than scrolling through a lot of code in the program, you can go directly to a specific address. Select **[Set Address]** from the popup menu, and the dialog shown in figure 3.5 is displayed.

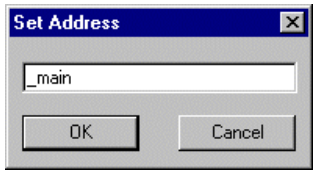


Figure 3.5: Set Address Dialog Box

Enter the address or label name in the edit box and either click on **"OK"** or press the **ENTER** key. The Disassembly window updates 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. This is detailed in section 7, Elf/Dwarf2 Support of this manual.

3.7 Looking at 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 Source or Disassembly window display will go to the current PC address. It also allows that you can display from an offset of the current PC by entering an expression with the PC register plus an offset, e.g., "#PC+0x100".

3.8 Source address column

Once your program has downloaded the source window updates and displays any addresses for the current source file. These are shown on the left-hand side of the source window. This feature is useful for deciding where you can set PC or hardware breakpoints. This is show below in figure 3.6.

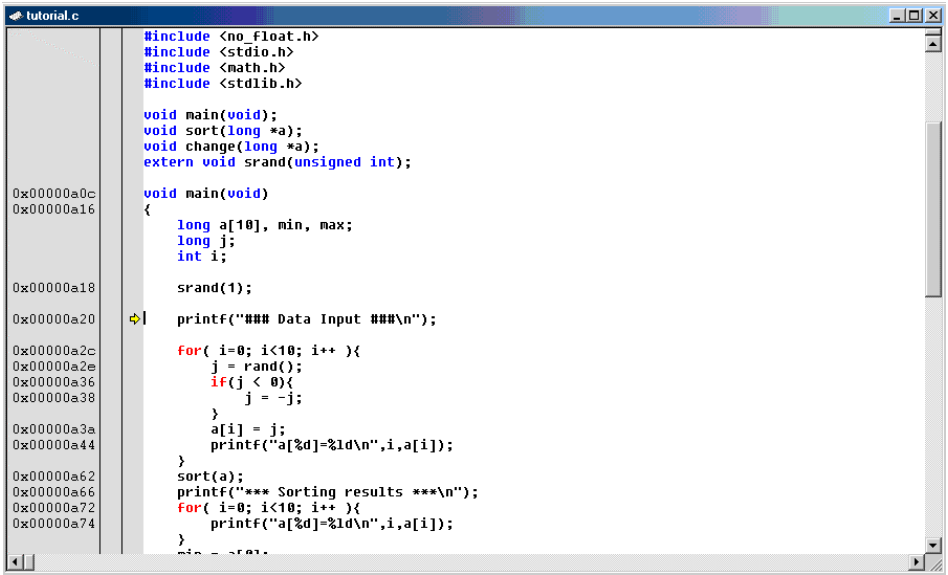


Figure 3.6: Source window and address column

- 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.

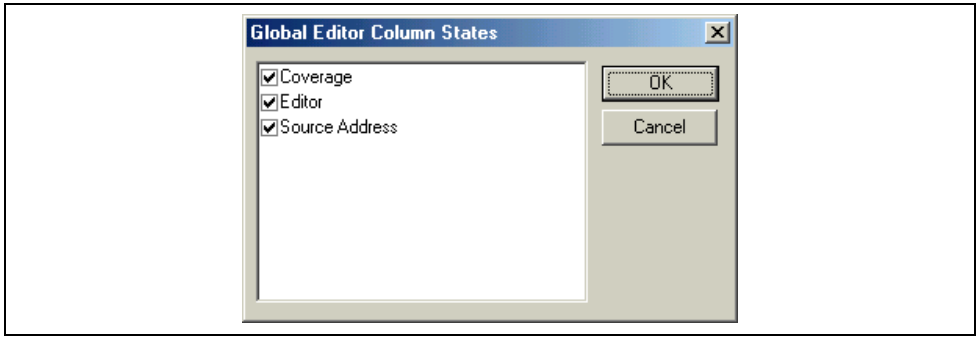


Figure 3.7: Global column state Dialog Box

3.9 Debugger columns

Any component can add columns to both the disassembly and source windows. This can change the look and feel of your Hitachi Embedded Workshop environment but very much depends on the installation you are running. Typical examples of columns are the coverage column, which graphically displays code coverage during debugger execution. Another example is the target component column, this can show the hardware breakpoints set to the target.

Right clicking on the column displays the respective pop-up menu for that column. This is different for every column and is managed by the individual component. Double clicking on the column also has a different effect depending on the column. For example in the editor and target column this sets a PC or hardware breakpoint.

4. Working with Memory

This section describes how to look at areas of memory in the CPU's address space. It will show you how to look at an area of memory in different formats, fill, move and test a block of memory, and save, load and verify an area of memory with a disk file.

4.1 Looking at an Area of Memory

To look at an area of memory, choose **[View->Memory…]**; using the **Ctrl+M** accelerator; or clicking the Memory Window toolbar button [ ] to open a Memory window. This will launch a **Set Address** dialog box shown in figure 4.1.

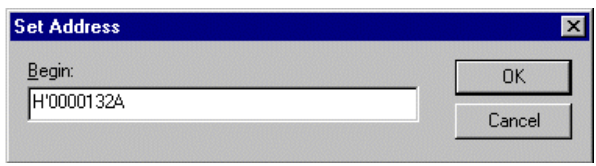


Figure 4.1: Set Address Dialog Box

Type in the start address or equivalent symbol for the address you want to see in the “Address” field. Click **“OK”** or press the **ENTER** key, and the dialog box closes and a Memory window opens:

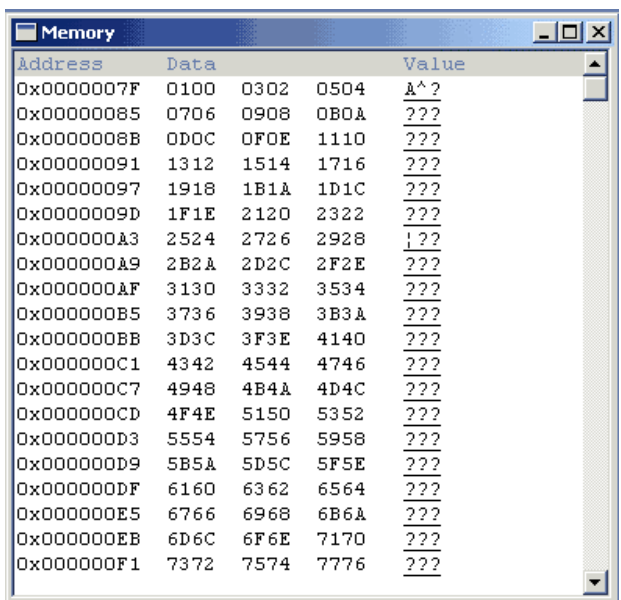


Figure 4.2: Memory Window

There are three display columns:

1. Address - The address of the first item in the Data column of this row.
2. Data - The data read from the debugging platform's physical memory at the access width and then converted to the displayed width.
3. Value - Data displayed in an alternative format.

4.2 Displaying Data in Different Formats

If you want to change the display format from the one you selected when you opened the view, selecting [Format] from the popup menu can do this. This dialog is displayed in figure 4.3.

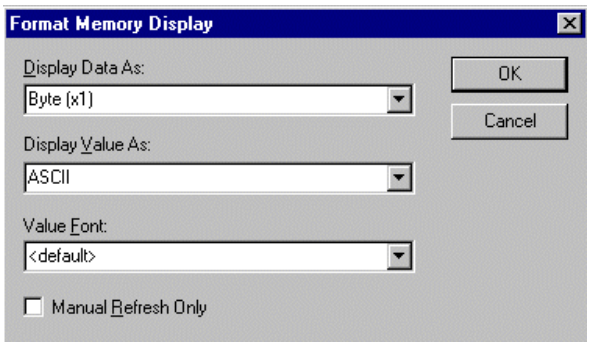


Figure 4.3: Format Memory Display Dialog Box

To display and edit memory in different widths, use the “Display Data As” drop list - for example, choose the **Byte** option and the display will be updated to show the area of memory as individual bytes.

The data can be interpreted into different formats - this is shown in the third column, “Value”. The list of formats depends on the data selection.

The font of the “Value” column can be different from the font used to display the data. This is useful for displaying double-byte character values when the data is displayed in Word format. The “<default>” selection will use the same font in the “Value” column as the rest of the window.

4.3 Looking at a Different Area of Memory

If you want to change the area of memory that the Memory window is displaying you can use the scroll bars. To quickly look at a new address you can use the Set Address dialog box. This can be opened either be choosing **[Start Address]** from the popup menu or by double-clicking in the “Address” column.

Enter the new address value, and click **“OK”** 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.

4.4 Modifying Memory Contents

To change the contents of memory is accessed via the Edit dialog box. Move the cursor on the memory unit (depending on your Memory window display choice) that you wish to change. Either double-click on the memory unit, or press the **ENTER** key. The dialog in figure 4.4 is displayed.

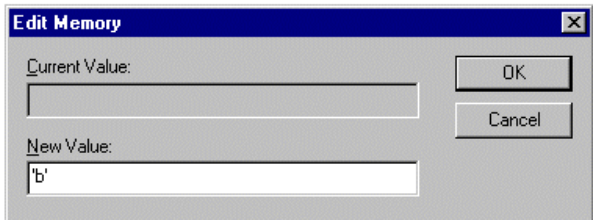


Figure 4.4: Edit Memory Dialog Box

Like any other data entry field in HEW, you can enter a formatted number or C/C++ expression. 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 memory.

4.4.1 Selecting a Memory Range

If the memory address range is in the Memory window, you can select the range by clicking on the first memory unit (depending on your 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, then you can enter the start addresses and byte count in the respective fields of the memory dialog boxes.

4.4.2 Finding a Value in Memory

To find a value in memory you must open a memory view, then choose **[Search]** from the popup menu. Alternatively, with a memory window in focus, just press **F3**.

The Search Memory dialog box is displayed as shown in figure 4.5.

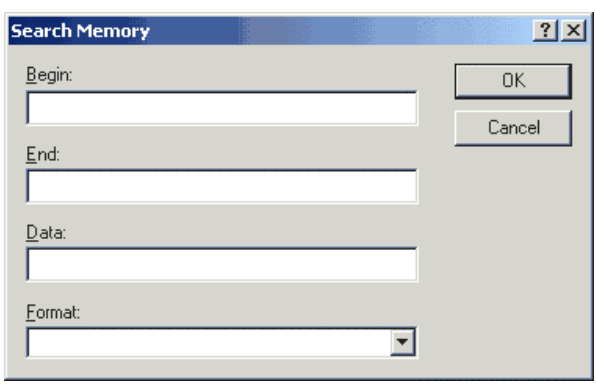


Figure 4.5: Search Memory Dialog Box

Enter the start and end addresses of the range in which to search (if an area of memory was selected in the Memory window then the Begin and End address values will be filled in automatically) and the data value to search for. The end address can also be prefixed by a '+' which will use the entered value as a range.

Select the search format (defaults to data display format) and click **"OK"** or press the **ENTER** key. The dialog closes and the HEW searches the range for the specified data. If the data is found, it will be highlighted in the Memory window, and a message showing the address at which the data has been found is displayed on the Status bar.

If the data cannot be found, the caret position in the Memory window remains unchanged and a message informing you that the data could not be found is displayed on the Status bar.

4.5 Filling an Area of Memory with a Value

You can set the contents of a range of memory addresses to a value using the memory fill feature.

4.5.1 Filling a Range

To fill a range of memory with the same value, choose **[Fill]** on a memory window's popup menu, or on the Memory drop-down menu. The Fill Memory dialog box and is shown in figure 4.6.

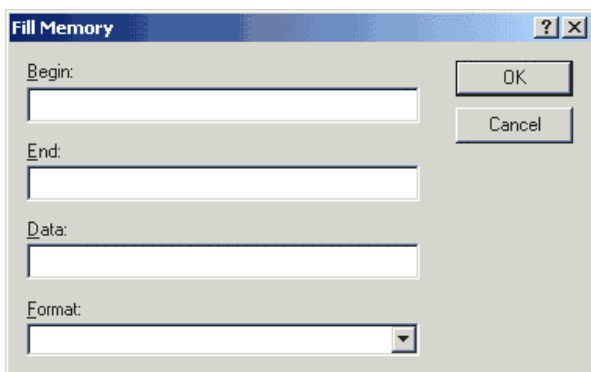


Figure 4.6: Fill Memory Dialog Box

If an address range has been selected in the Memory view, the specified start and end address will be displayed. Select the format from the "Format" drop list and enter the data value in the "Data" field. Click the **"OK"** button or press the **ENTER** key, the dialog box closes and the new value is written into the memory range.

4.6 Copying an Area of Memory

You can copy an area of memory using the memory copy feature. Select a memory range and then choose **[Copy…]** from the popup menu. The Copy Memory dialog box opens:

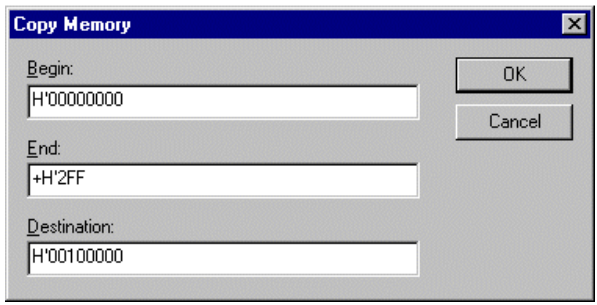


Figure 4.7: Copy Memory Dialog Box

The source start address and end address specified in the Memory window will be displayed in the “Begin” and “End” fields. Enter the destination start address in the “Destination” field and click the **“OK”** button or press the **ENTER** key, the dialog box closes and the memory block will be copied to the new address.

4.7 Testing an Area of Memory

Note: The exact test is target dependent. However, in all cases the current contents of the memory will be overwritten - YOUR PROGRAM OR DATA WILL BE ERASED. The Test Memory function is not supported by some debugging platforms.

You can test an area of memory in the address space using the memory test feature. Select a memory and choose **[Test]** from the popup menu. The Test Memory dialog box is displayed as shown in figure 4.8.

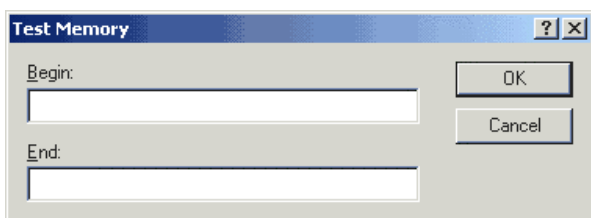


Figure 4.8: Test Memory Dialog Box

The start address and end address specified in the Memory view will be displayed in the “Begin” and “End” fields. Click the “OK” button or press the **ENTER** key, the dialog box closes and the HEW will perform a test on the memory range.

4.8 Saving and Verifying an Area of Memory

You can save an area of memory in the address space to a disk file using the save memory feature. Open the "Save Memory As" dialog box by choosing **[File->Save memory...]**:

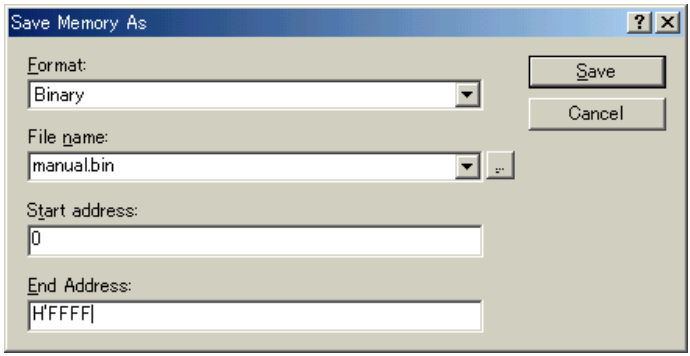


Figure 4.9: Save Memory As Dialog Box

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" drop-list contains the previous four file names used for saving memory, or clicking the "Browse…" button can launch a standard File Save As dialog. Click the "OK" button or press the **ENTER** key, the dialog box closes and the memory block will be saved to the disk as a Motorola S-record format file. When the file save is complete a confirmation message box may be displayed.

You can verify an area of memory in the address space using the verify memory feature. Open the Verify Memory dialog box by choosing **[File->Verify memory...]**:

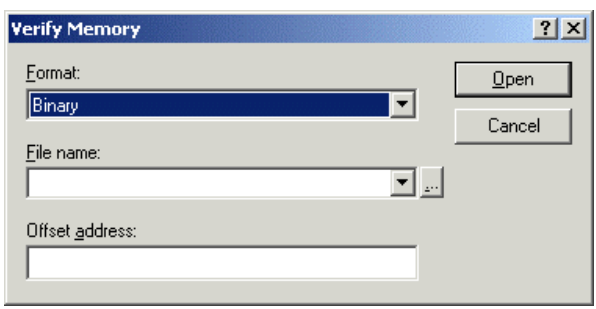


Figure 4.10: Verify Memory Dialog Box

4.9 Looking at I/O Memory

As well as a CPU and ROM/RAM, a micro-controller also contains on-chip peripheral modules. The exact number and type of peripheral modules differ between devices but typical modules are DMA controllers, serial

Communications interfaces, A/D converters, integrated timer units, a bus state controller and a watchdog timer. Accessing registers, which are mapped to the micro-controller's address space, programs the on-chip peripherals.

The Memory window only allows you to look at data in memory as byte, word, long word, single-precision floating-point, double precision floating-point, or ASCII values, so HEW also provides an IO window to ease inspection and setting up of these registers.

4.9.1 Opening an IO Window

To open an IO window select **[View->IO]** or click the IO Window toolbar button [ ]. Modules that match the on-chip peripherals organize the I/O register information. When an IO window is first opened, only a list of module names is displayed.

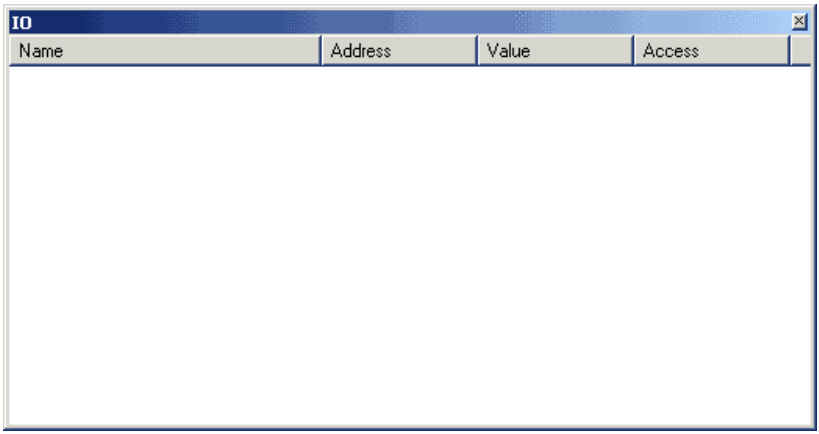


Figure 4.11: IO Window

4.9.2 Expanding an 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.

To display to bit level, expand the I/O register in a similar way.

The bits are color coded as follows:

Black	Normal read/write
Red	Value changed
Grey	Peripheral disabled (by peripheral control registers)

4.9.3 Modifying I/O Register Contents

To edit the value in an I/O register you can type hex 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.

5. Executing Your Program

This section describes how you can execute your program's code. You will learn how to do this by either running your program continuously or stepping single or multiple instructions at a time.

5.1 Running from Reset

To reset your user system and run your program from the *Reset Vector* address choose **[Debug->Reset Go]**, or click the Reset Go toolbar button [ ].

The program will run until it hits a breakpoint or a break condition is met. You can stop the program manually at any time 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 it is important to make sure that this location contains the address of your startup code.

5.2 Continuing Run

When your program is stopped and the debugger is in *break mode*, the HEW will display an yellow arrow mark in the gutter of the line in the editor and Disassembly windows 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 Continue toolbar button [ ], or choose **[Debug->Go]**.

5.3 Running to the Cursor

Sometimes as you are going through your application you may want to run only 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 a Source 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 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 Stop toolbar button [ ].
2. The Go To Cursor feature requires a temporary breakpoint - if you have already used all those available then the feature will not work, and the menu option will be disabled.

5.4 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 Source window, then 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.

5.4.1 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]**.

5.4.2 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]**.

5.4.3 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. Or alternatively you may have stepped into a function by accident, when you meant to step over it and so want to return to the calling function without stepping all the way through the current function. You can do this with the Step Out feature.

To step out of the current function either click the Step Out toolbar button [ ], or choose **[Debug->Step Out]**.

5.5 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:

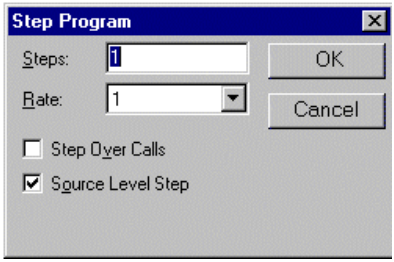


Figure 5.1: Step Program Dialog Box

Enter the number of steps in the “Steps” field and select whether you want to step over function calls by the “Step Over Calls” check box. If you are using the feature for automated stepping, select the step rate from the list in the “Rate” field. Click **“OK”** or press **ENTER** to start stepping.

6. 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 command and by setting breakpoints at specific locations in your code.

6.1 Halting Execution

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). To stop the program click on the Halt toolbar button, or choose **[Debug->Halt]**.

The last break cause can also be viewed in the “Debug” pane of the Output window.

6.2 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 to 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 supported by the target, it may provide a Breakpoints window.

➲ To set a program (PC) breakpoint

1. Make sure that the Disassemble or a source window is open at the place you want to set a (PC) breakpoint.
2. Select **[Toggle Breakpoint]** from the pop-up menu, or press **F9**, at the line showing the address at which you want the program to stop. You can also double click in the source gutter to set a (PC) breakpoint.
3. You will see a red circle appear in the gutter to indicate that a (PC) breakpoint has been set.
4. It is also possible to add, remove and edit the current breakpoint set up by using **[Edit->Breakpoints]**.

Now when you run your program and it reaches the address at which you set the (PC) breakpoint, execution halts with the message "Break = PC Break" displayed in the “Debug” pane of the Output window, and the Source or Disassembly window is updated with the program (PC) breakpoint line highlighted.

Note: The line or instruction at which you set a program (PC) breakpoint is not actually executed; the program stops just before it is about to execute it. If you choose to Go or Step after stopping at the (PC) breakpoint, then the highlighted line will be the next instruction to be executed.

6.2.1 Using the Breakpoint Dialog Box

The breakpoint dialog box is displayed in Figure 6.1. It allows you to view the current breakpoints set in the workspace and view the code associated with each. From this dialog box it is also possible to remove one or all breakpoints.

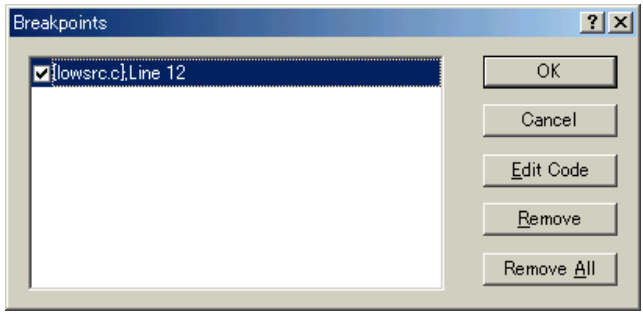


Figure 6.1: Breakpoints Dialog Box

6.2.2 Toggling PC Breakpoints

It is possible to toggle PC Breakpoints by either double clicking in the BP column of the line at which the (PC) breakpoint is set or placing the cursor on the line and using the **F9** key. The display will cycle through the available standard breakpoint types - a colored circle will be shown in the gutter.

7. Elf/Dwarf2 Support

The HEW supports the Elf/Dwarf2 object file format for debugging applications written in C/C++ and assembly language for Hitachi microcomputers. 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 defined.
- Stack Trace

7.1 C/C++ Operators

The C/C++ language operators are available:

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

```
Buffer_start + 0x1000
#R1 | B'10001101
((pointer + (2 * increment_size)) & H'FFFF0000) >> D'15
!(flag ^ #ER4)
```

7.2 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++)
```

7.3 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.

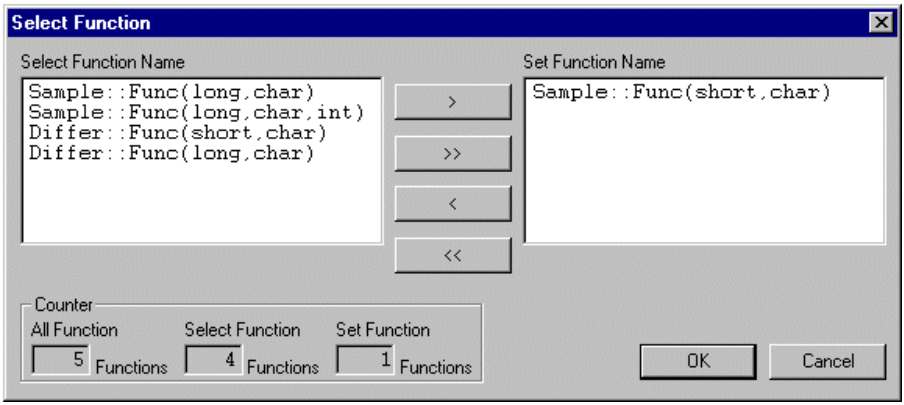


Figure 7.1: Select Function Dialog Box

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" list box

  Displays the same-name functions or member functions and their detailed information.

- "Set Function Name" list box

  Displays the function to be set and their detailed information.

- "Counter group" edit box

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.

7.4 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.

7.5 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(s) will be moved from "Set Function Name" list box back to the "Select Function Name" list box.

7.6 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.

7.7 Overlay Function

Programs making use of the overlay function can be debugged. This section explains the settings for using the overlay function.

7.8 Displaying Section Group

When the overlay function 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]**.

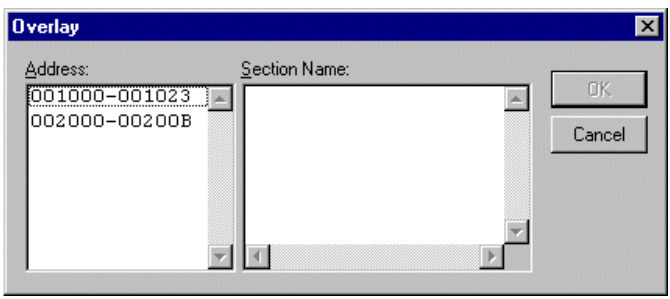


Figure 7.2: Overlay Dialog Box (at Opening)

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 by the overlay function. Click to select one of the address ranges in the "Address" list box.

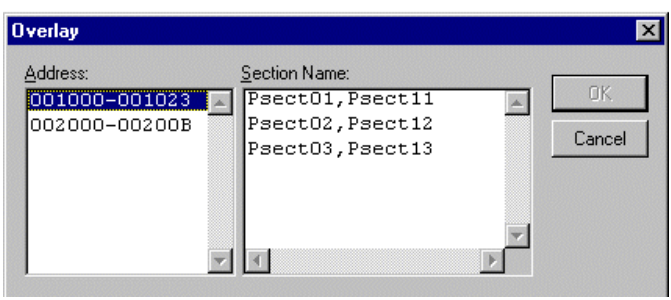


Figure 7.3: Overlay Dialog Box (Address Range Selected)

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 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.

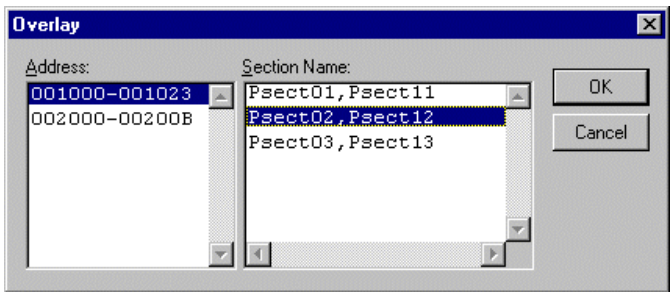


Figure 7.4: Overlay Dialog Box (Highest-Priority Section Group Selected)

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.

7.9 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 Source 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.

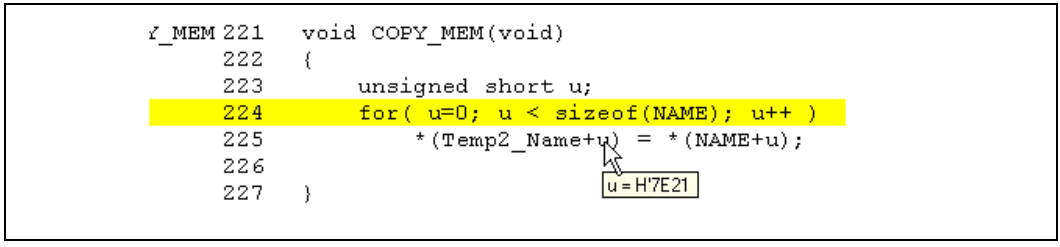


Figure 7.5: Locals View

7.10 Local Variables

To look at local variables, open the Locals window by choosing **[View->Locals]**.

The Locals window opens:

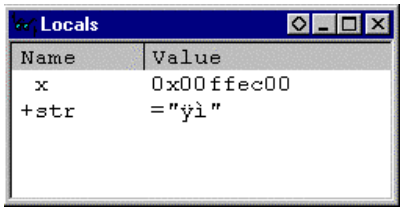


Figure 7.6: Locals Window

As you debug your program the Locals window will be updated. If a local variable is not initialized when defined, then the value in the Locals window will be undefined until a 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.

7.11 Watch Items

HEW allows you to open Watch windows, which contain a list of variables and their values. To open a Watch window choose **[View->Watch]**; or click on the Watch Window toolbar button [] if it is visible. A Watch window opens. Initially the contents of the window will be blank.

7.11.1 Adding a Watch

Use the Add Watch dialog box in the Watch window to add Watch items to the Watch window.

➲ To use Add Watch from a Watch View:

Open the Watch view.

Choose **[Add Watch]** from the popup menu.

The Add Watch dialog box opens:

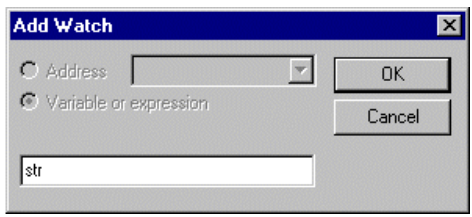


Figure 7.7: Add Watch Dialog Box

Enter the name of the variable that you wish to watch and click **"OK"**. The variable is added to the Watch window.

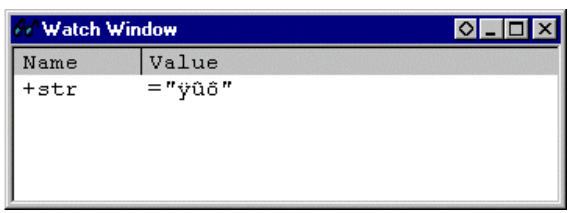


Figure 7.8: Watch Window

Note: If the variable that you have added is a local variable that is not currently in scope, HEW will add it to the Watch window but its value will be blank, or set to a question mark, '?'.

7.11.2 Expanding a Watch

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, double 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.

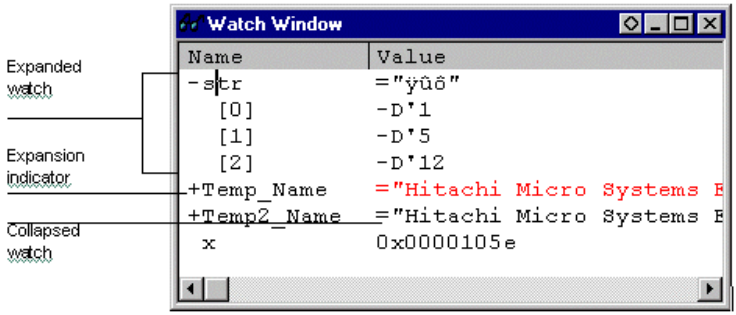


Figure 7.9: Expanding a Watch

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.

7.11.3 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:

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:

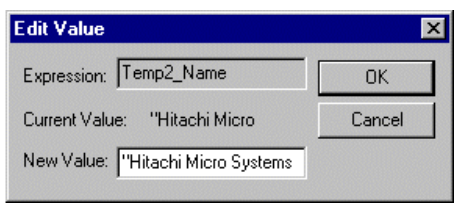


Figure 7.10: Edit Value Dialog Box

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

To delete a watch item, select it and choose **[Delete]** from the popup menu. The item is deleted and the Watch window updated.

8. 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.

HEW displays the results of measurement in three windows, according to the method and purpose of viewing the profile data.

8.1 Stack Information Files

The profile function allows the HEW to read the stack information files (extension: "**.SNI**") which are output by the Hitachi 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 popup 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, select "Other" from the "Category:" list box and check the "Stack information output" box in the "Link/Library" pane of the Standard Toolchain dialog box.

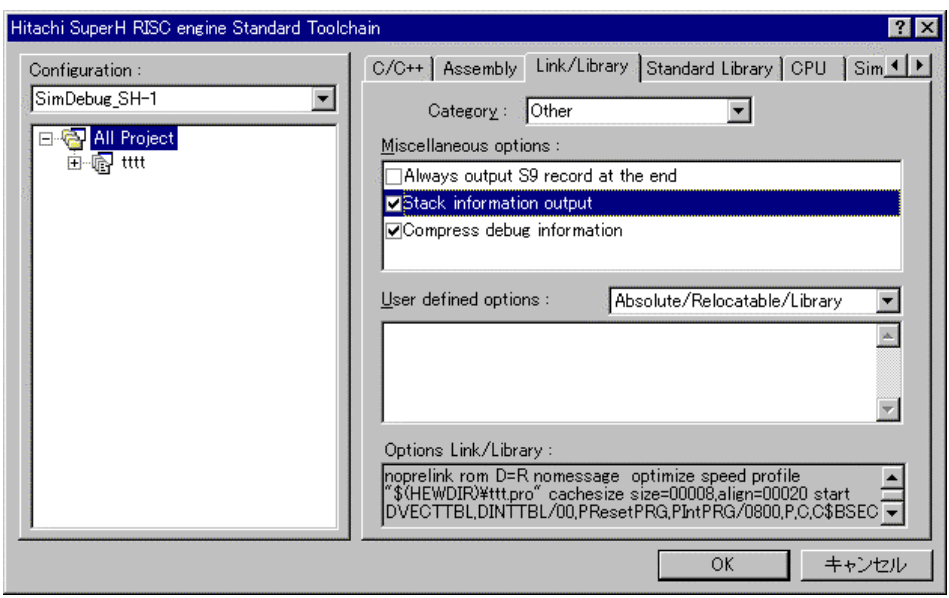


Figure 8.1: Standard Toolchain Dialog Box (1)

8.2 Profile Information Files

To create a profile information file, select 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 Hitachi 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, select “Optimize” from the “Category:” list box and check the “Include Profile:” box in the “Link/Library” pane of the Standard Toolchain dialog box, and specify the name of the profile information file.

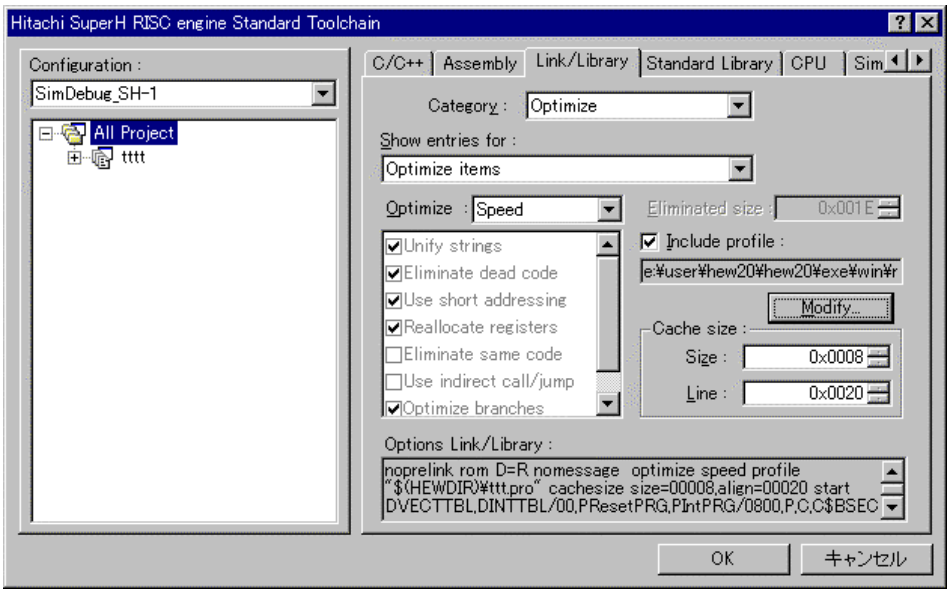


Figure 8.2: Standard Toolchain Dialog Box (2)

To enable the settings in the “Include Profile:” box, specify the “Optimize” list box as some setting other than “None”.

8.3 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” pane of the Options dialog box (figure 8.3) that can be opened by selecting **[Tools->Options]** from the main menu

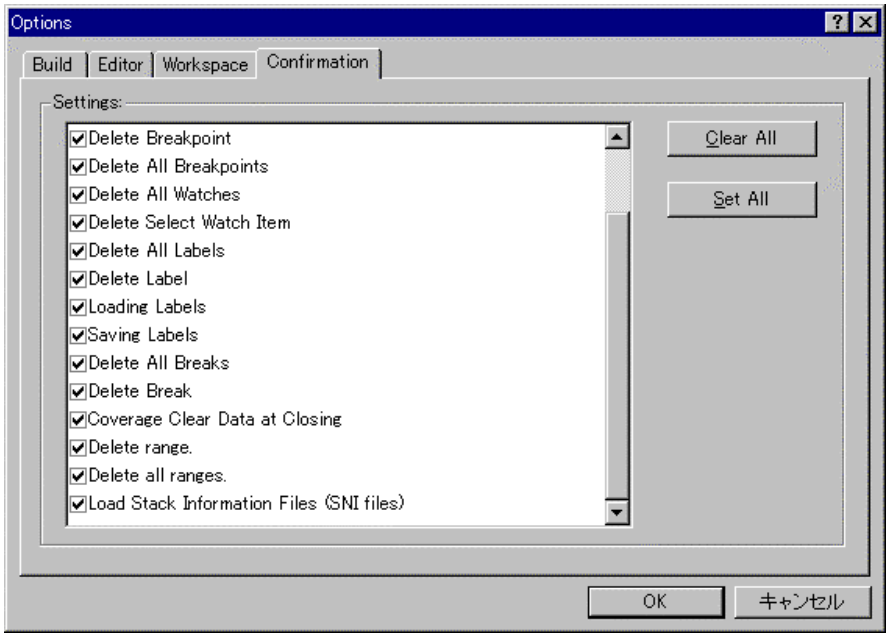


Figure 8.3: Options Dialog Box

8.4 Enabling the Profile

Select **[View->Profile]** to open the Profile window.

Select “Enable Profiler” from the pop-up menu of the Profile window. The item on the menu will be checked.

8.5 Specifying the measurement for profiling

You can specify whether to trace function calls or not. If you choose to trace them, their relationship among function calls is shown in the tree structure. If not, it not available, but it takes less time in measurement.

When you do not trace the function calls, select “Disable Tree (Not traces function call)” from popup.

If a function call is not used in a normal way (e.g. task switching of the OS), the Profile view may bi displayed incorrectly. In this case, do not trace function calls and just measure the profile.

8.6 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 tabs; a “List” tab and a “Tree” tab.

8.6.1 List Tab

This tab lists functions and global variables and displays the profile data for each function and variable.

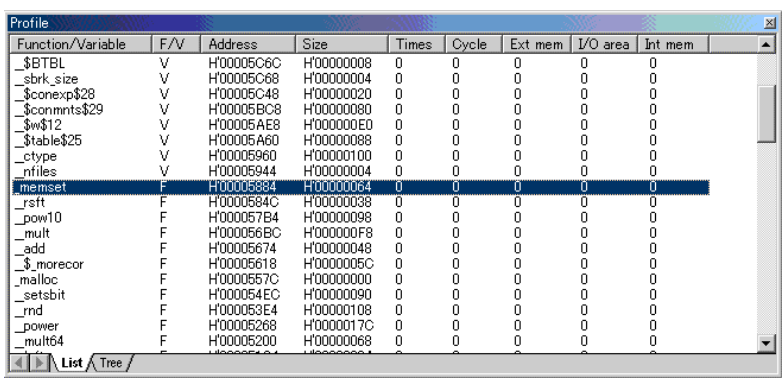


Figure 8.4: List Tab

8.6.2 Tree Tab

This tab displays the relation of function calls as a tree diagram along with the profile data that are values when the function is called.

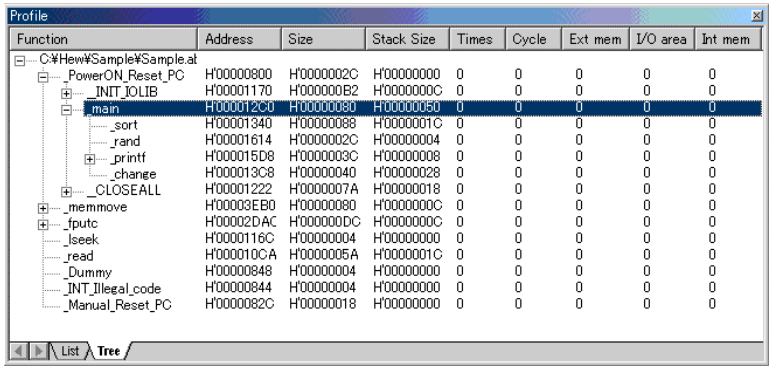


Figure 8.5: Tree Tab

8.6.3 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.

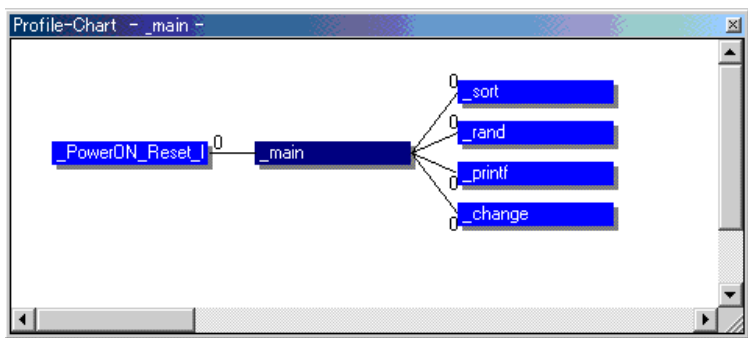


Figure 8.6: Profile-Chart Window

8.7 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.
- Others Measurement of a variety of target-specific data is also available. For details, refer to the simulator or emulator manual for the target platform that you are using.

8.8 Creating Profile Information Files

To create a profile information file, select 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.

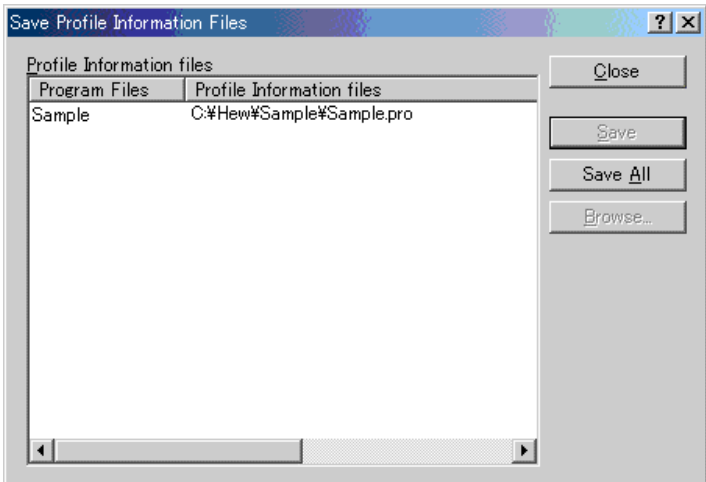


Figure 8.7: Save Profile Information Files Dialog Box

8.9 Notes

1. The number of executed cycles for an application program as measured by the profile function includes a margin of error. The profile function only allows the measurement of the proportions of execution time that the functions occupy in the overall execution of the application program. Use the Performance Analysis function to precisely measure the numbers of executed cycles.

2. The names of the corresponding functions may not be displayed when the profile information on a load module with no debug information is measured.

3. The stack information file (extension: “**.SNI**”) must be in the same directory as the load module file (extension: “**.ABS**”).

4. It is not possible to store the results of measurement.

5. It is not possible to store the results of measurement.

Appendix A - 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.

A.1 File format

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.

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>]]]].

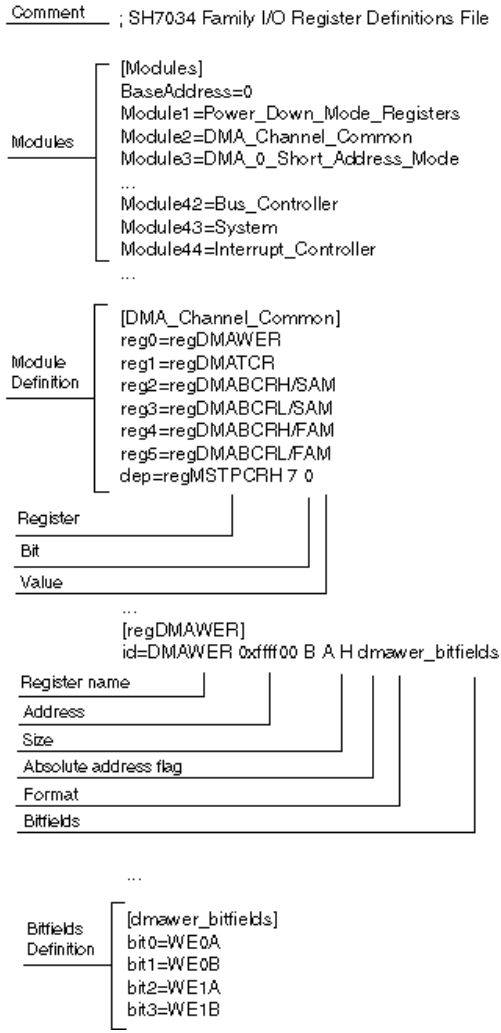
1. <name> register name to be displayed.
2. <address> address of the register.
3. <size> which may be B, W or L for byte, word, or long word (default is byte).
4. <absolute> which can be set to A if the register is at an absolute address. This is only relevant if the I/O area address range moves about 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.

Example :


Comment
; SH7034 Family I/O Register Definitions File
Modules
[Modules]
BaseAddress=0
Module1=Power_Down_Mode_Registers
Module2=DMA_Channel_Common
Module3=DMA_0_Short_Address_Mode
...
Module42=Bus_Controller
Module43=System
Module44=Interrupt_Controller
...
Module Definition
[DMA_Channel_Common]
reg0=regDMAWER
reg1=regDMATCR
reg2=regDMABCRH/SAM
reg3=regDMABCRL/SAM
reg4=regDMABCRH/FAM
reg5=regDMABCRL/FAM
dep=regMSTPCRH 7 0
Register
Bit
Value
...
[regDMAWER]
id=DMAWER 0xffff00 B A H dmawer_bitfields
Register name
Address
Size
Absolute address flag
Format
Bitfields
...
Bitfields Definition
[dmawer_bitfields]
bit0=WE0A
bit1=WE0B
bit2=WE1A
bit3=WE1B

Appendix B - Symbol File Format

In order for HEW to be able to understand and 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
```