H8S, H8/300 Series

High-performance Embedded Workshop 3

(for Windows® 98/Me, Windows NT® 4.0, Windows® 2000 and Windows® XP)

User's Manual

Renesas Development Environment System

User's Manual

Rev.1.00
2003.7.3

Renesas Technology
www.renesas.com

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.

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: 3

Introduction

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

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

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

About This Manual

This manual describes the HEW system. This manual is composed of two parts. HEW part describes information on the basic "look and feel" of the HEW and customizing the HEW environment and detail the build. Simulator/Debugger part describes Debugger functions of the High-performance Embedded Workshop.

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

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

Main application

1. Overview

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

1.1 Workspaces, Projects and Files

Just as a word processor allows you to create and modify documents, the High-performance Embedded Workshop allows you to create and modify workspaces. A workspace can be thought of as a container of projects and, similarly, a project can be 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.

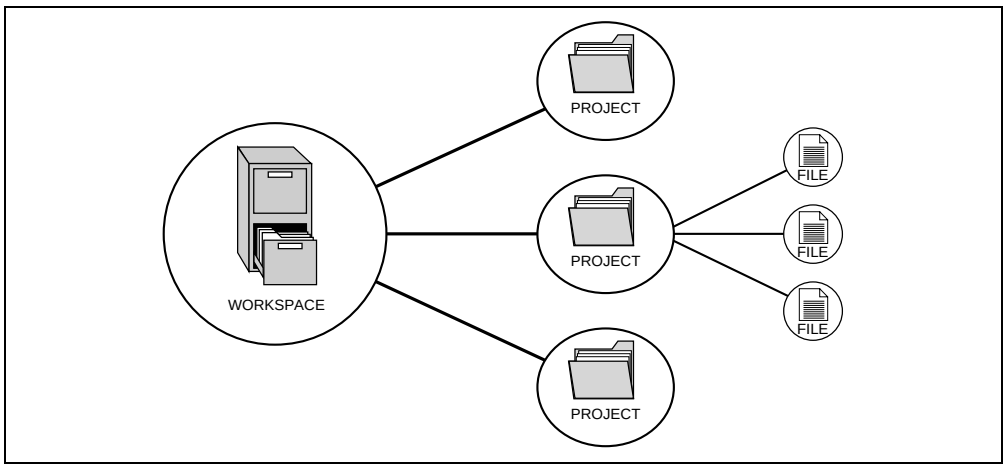


Figure 1.1: Workspaces, Projects and Files

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 The Main Window

The HEW main window appears as shown in figure 1.2.

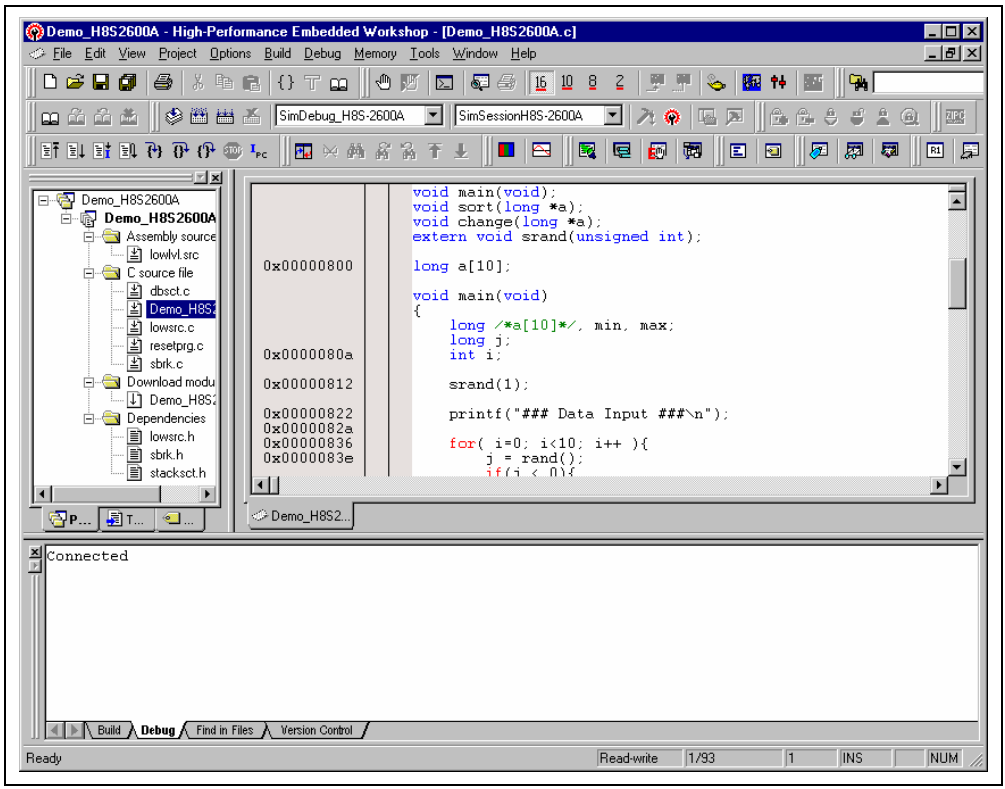


Figure 1.2: HEW Main Window

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

1.2.1 The Title Bar

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

1.2.2 The Menu Bar

The menu bar contains nine menus: File, Edit, View, Project, Options, Build, Tools, Window and Help. All of the menu options are grouped logically under these headings. For instance, if you wanted to open a file then the file menu is where you will find the right menu option, if you wanted to set-up a tool then the tools menu is the correct selection. The following sections will cover the functions of the various menu options, as they become

relevant. However, at this stage, it is worth taking a few moments to familiarize yourself with the options that each menu provides.

1.2.3 The Toolbars

The toolbars provide a shortcut to the options, which you will use the most often. There are eight default toolbars: Bookmarks, Debug, Debug Run, Editor, Search, Standard, Templates, Version Control, and Difference (as shown in figure 1.3 to 1.11). Toolbars can be created, modified and removed via the **[Tools->Customize…]** menu option (see chapter 6, "*Customizing the Environment*", for further information).

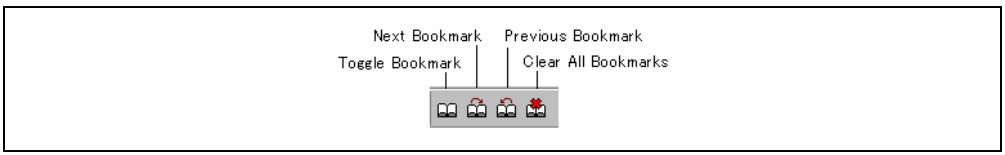


Figure 1.3: Bookmarks Toolbar

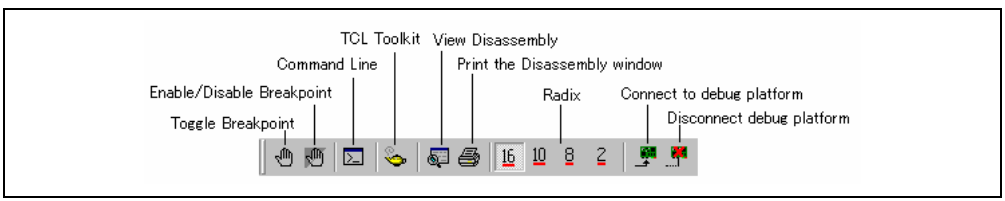


Figure 1.4: Debug Toolbar

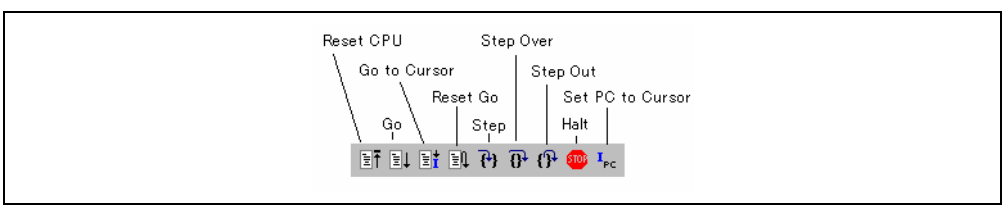


Figure 1.5: Debug Run Toolbar

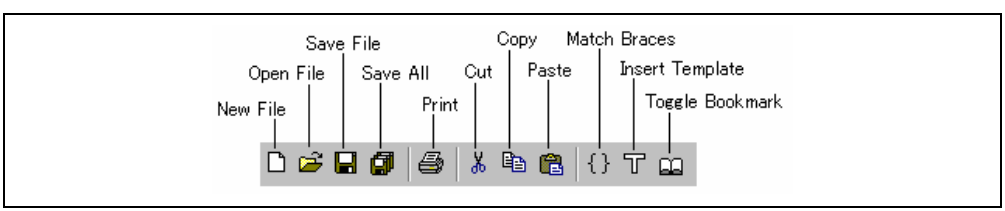


Figure 1.6: Editor Toolbar

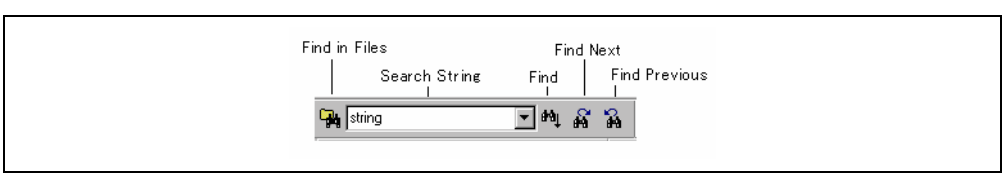


Figure 1.7: Search Toolbar

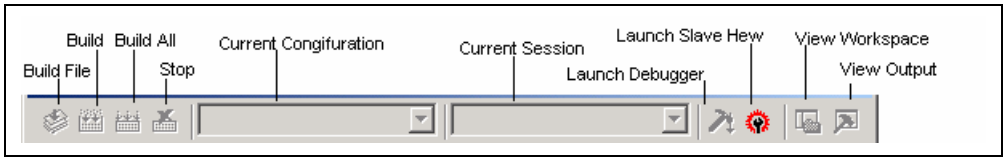


Figure 1.8: Standard Toolbar

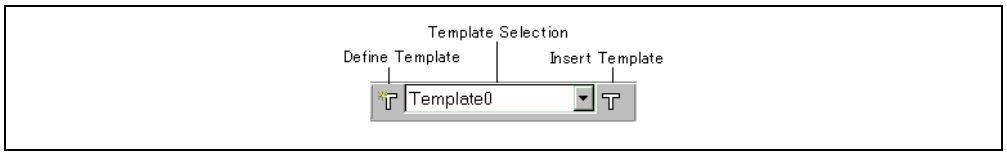


Figure 1.9: Templates Toolbar

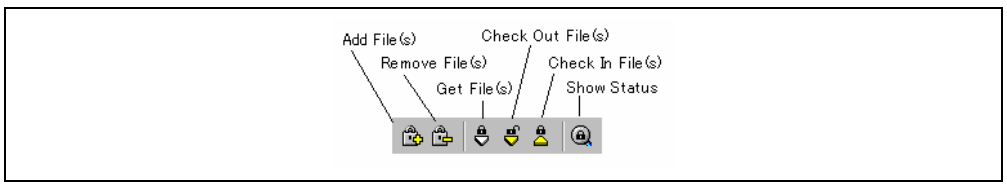


Figure 1.10: Version Control Toolbar

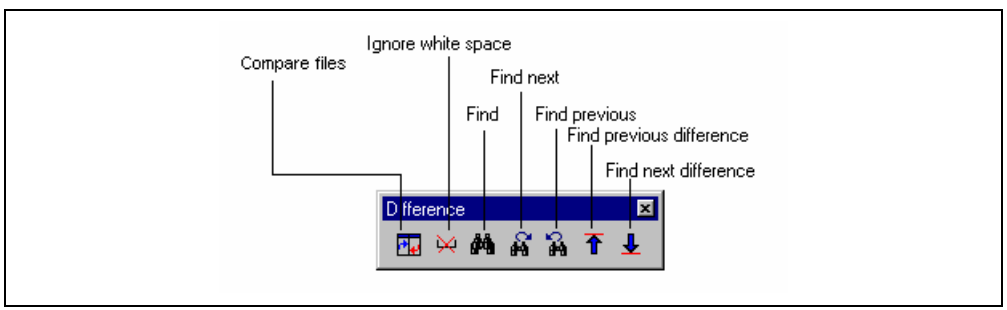


Figure 1.11: Difference Toolbar

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

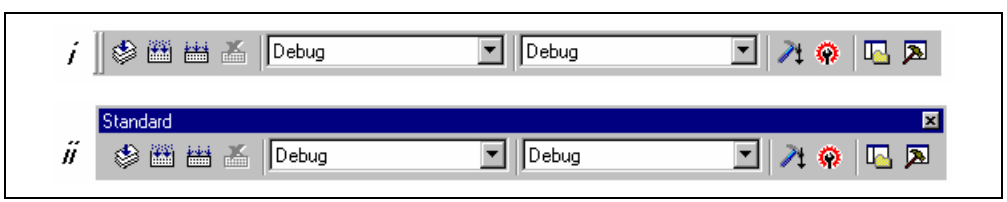


Figure 1.12: Standard Toolbar, Docked and Floating

➲ To dock the menu bar or a toolbar:

1. Double-click on the title bar of a floating menu bar or toolbar.

or:

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

➲ To float the menu bar or a toolbar:

1. Double-click on the control bar of a docked menu bar or toolbar.

or:

2. Drag the control bar of a docked menu bar or toolbar and draw it away from the edge of the HEW main frame and from an edge of the other docked windows, menu bar or toolbar.

1.2.4 The Workspace Window

The "Workspace" window has three panes. The "Projects" tab shows the current workspace, projects and files (figure 1.13). You can quickly open any project file or dependent file by double clicking on its corresponding icon.

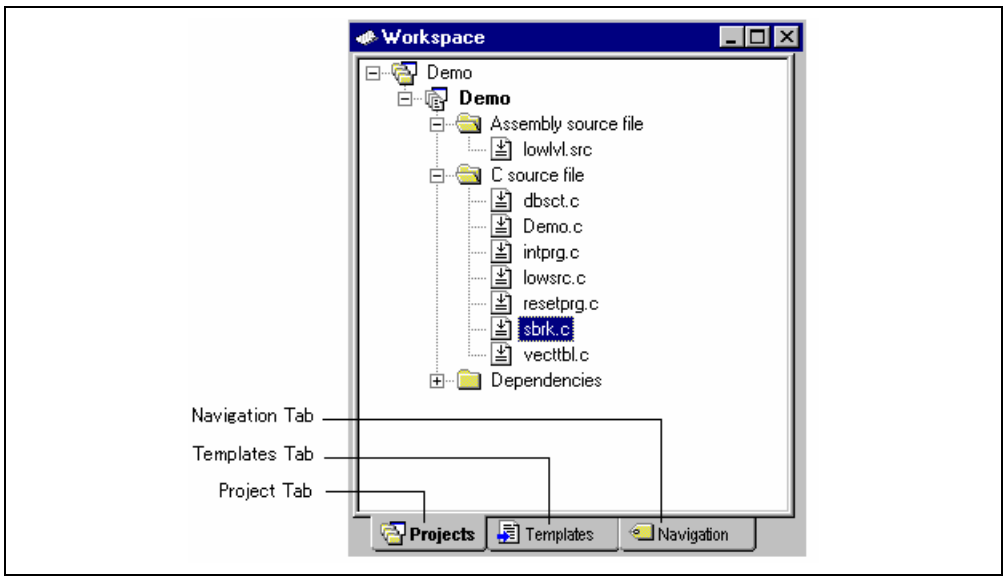


Figure 1.13: Workspace Window Projects Tab

The "Navigation" tab provides jumps to various textual constructs within your project's files. What is actually displayed within the navigation tab depends upon what components are currently installed. Figure 1.14 shows ANSI C functions. See chapter 2, "*Build Basics*", for more information on the "Workspace" window.
The "Templates" tab displays template settings. See 4.12, "Templates", for more information about a template.

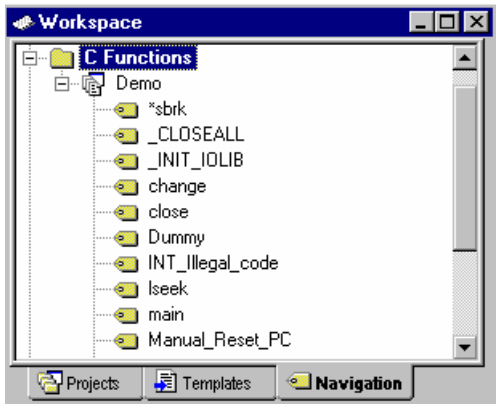


Figure 1.14: Workspace Window Navigation Tab

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

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

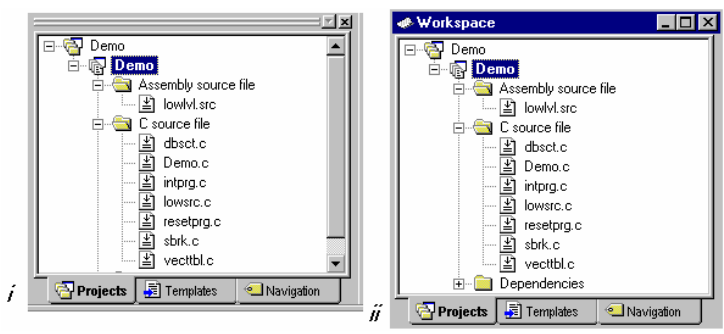


Figure 1.15: Workspace Window, Docked and Floating

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

Figure 1.16: Control Bar of Docking Window

- To dock the "Workspace" window or the "Output" window:

[Allow Docking] must be checked on the pop-up menu of the window to dock the "Workspace" window or the "Output" window. (The pop-up menu will be displayed when you click the right mouse button anywhere inside the window.) Then you have two ways to dock the window.

1. Double-click on the control bar of a floating window.

or:

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

➲ To float the "Workspace" window or the "Output" window:

[Allow Docking] must be checked on the pop-up menu of the window to float the "Workspace" window or the "Output" window. (The pop-up menu will be displayed when you click the right mouse button anywhere inside the window.) Then you have two ways to float the window.

1. Double-click on the control bar of a docking window.

or:

2. Drag the control bar of a docked window and draw it away from the edge of the HEW main frame and from an edge of the other docked windows, menu bar or toolbar.

➲ To hide the "Workspace" window or the "Output" window:

Click on the close button, which is located in the top right corner of the window. Or push the right mouse button anywhere inside a floating window and select **[Hide]** on the pop-up menu.

➲ To display the "Workspace" window or the "Output" window:

Select **[View->Workspace]** or **[View->Output]**, respectively.

1.2.5 The Editor Window

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

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

If you are unsure what purpose a column has or what the information it is displaying is if you place the cursor over the column a tool tip is displayed showing its identity.

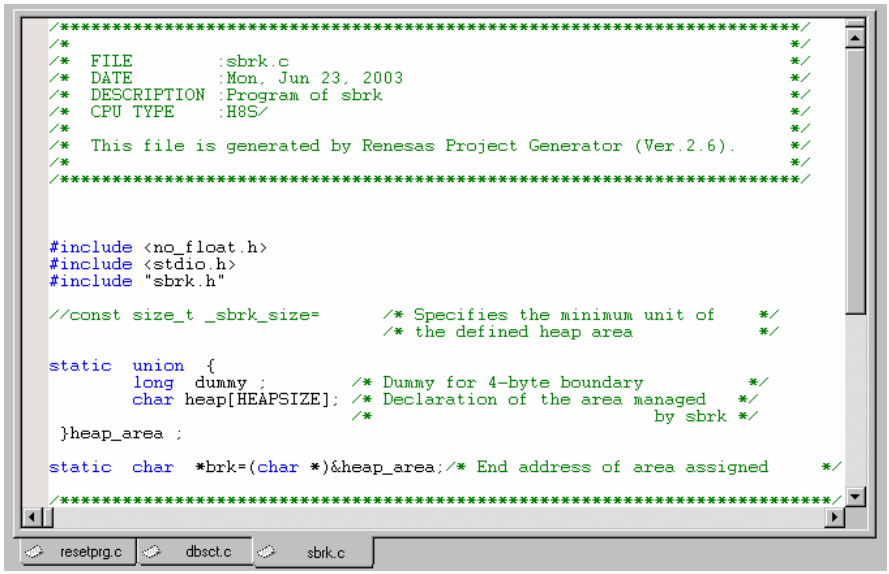


Figure 1.17: Editor Window

The editor window can be customized via the "Format Views" dialog box, which can be invoked via the **[Tools->Format Views…]** menu option. This dialog allows you to configure fonts, colors, tabs and so on for the editor window. It also allows the user to change the look of other views, which have been installed by HEW. If you would prefer to use your favorite editor rather than the HEW internal editor then specify your alternative in the "Options" dialog box, which can be invoked via the **[Tools->Option…]** menu option. For further details on how to use and configure the editor, refer to chapter 4, "*Using the Editor*".

1.2.6 The Output Window

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

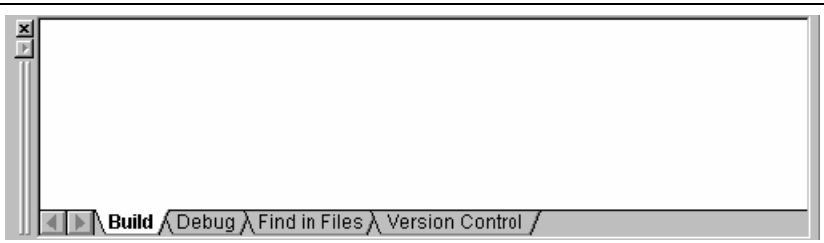


Figure 1.18: Output Window

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

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

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

1.2.7 The Status Bar

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

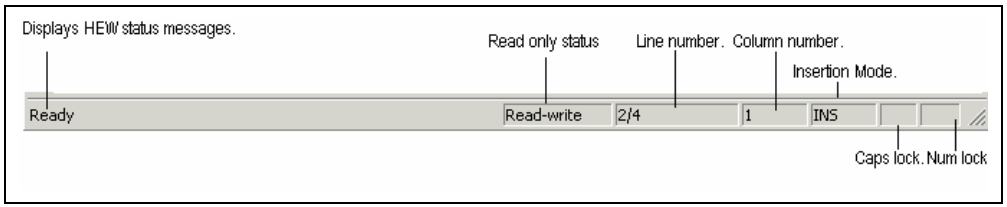


Figure 1.19: Status Bar

1.3 The Help System

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

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

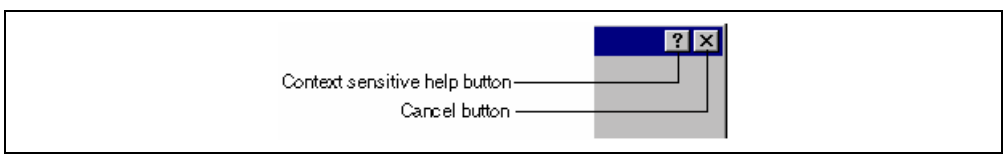


Figure 1.20: Help Button

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

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

1.4 Launching the HEW

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

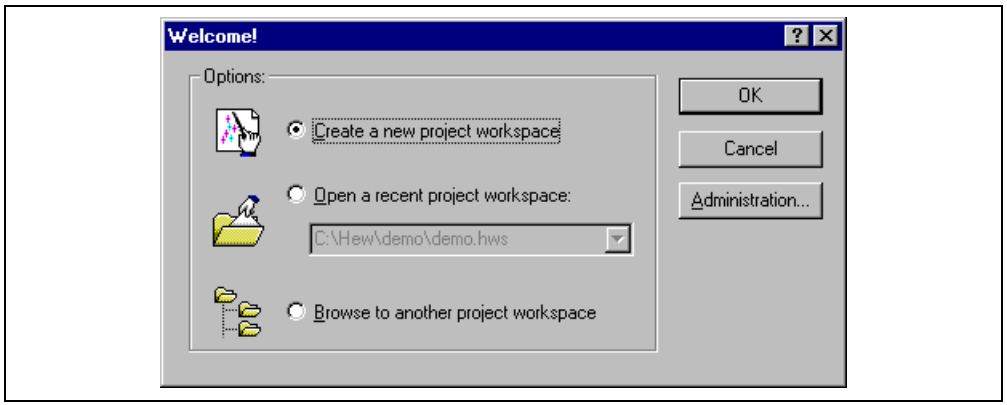


Figure 1.21: Welcome! Dialog

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

1.5 Exiting the HEW

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

1.6 Component System Overview

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

2. Build Basics

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

2.1 The Build Process

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

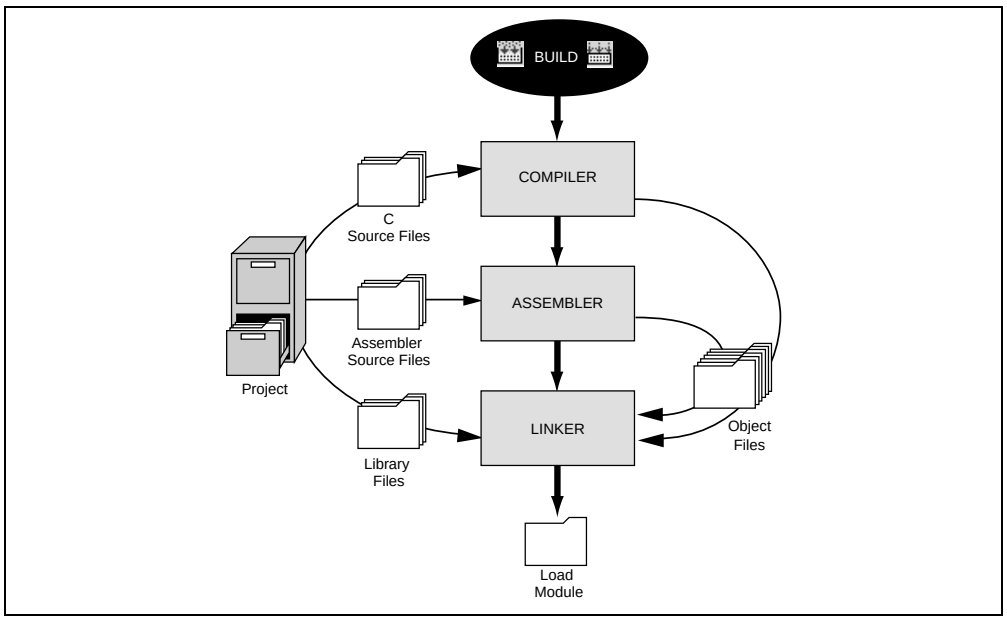


Figure 2.1: Typical Build Process

In the example shown in figure 2.1 the compiler is the first phase, the assembler is the second phase and the linker is the third and final phase. During the compiler phase, the C source files from the project are compiled in turn, during the assembler phase, the assembler source files are assembled in turn. During the linker phase all library files and output files from the compiler and assembler phases are linked together to produce the load module. This module can then be downloaded and used by the debugger functionality in HEW.

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

2.2 Project Files

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

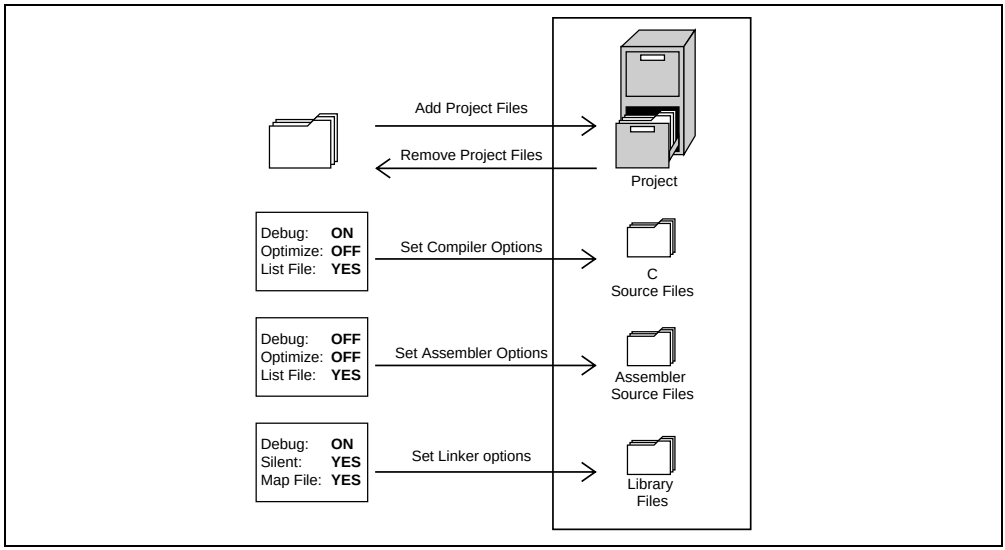


Figure 2.2: Editing a Project

2.2.1 Adding Files to a Project

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

➲ To add a files to a project:

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

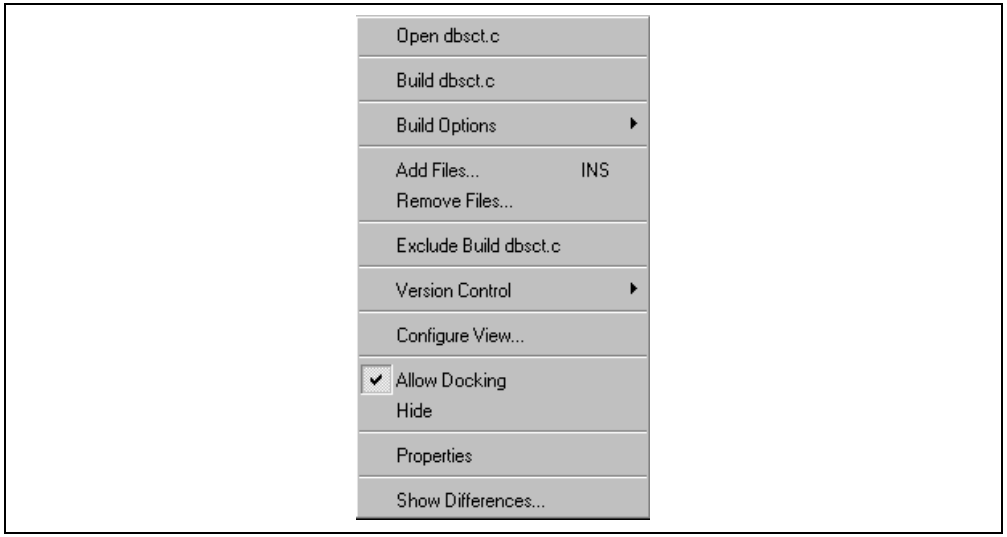


Figure 2.3: Project Pop-up Menu

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

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

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

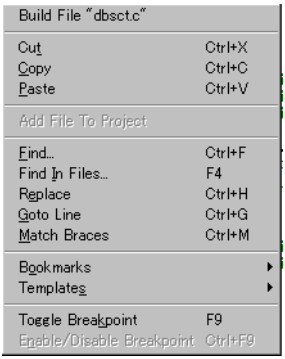


Figure 2.4: Editor Window Pop-up Menu

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

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

2.2.2 Removing Files from a Project

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

➲ To remove files from a project:

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

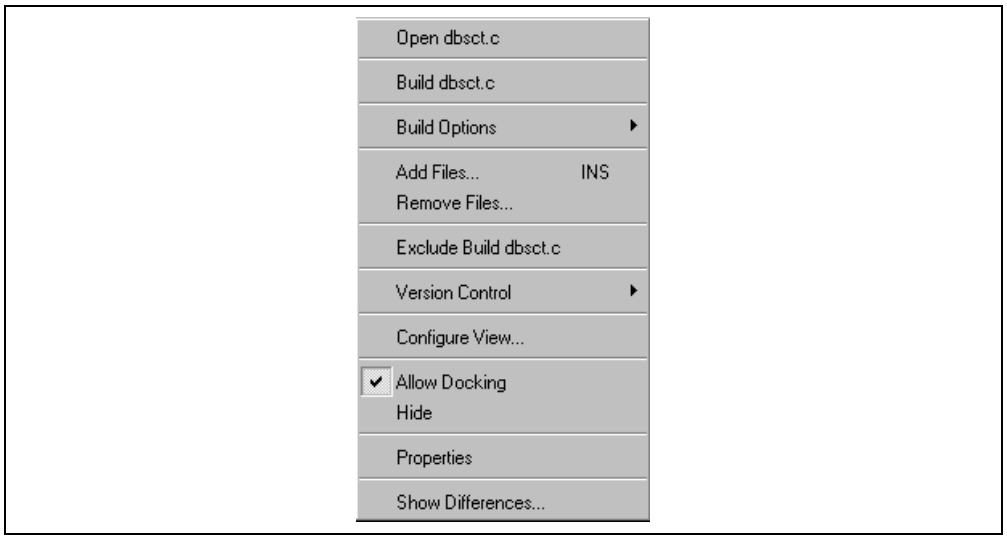


Figure 2.5: Projects Tab Pop-up Menu

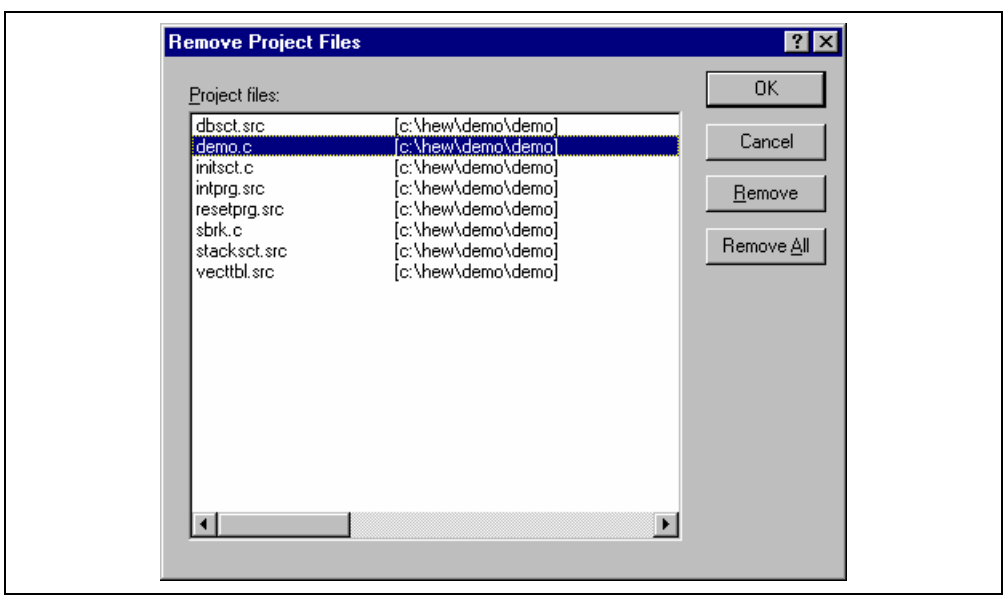


Figure 2.6: Remove Project Files Dialog

2. Select the file or files which you want to remove from the "Project files" list.
3. Click the "Remove" button to remove the selected files or click "Remove All" to remove all project files.
4. Click "OK" to remove the files from the project.

➲ To remove selected files from a project:

1. Select the files, which you want to remove, in the "Projects" tab of the "Workspace" window. Multiple files can be selected by holding down the **SHIFT** or **CTRL** key.
2. Press the **DEL** key. The files will be removed.

2.2.3 Excluding a Project File from Build

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

➲ To exclude a file in a project from build:

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

2.2.4 Including a Project File in Build

An excluded file can be included in the project again.

➲ To include a file which has been excluded from build:

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

2.3 File Extensions and File Groups

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

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

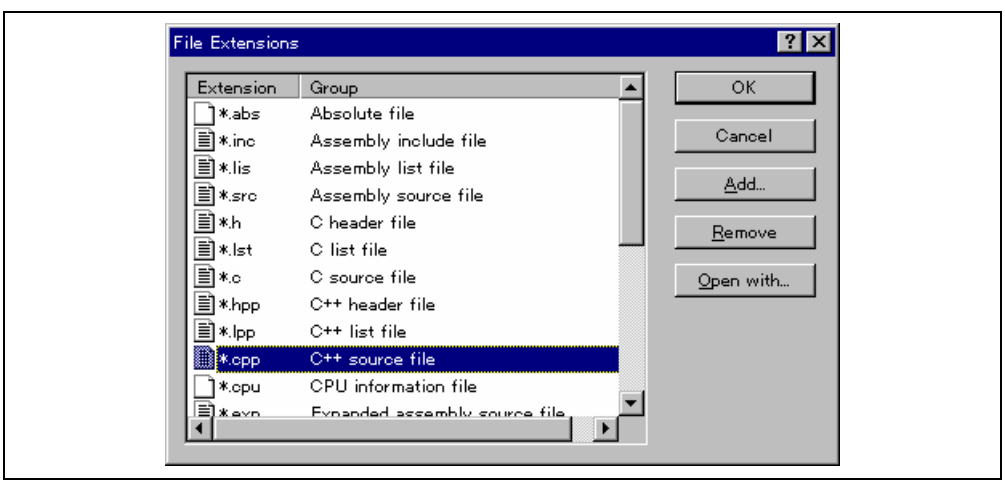


Figure 2.7: File Extensions Dialog

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

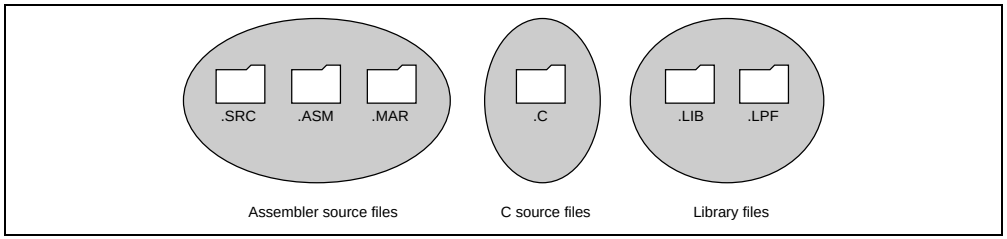


Figure 2.8: File Extensions and Groups

When creating a new extension you should consider whether the extension belongs to a group, which is already defined, or whether you need to create a new file group. If you are adding a completely new type of file then you will want to create a new file group. This process is described below.

➲ To create a new file extension in a new file group:

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

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

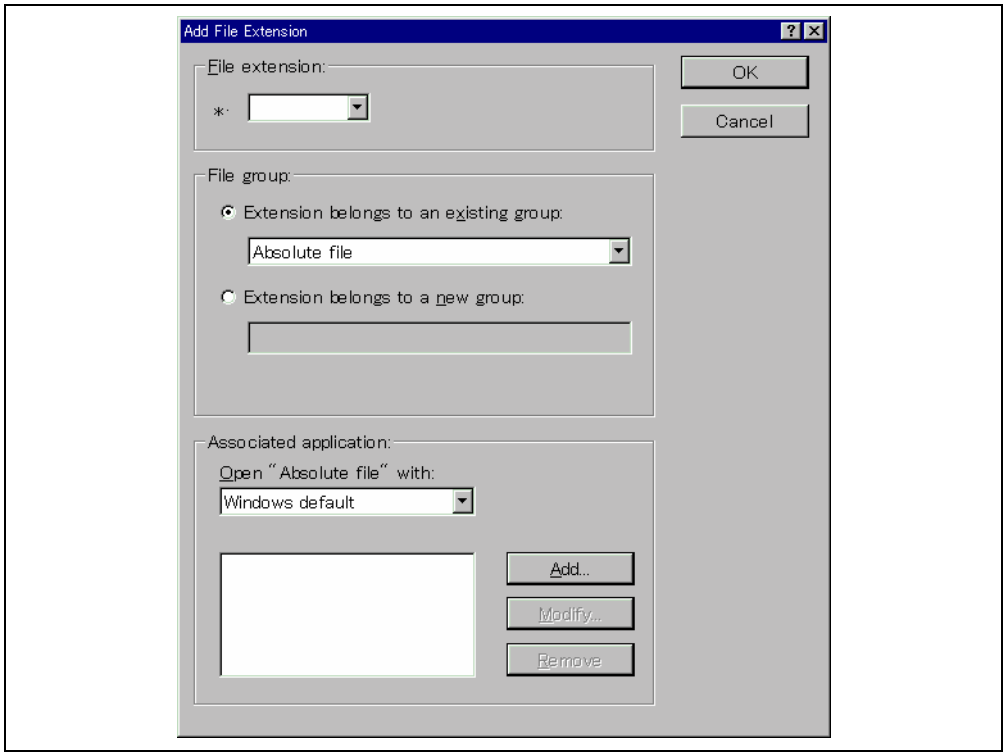


Figure 2.9: Add File Extension Dialog (New Group)

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

➲ To create a new file extension in an existing file group:

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

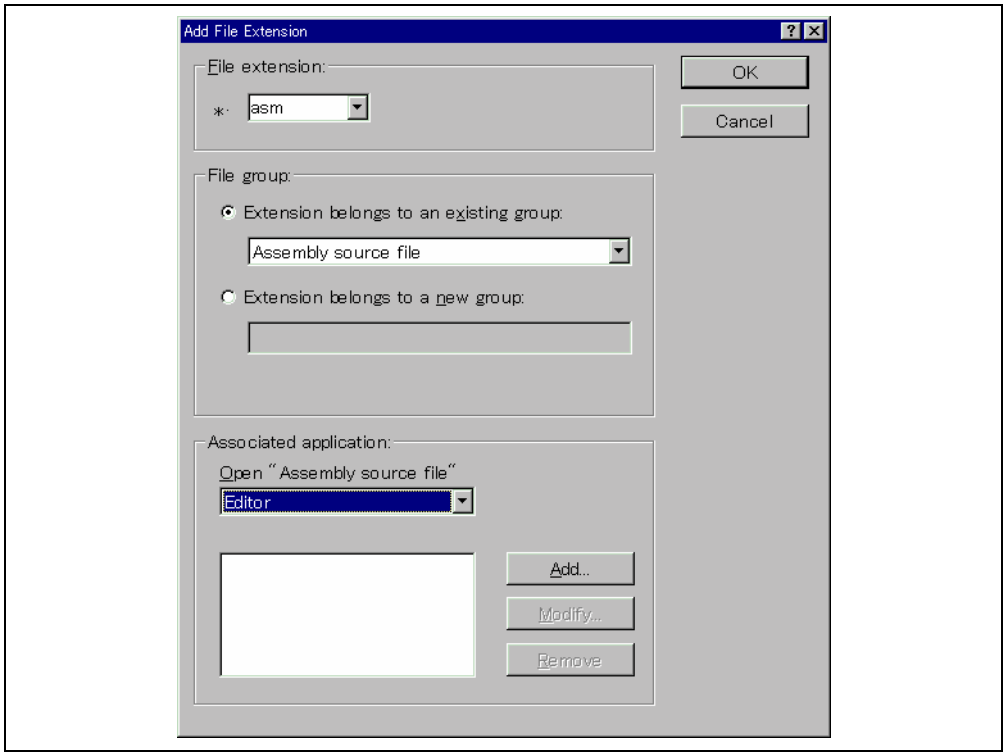


Figure 2.10: Add File Extension Dialog (Existing Group)

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

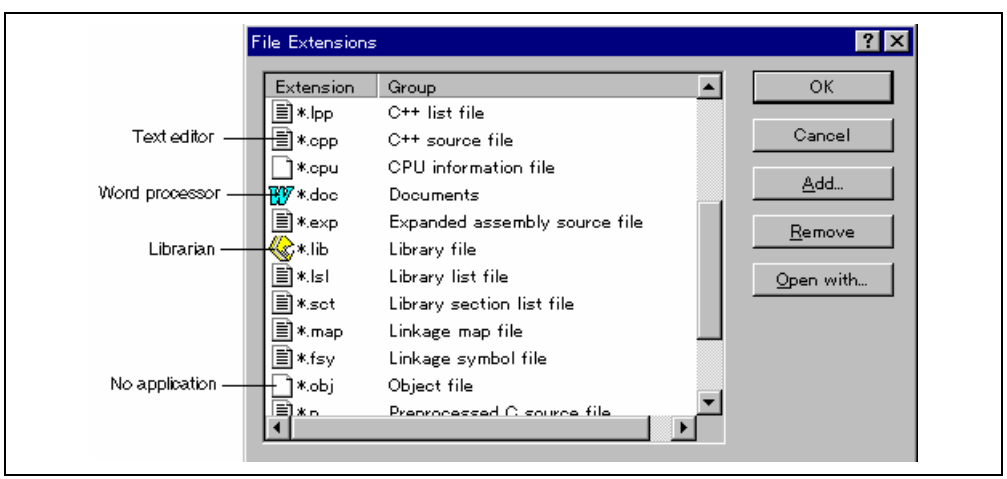


Figure 2.11: File Groups and Applications

➲ To associate an application with a file group:

1. Select the file group to be associated from the "File Extensions" dialog (figure 2.11).
2. Click the "Open with…" button. The "Modify File Extension" dialog will be displayed (figure 2.12).

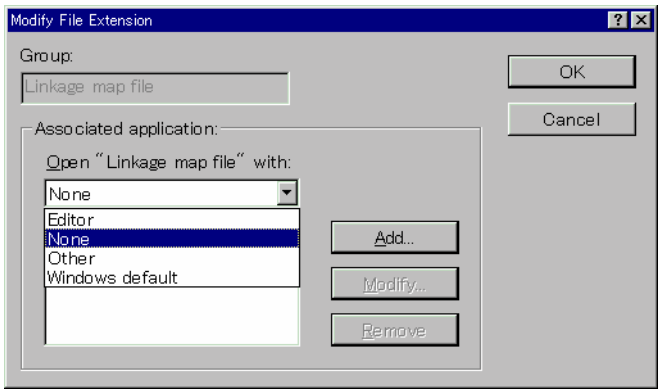


Figure 2.12: Modify File Extension Dialog

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

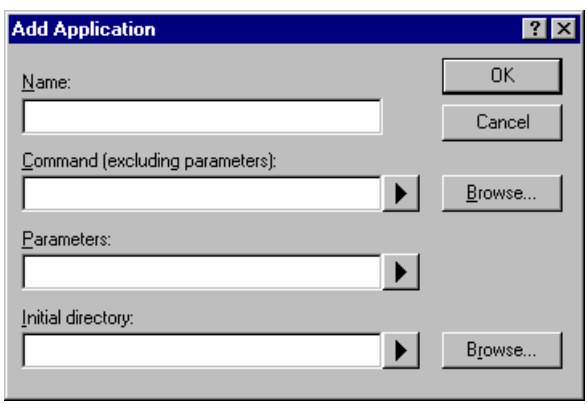


Figure 2.13: Add Application Dialog

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

7. Click "OK" to set the application for the selected file group.

2.4 Specifying How to Build a File

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

➲ To set options for a build phase:

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

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

2.5 Build Configurations

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

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

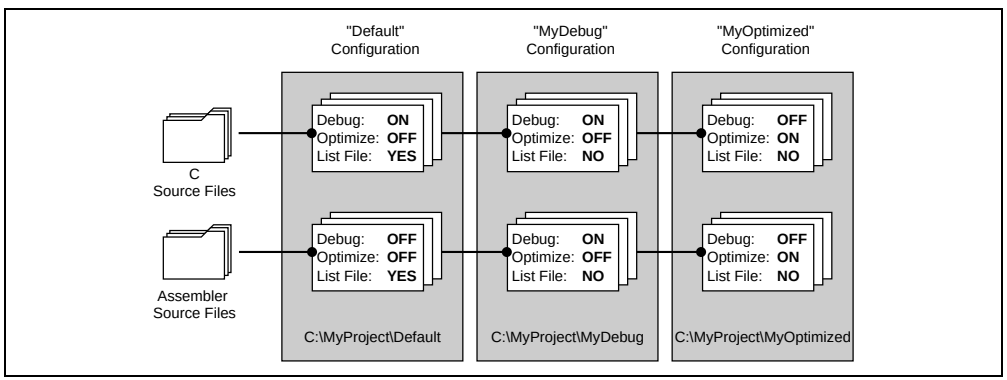


Figure 2.14: Configurations and File Options

2.5.1 Selecting a Configuration

The current configuration can be set in two ways:

Either:

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

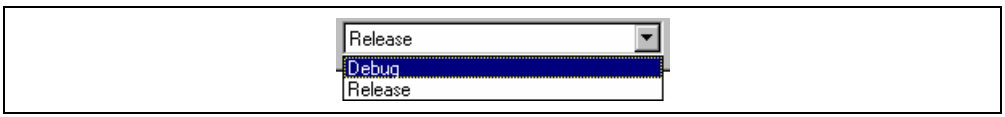


Figure 2.15: Toolbar Selection

or:

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

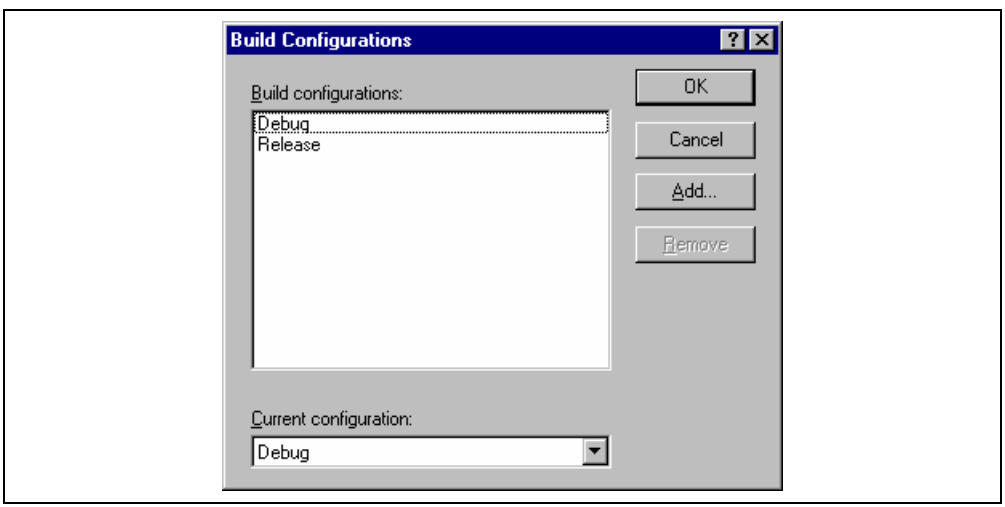


Figure 2.16: Build Configurations Dialog

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

2.5.2 Adding and Deleting Configurations

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

➲ To add a new configuration:

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

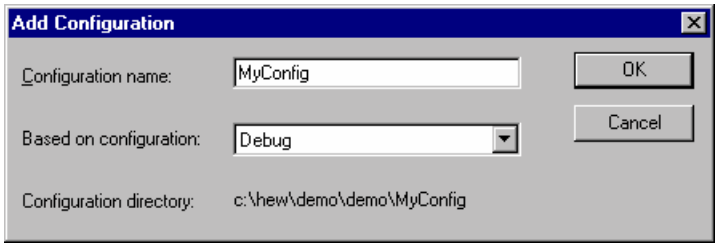


Figure 2.17: Add Configuration Dialog

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

➲ To remove a configuration:

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

2.6 Building a Project

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

2.6.1 Building a Project

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

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

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

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

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

2.6.2 Building Individual Files

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

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

2.6.3 Stopping a Build

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

➲ To stop a build:

1. Select **[Build->Stop Build]** or click the stop build toolbar button (). The build will be stop after the current file has been built.
2. Wait until the message "Build Finished" appears in the "Output" window before continuing.

➲ To forcibly terminate a current tool

1. Select **[Build->Terminate Current Tool]**. The HEW will attempt to stop the tool immediately.

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

2.6.4 Building Multiple Projects

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

➲ To build multiple projects:

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

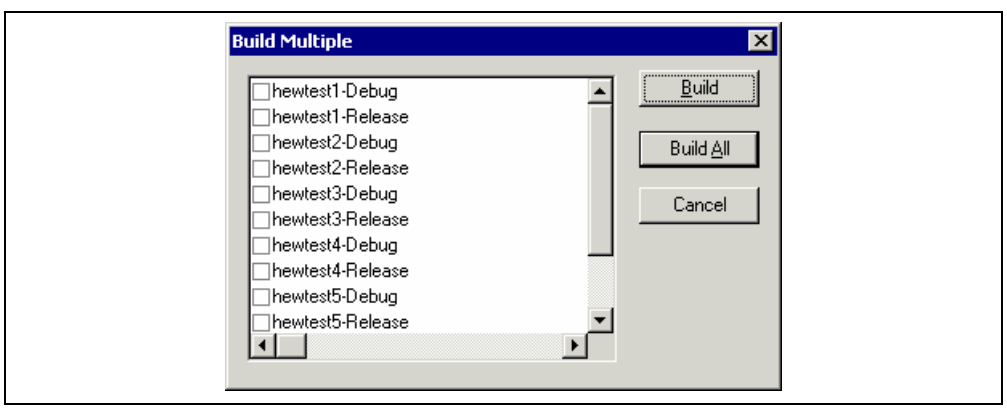


Figure 2.18: Build Multiple Dialog

2.6.5 The Output Window

When a tool executes (i.e. compiler, assembler, linker etc.) its output is displayed in the "Output" window. If any of the tools produce any errors or warnings then they are displayed along with the source file name and the line

number at which the error is located. To quickly locate a specific bug, double click on a given error/warning to invoke the current editor.

2.6.6 Controlling the Content of the Output Window

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

➲ To view or hide extra information during a build:

1. Select **[Tools->Options...]**. The "Options" dialog will be displayed.
2. Select the "Build" tab (figure 2.19).
3. Set the three check boxes in the "Show" group as follows. "Command line" controls whether the command line is shown as each tool is executed. "Environment" controls whether the environment is shown as each tool is executed. "Initial directory" controls whether the current directory is shown as each tool is executed.

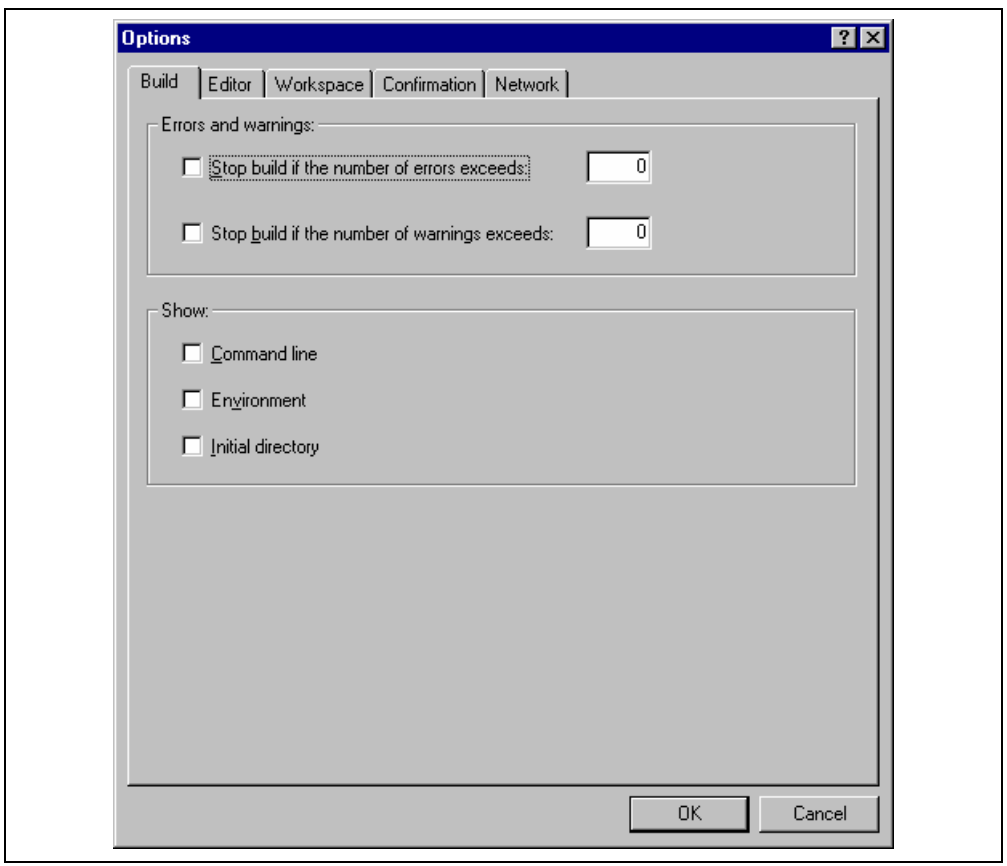


Figure 2.19: Options Dialog Build Tab

2.7 File Dependencies

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

➲ To update a project's dependencies:
Select **[Build->Update All Dependencies]** or click the right mouse button on a project icon in the "Projects" tab of the "Workspace" window and select **[Update All Dependencies]** from the pop-up menu.

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

2.8 Configuring the Workspace Window

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

2.8.1 Show Dependencies under Each File

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

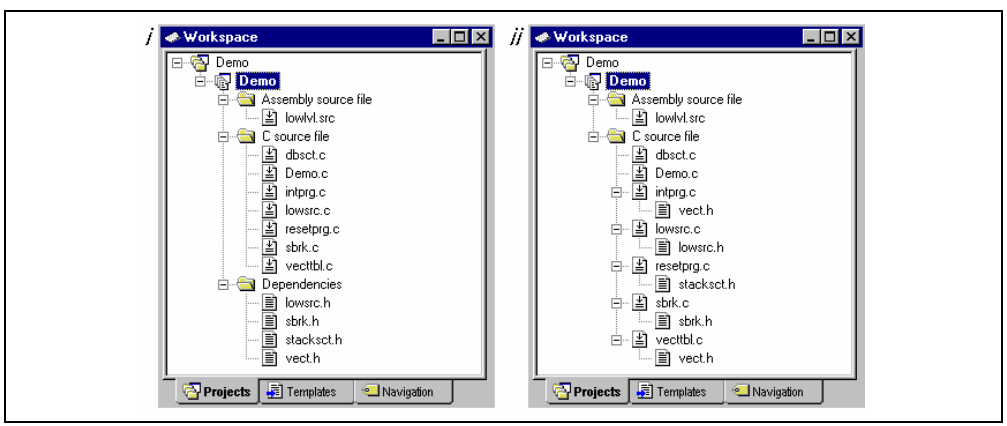


Figure 2.20: Dependencies under Each File

2.8.2 Show Standard Library Includes

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

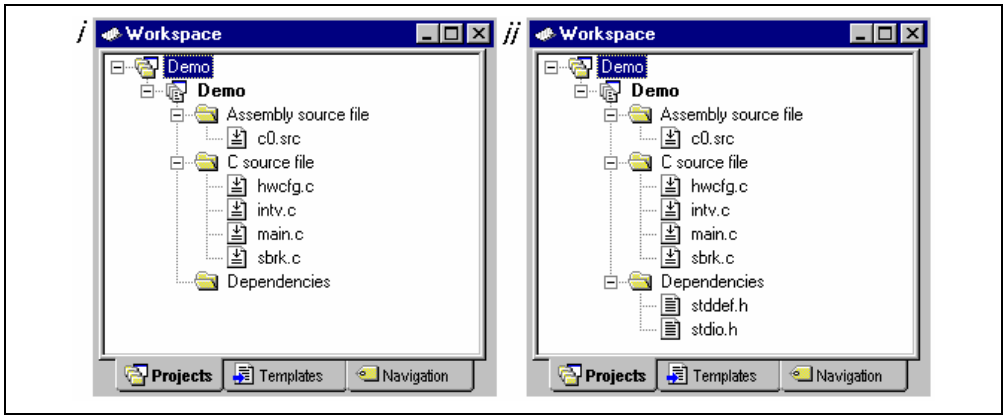


Figure 2.21: Standard Library Includes

2.8.3 Show File Paths

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

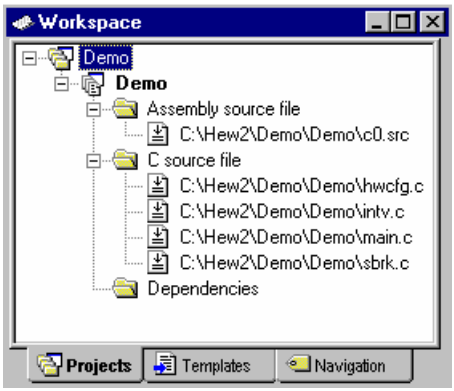


Figure 2.22: File Paths Shown

2.9 Setting the Current Project

➲A workspace can contain more than one project but only one of the projects can be active at any time. This active project is the one, which build actions and debug operations can be performed on. There are three states a project can be in. The current project, a loaded project or an unloaded project. If the project is loaded it is possible to open the project files directory and view the files. It is also possible to change the builder or debugger options for the project. A loaded project can have tool executions performed on it from the **[Tools]** menu. If the project is unloaded, its icon appears "grayed" in the "Projects" tab of the "Workspace" window and no actions can be performed upon it. You can set which project is active in a workspace.

➲ To set a project as the current project:

1. Select the project from the "Projects" tab of the "Workspace" window.
2. Click the right mouse button to display the pop-up menu and select the **[Set as Current Project]** option.

or:

1. Select the project, which you want to make active from the **[Project->Set Current Project]** sub-menu.

➲ To load a project in the workspace:

1. Select the unloaded project from the "Projects" tab of the "Workspace" window.
2. Click the right mouse button to display the pop-up menu and select the **[Load Project]** option.
3. This will also set the project that was just loaded as the current project.

➲ To unload a project in the workspace:

1. Select the active project from the "Projects" tab of the "Workspace" window.
2. Click the right mouse button to display the pop-up menu and select the **[Unload Project]** option.

2.10 Inserting a Project into a Workspace

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

➲ To insert a new project into a workspace:

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

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

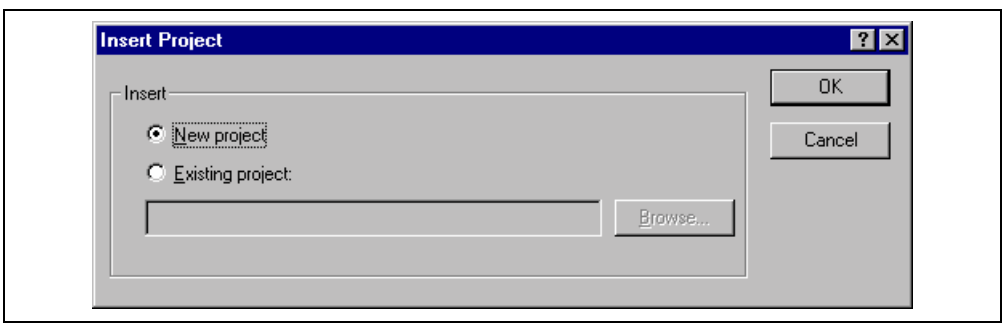


Figure 2.23: Insert Project Dialog

➲ To insert an existing project into a workspace:

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

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

2.11 Specifying Dependencies between Projects

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

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

➲ To make projects depend upon another:

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

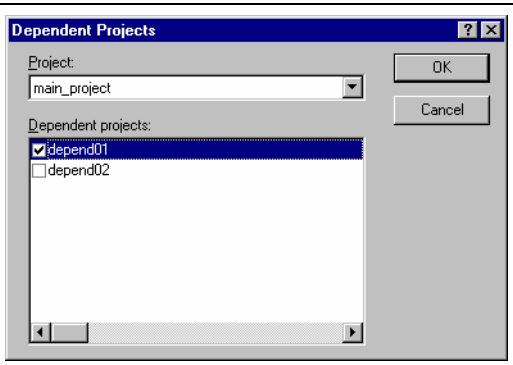


Figure 2.24: **Dependent Projects dialog**

2.12 Removing a Project from a Workspace

➲ To remove a project from a workspace:

1. Select the project from the "Projects" tab of the "Workspace" window and click the right mouse button to invoke a pop-up menu.
2. Select the **[Remove Project]** option.

or:

1. Select the project from the "Projects" tab of the "Workspace" window.
2. Press the **DEL** key.
3. Dialog box is displayed. You confirm to remove the project. In the "Options" dialog box of the "Tools" menu, you can select this confirmation is performed or not.

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

2.13 Loading/unloading a Project into/from a Workspace

➲ To load a project into a workspace:

1. Select the project from the "Projects" tab of the "Workspace" window and click the right mouse button to invoke a pop-up menu.
2. Select the **[Load Project]** option.

➲ To unload a project from a workspace:

1. Select the project from the "Projects" tab of the "Workspace" window and click the right mouse button to invoke a pop-up menu.
2. Select the **[Unload Project]** option.

Note: You can load or unload plural projects at a time. That is more efficient to load or unload a project individually.

2.14 Relative projects paths in the workspace

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

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

➲ To change a projects relative path flag:

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

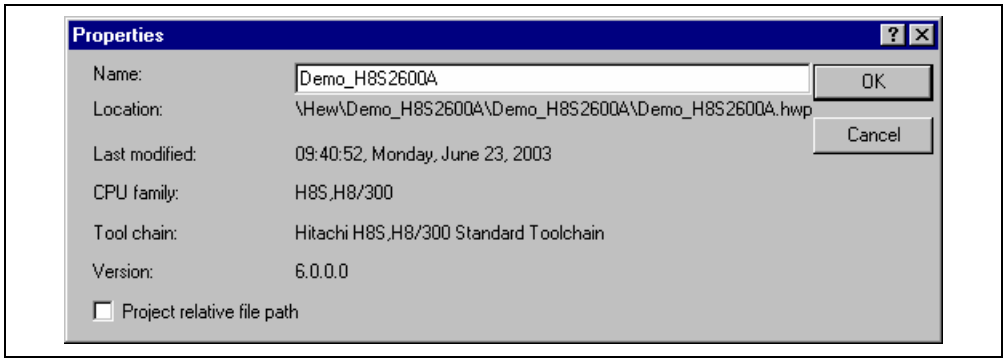


Figure 2.25: Properties Dialog

2.15 User folders in the workspace

In the High-performance Embedded Workshop it is possible to add folders to your workspace window. This allows you to logically group your files into certain areas within a project. The folder can be set to any name and this is entered in a dialog.

- To add a user folder:
  1. Select the project in the workspace window.
  2. Right click and then add folder.
  3. Enter the name and click OK.
  4. You can now drag and drop files into this folder to group them logically.
- To remove a user folder:
  1. Select the project in the workspace window.
  2. Right click and then select Remove folder. Note that the folder must be empty and that the delete key can also be used instead of the pop-up.
- To modify a user folder name:
  1. Select the project in the workspace window.
  2. Right click and then select Modify folder name.
  3. Enter the new name in the dialog.
  4. Click "OK".

3. Advanced Build Features

This chapter explains the more advanced build concepts.

3.1 The Build Process Revisited

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

3.1.1 What is a Build?

Building a project means applying a set of tools upon certain input files in order to produce the desired output. Thus, we apply a compiler upon C/C++ source files in order to create object files, we apply an assembler upon assembler source files in order to create object files and so forth. At each step or "phase" of the build, we apply a different tool upon a different set of input files. Figure 3.1 presents another view of the build process.

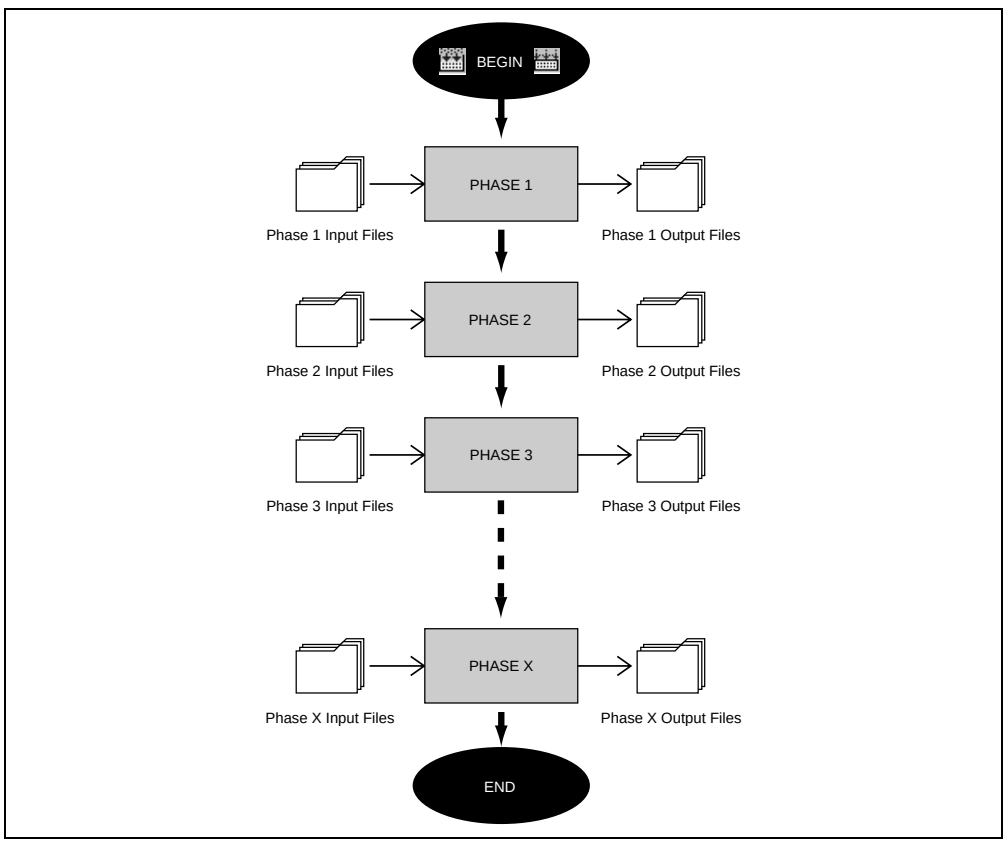


Figure 3.1: Build Process

The High-performance Embedded Workshop provides the ability to change this build process via its "Build Phases" dialog, which can be, accessed via the **[Options->Build Phases…]** (figure 3.2). On the left-hand side

are the phases that are defined in the current project (Figure 3.2 shows a standard set of build phases). The remainder of this chapter details the various functions that the "Build Phases" dialog provides.

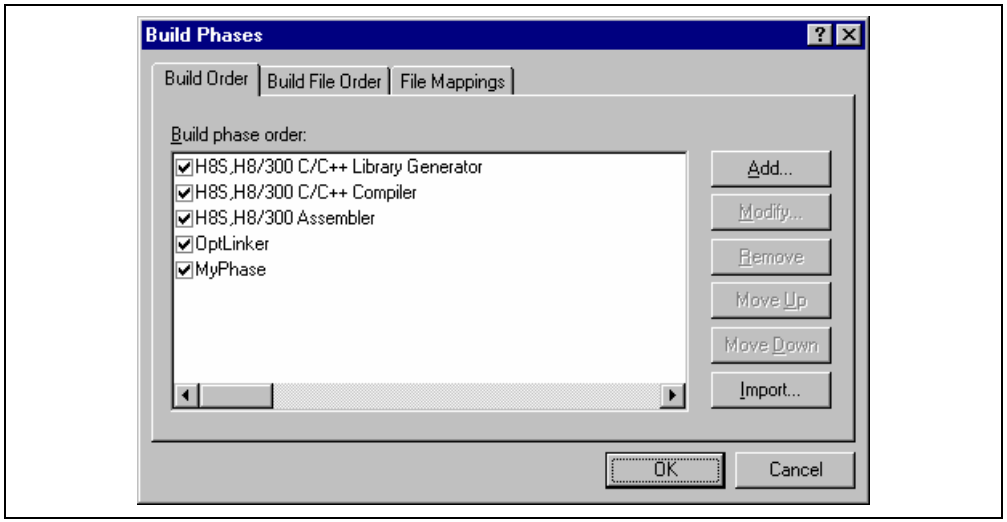


Figure 3.2: Build Phases Dialog

3.2 Creating a Custom Build Phase

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

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

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

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

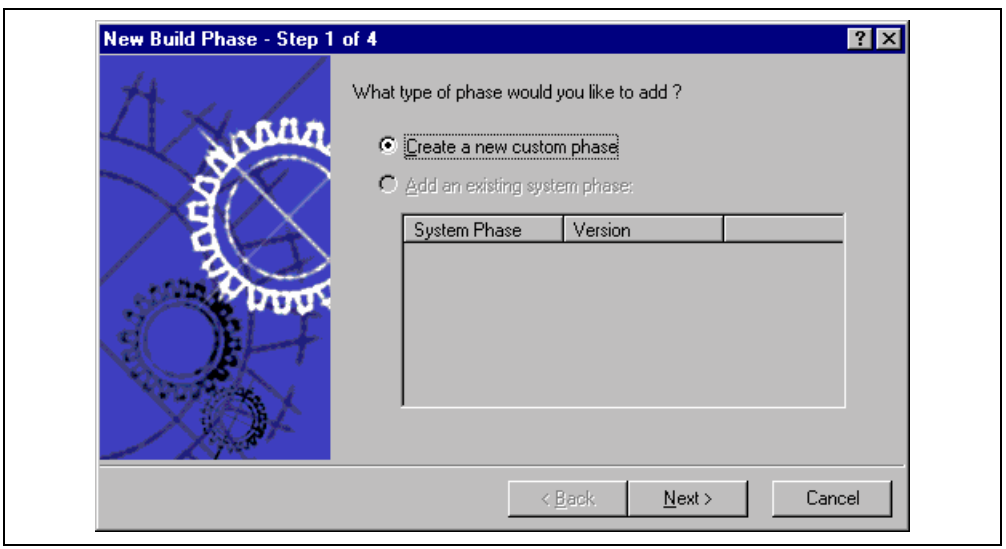


Figure 3.3a: New Build Phase Dialog (Step 1)

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

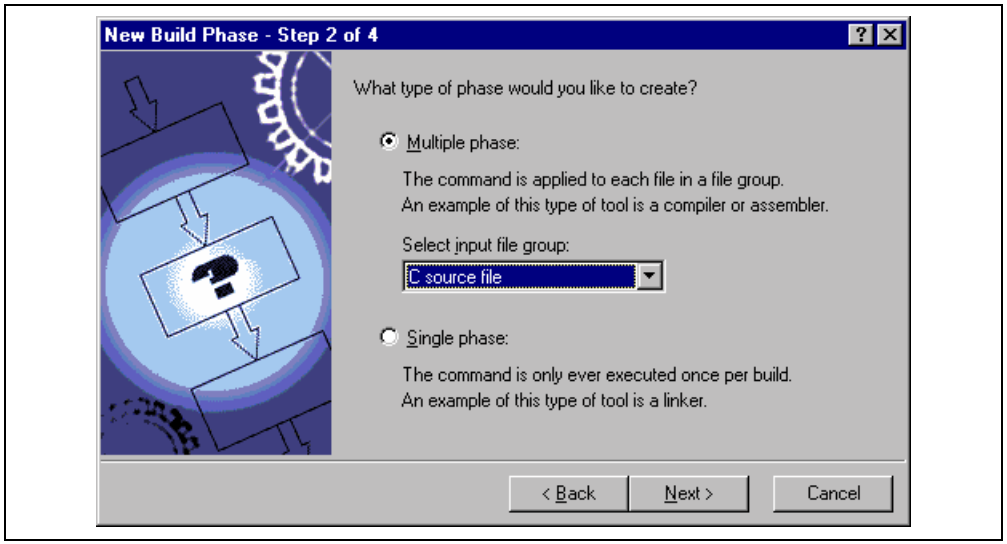


Figure 3.3b: New Build Phase Dialog (Step 2)

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

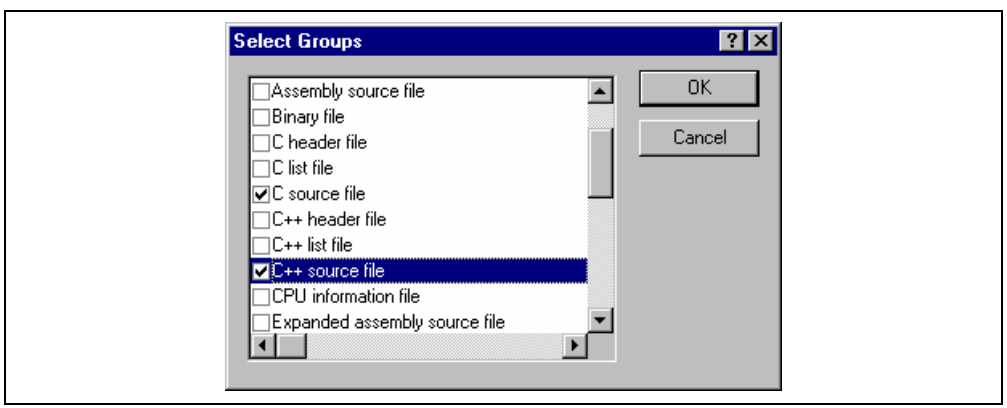


Figure 3.3c: Modify multiple input file groups

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

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

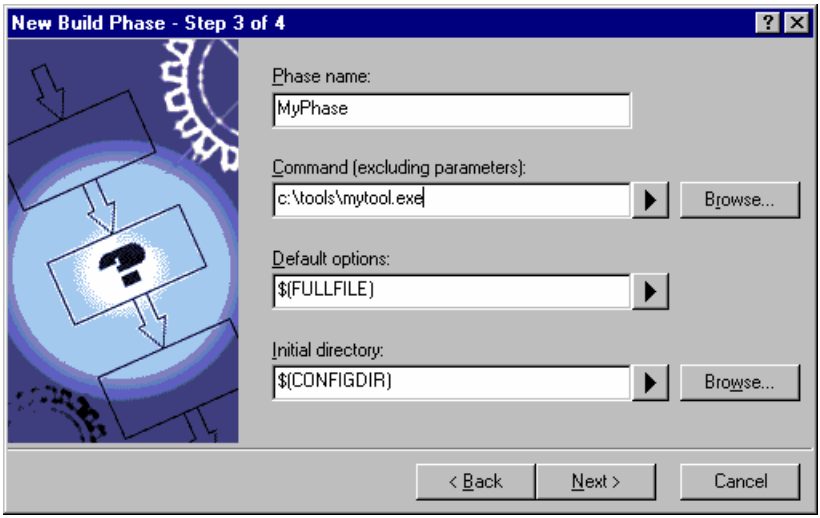


Figure 3.3d: New Build Phase Dialog (Step 3)

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

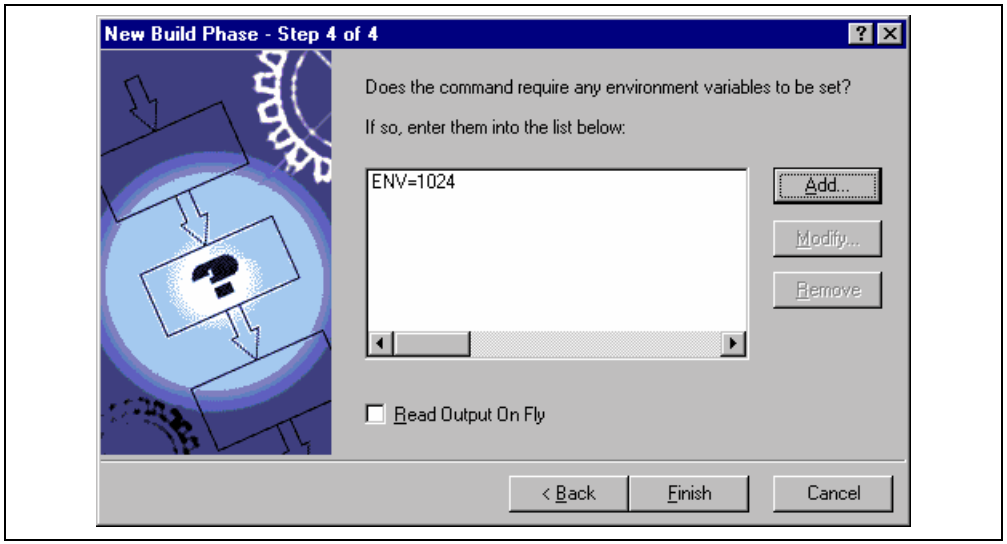


Figure 3.3e: New Build Phase Dialog (Step 4)

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

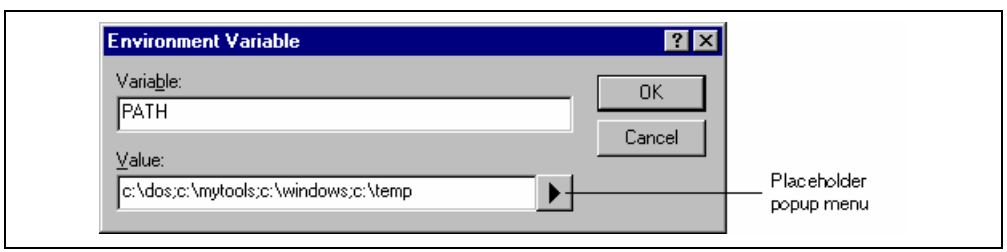


Figure 3.4: Environment Variable Dialog

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

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

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

3.3 Ordering Build Phases

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

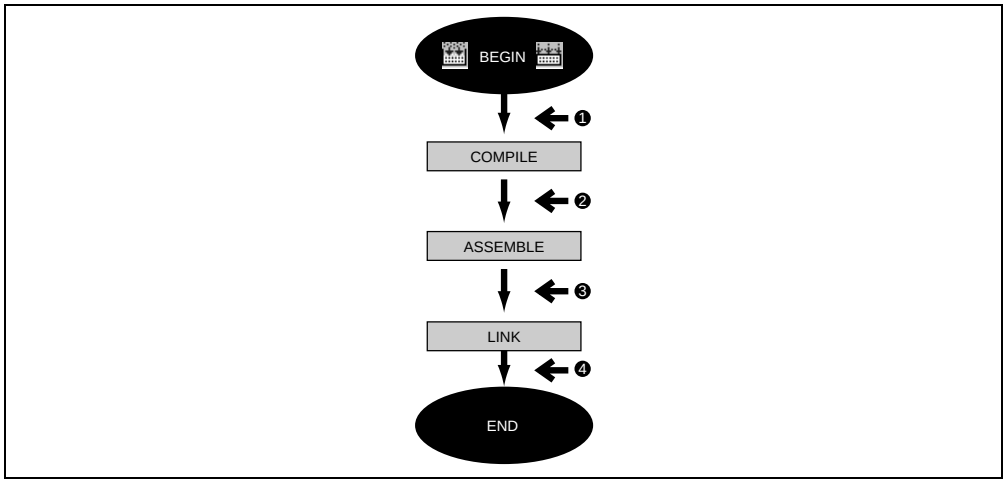


Figure 3.5: Typical Build Process

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

3.3.1 Build Phase Order

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

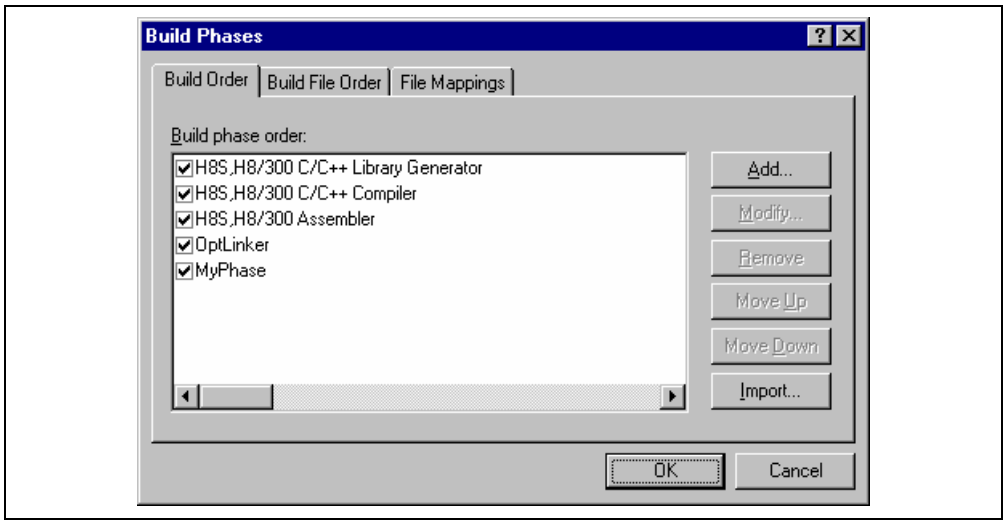


Figure 3.6: Build Phases Dialog Build Order Tab

In addition the following operations can be performed:

- To remove a phase:
  1. Select the phase that you would like to remove.
  2. Click the "Remove" button.
- To view the properties of a system phase:
  1. Select the system phase that you would like to examine.
  2. Click the "Modify…" button.
- To move a phase:
  1. Select the phase that you would like to move.
  2. Click the "Move Up" or "Move Down" button.
- To import a phase:
  1. Click the import button. A dialog is displayed which allows the user to browse to an existing project to import a custom phase from.
  2. Choose the location of the project you wish to import a custom phase from. Once selected a dialog is displayed which lists the custom phases in the imported project.
  3. Selecting a phase name and then clicking properties displays the custom phase details. This allows you to decide whether the phase does the functionality you require.
  4. Once you have decided which phase to import highlight it in the list and then click OK. The phase will then be added to the build phases dialog at the bottom of the build order.

➲ To modify a custom phase:

1. Select the custom phase that you would like to modify.
2. Click the "Modify…" button. The modify phase dialog will be invoked with the "Command" tab selected (figure 3.7).
3. Change the contents of the fields as appropriate.
4. Set the "Don't check for input file(s) existence before executing" check box if you don't want the HEW to abort the execution of the phase if any of the input files don't exist.
5. Select "Read Output On Fly" checkbox to display build output as it happens, rather than showing the output at the end of each phase execution.

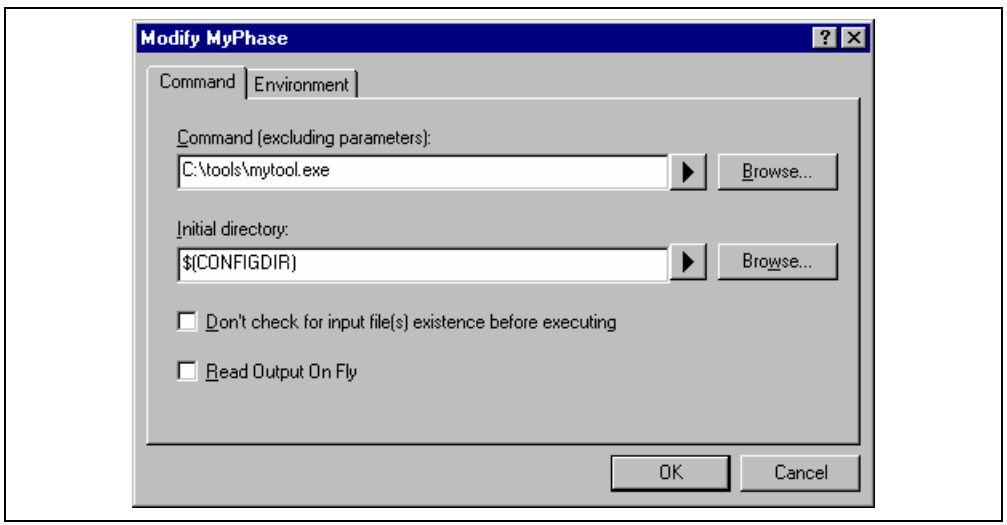


Figure 3.7: Modify Phase Dialog Command Tab

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

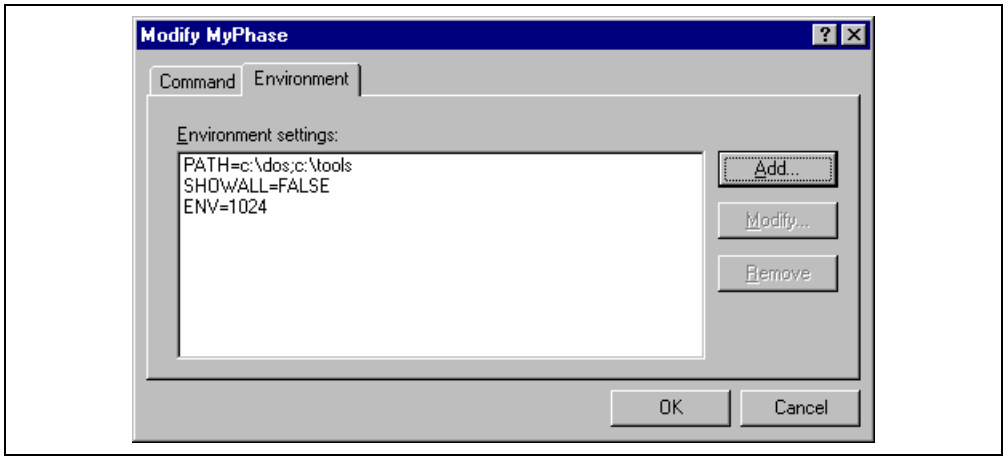


Figure 3.8: Modify Phase Dialog Environment Tab

3.3.2 Build File Phase Order

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

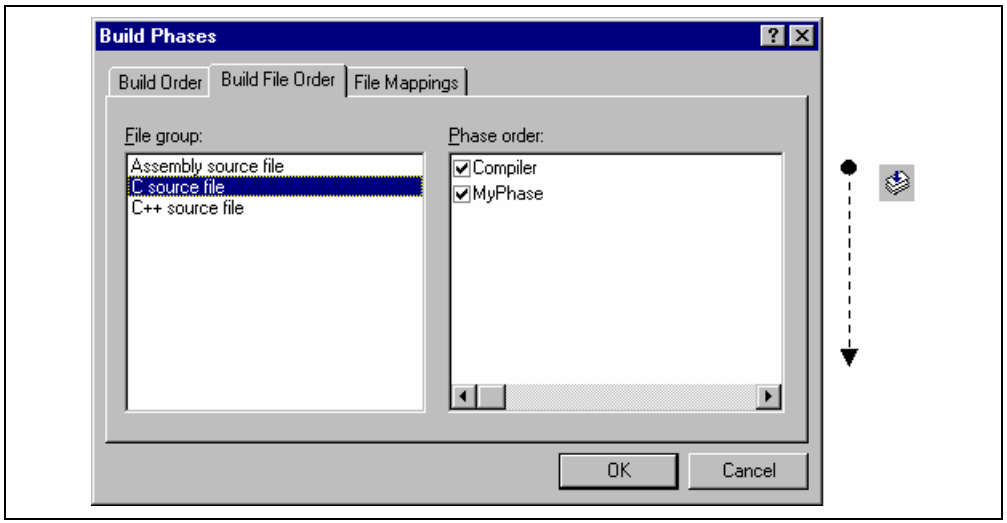


Figure 3.9: Build Phases Dialog Build File Order Tab

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

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

3.4 Setting Custom Build Phase Options

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

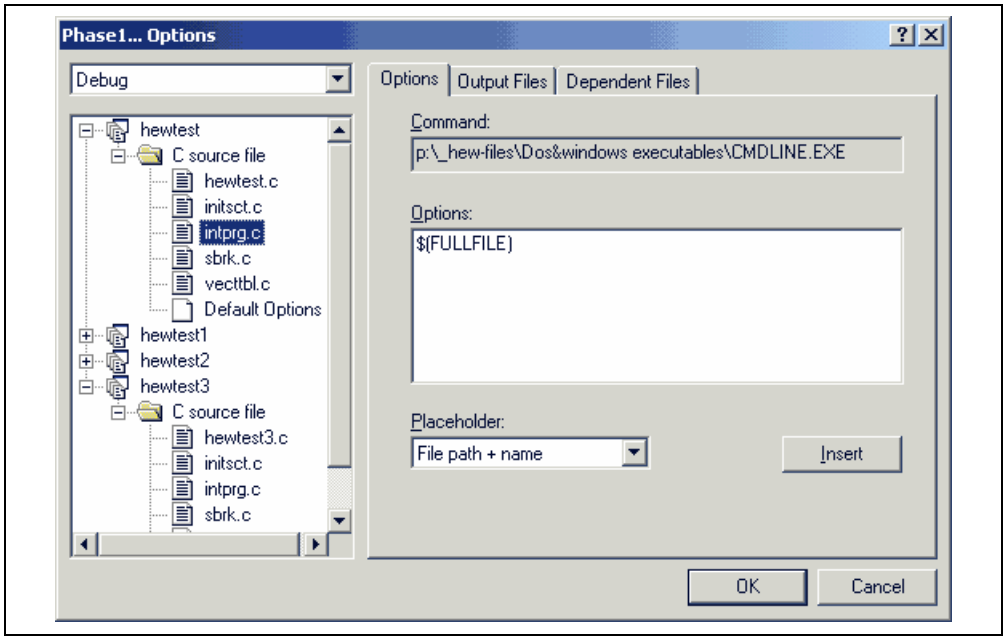


Figure 3.10: Custom Options Dialog

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

3.4.1 Options Tab

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

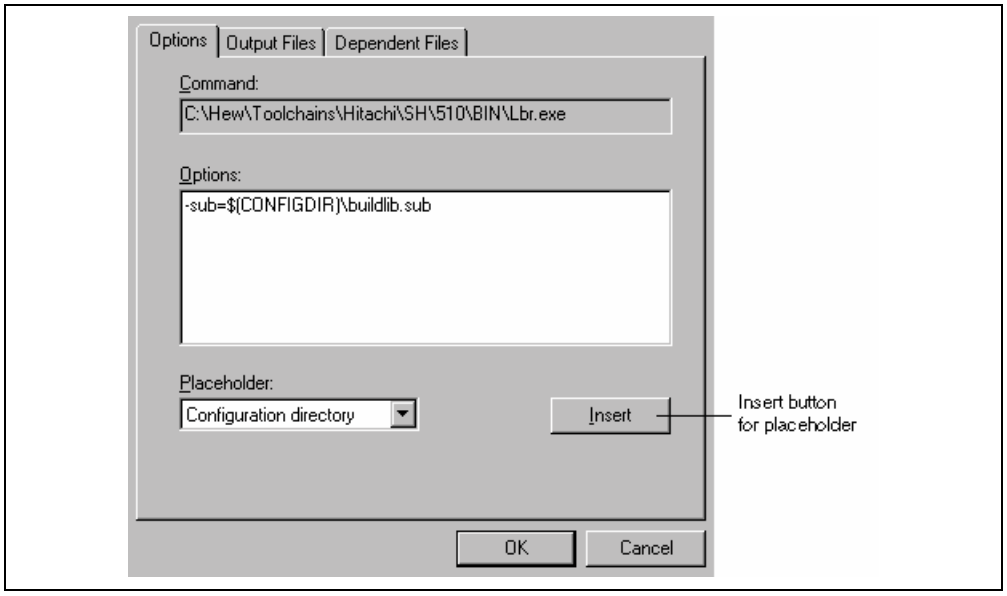


Figure 3.11: Custom Options Options Tab

3.4.2 Output Files Tab

The "Output Files" tab (figure 3.12) is where you can specify the output file or files that will be produced by the phase. Before each file is passed into this phase, the HEW checks that the output files are of a less recent date than the input file. If so, the phase will be executed for that file (i.e. input files have been modified since the output file or files were last produced). If the files are up to date then the phase will not be executed.

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

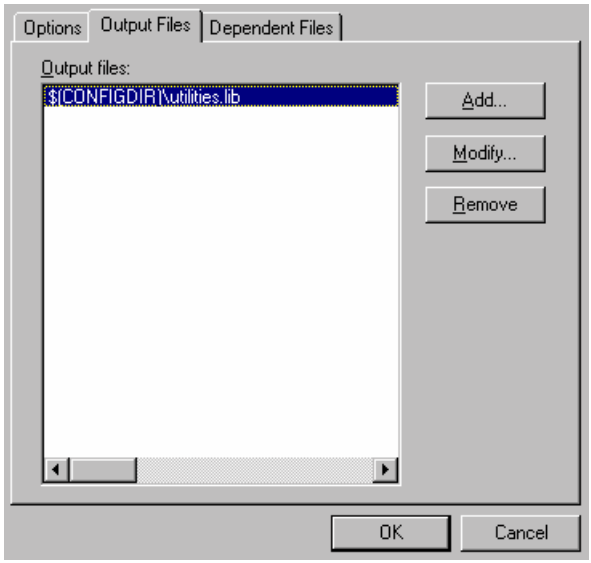


Figure 3.12: Custom Options Output Files Tab

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

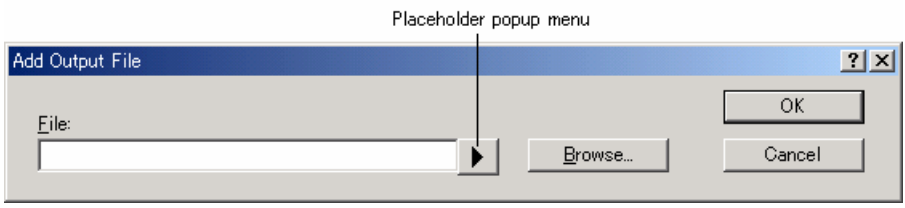


Figure 3.13: Add Output File Dialog

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

3.4.3 Dependent Files Tab

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

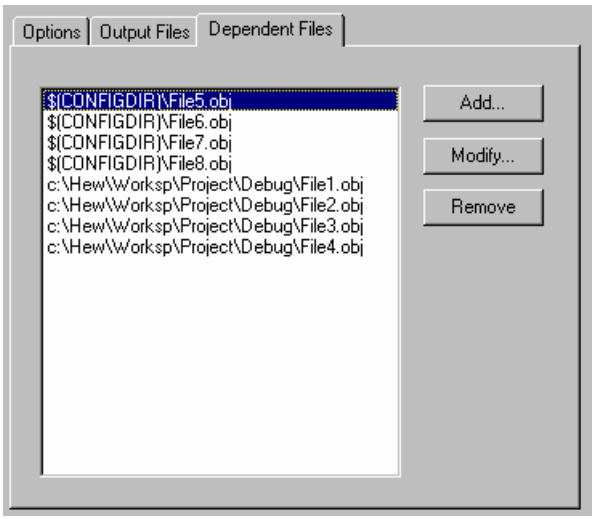


Figure 3.14: Dependent Files Tab in Custom Options

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

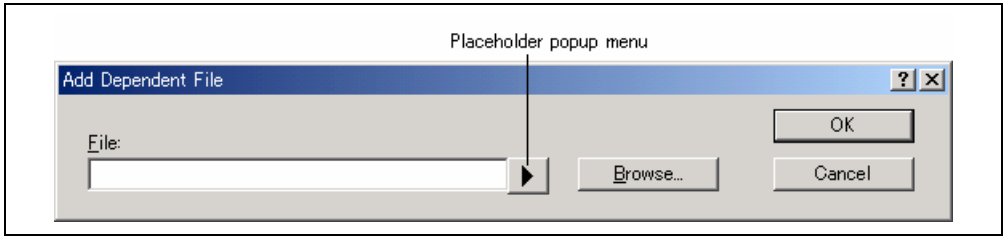


Figure 3.15: Add Dependent File Dialog

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

3.5 File Mappings

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

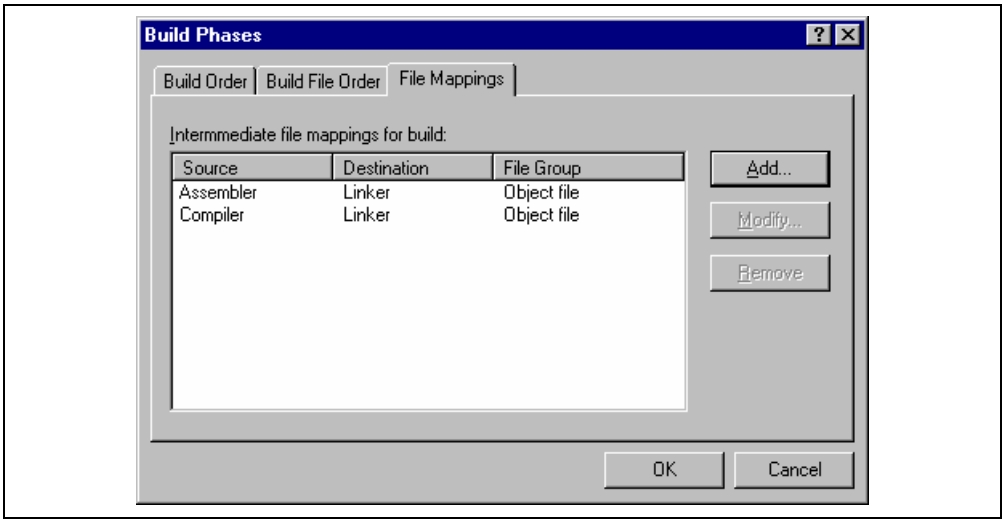


Figure 3.16: Build Phases Dialog File Mappings Tab

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

➲ To add a file mapping:

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

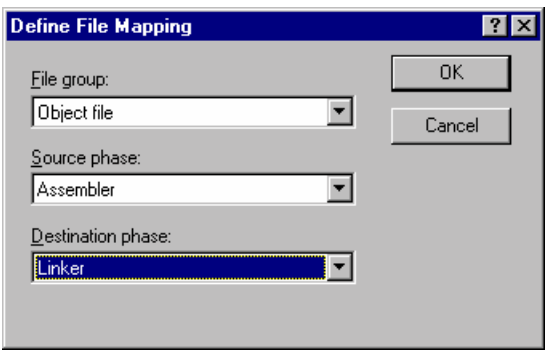


Figure 3.17: Define File Mapping Dialog

➲ To modify a file mapping:

1. Select the mapping to be modified.
2. Click "Modify…" button. The "Define File Mapping" dialog will be invoked (figure 3.17).
3. Modify the options as necessary.
4. Click "OK" to commit the changes.

3.6 Controlling the Build

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

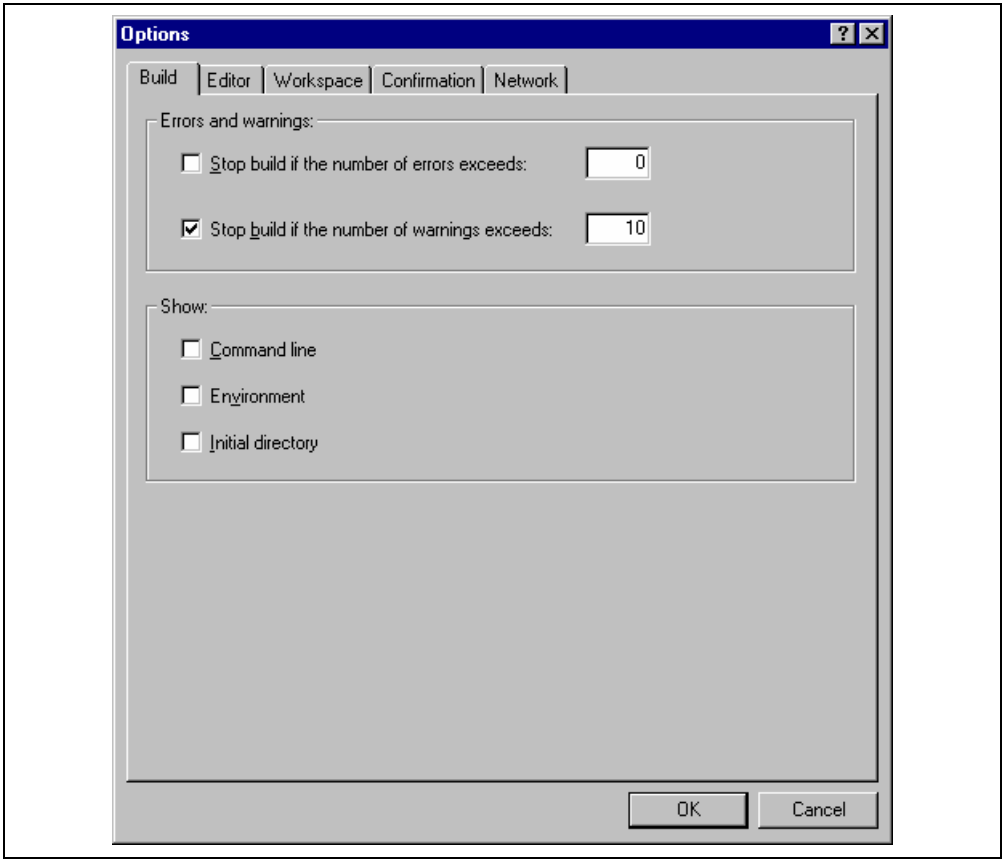


Figure 3.18: Options Dialog Build Tab

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

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

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

3.7 Logging Build Output

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

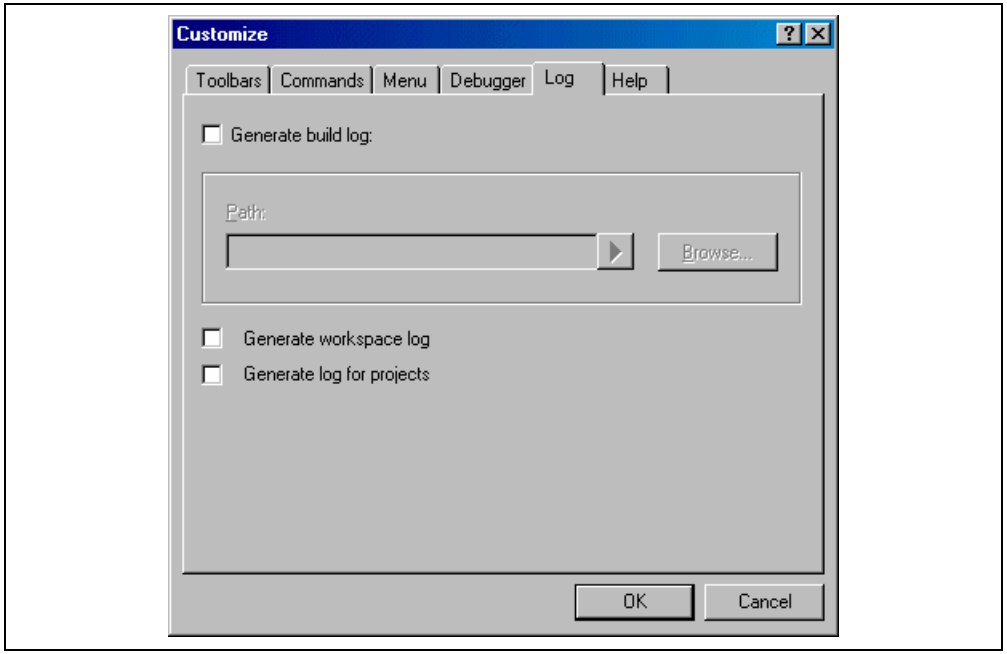


Figure 3.19: Tools Customize Dialog Log Tab

3.8 Changing Toolchain Version

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

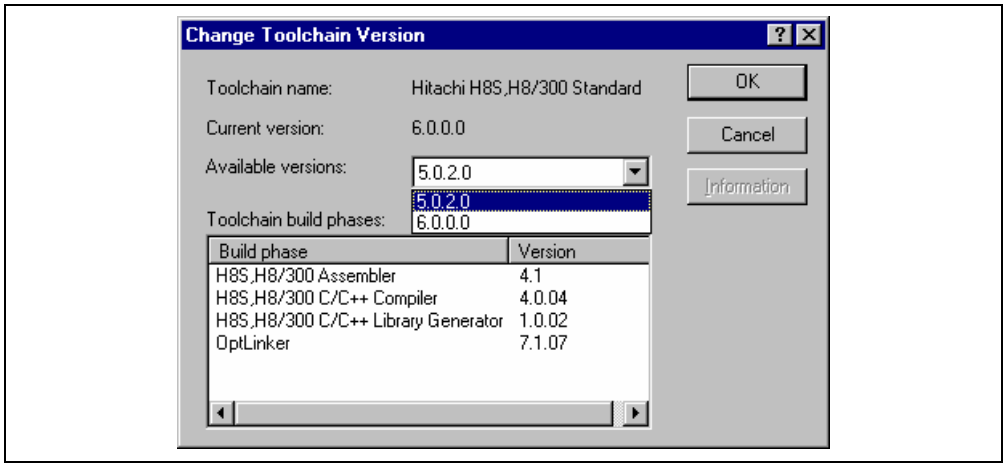


Figure 3.20: Change Toolchain Version Dialog

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

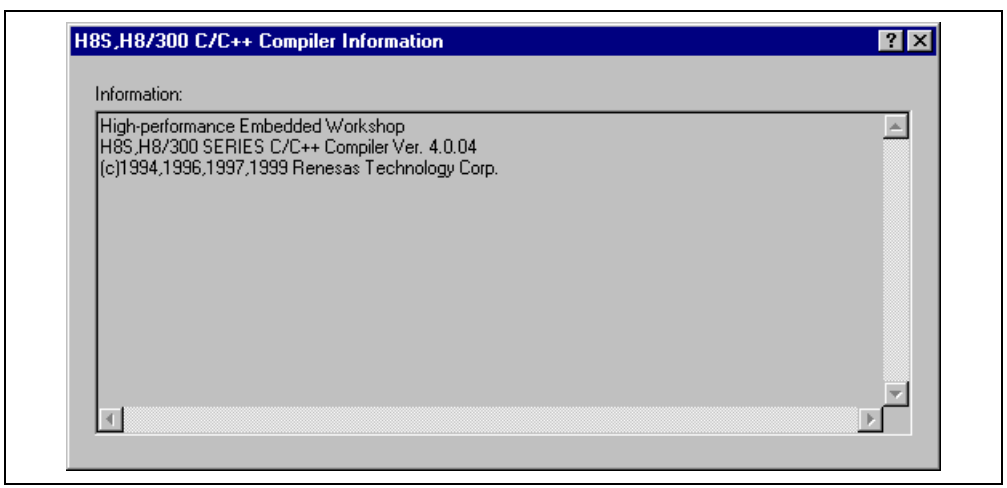


Figure 3.21: Toolchain Information Dialog

3.9 Using an External Debugger

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

The "Debugger" tab of the "Customize" dialog (figure 3.22) is where the High-performance Debugging Interface related information is configured. You may wish to use an older version of the debugger if certain targets are not currently supported in the new environment. Invoke it by selecting **[Tools->Customize…]** and then selecting the "Debugger" tab.

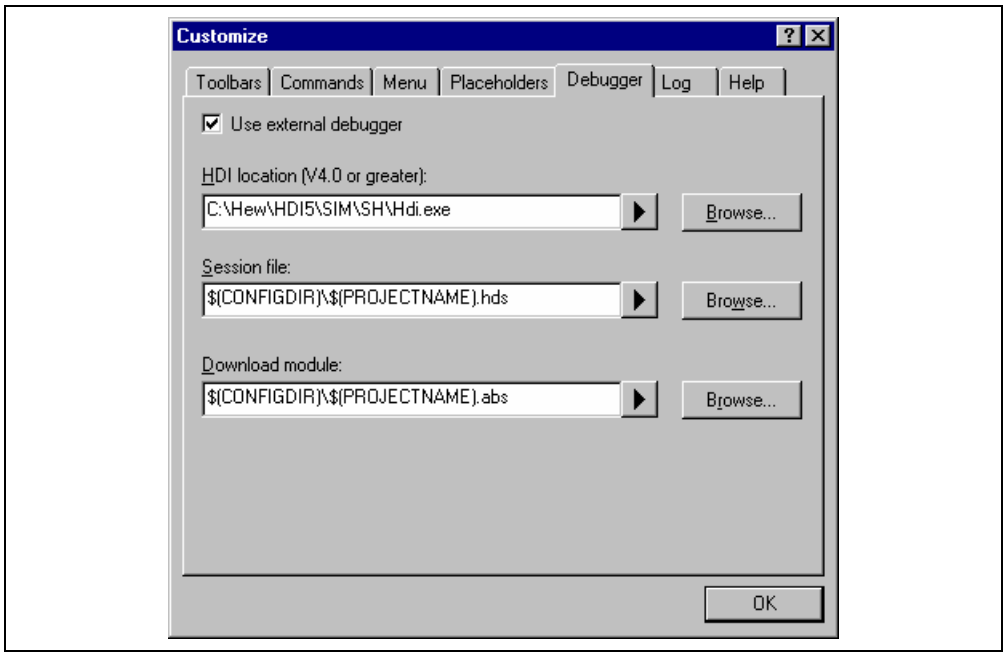


Figure 3.22: Customize Dialog Debugger Tab

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

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

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

3.10 Generating a Makefile

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

➲ To generate a makefile:

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

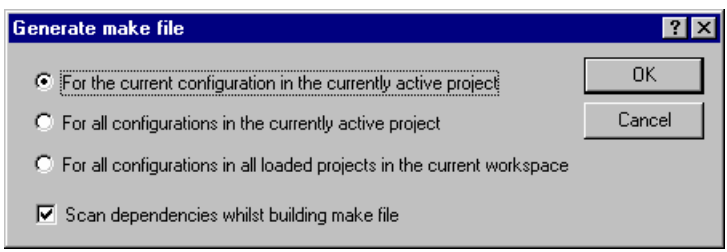


Figure 3.23: Generate makefile Dialog

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

➲ To execute a makefile:

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

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

4. Using the Editor

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

4.1 The Editor Window

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

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

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

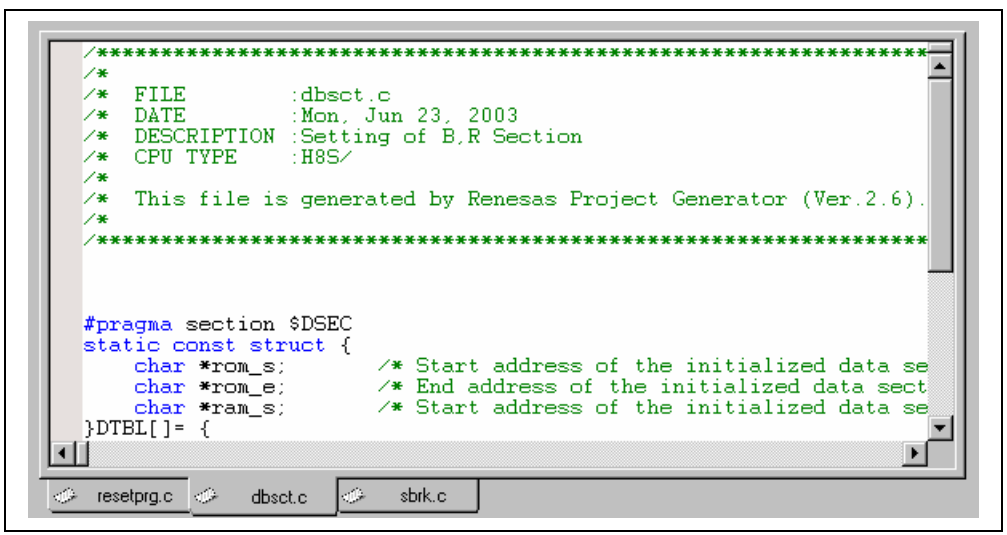


Figure 4.1: Editor Window

4.2 Working with Multiple Files

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

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

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

➲ To show files in notebook:

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

4.2.1 The Editor Toolbars

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

4.2.2 Editor Toolbar Buttons

New File

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

Open File

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

Save File

Saves the active source file.

Save All Files

Saves all of the files in the editor.

Print File

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

Cut

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

Copy

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

Paste

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

Find

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

Find in Files

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

Match Braces

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

Insert Template

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

Toggle Bookmark

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

4.2.3 Search Toolbar Buttons

Find in Files

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

Find

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

Find Next

Finds the next occurrence of the current search string.

Find Previous

Finds the previous occurrence of the current search string.

4.2.4 Bookmarks Toolbar Buttons

Toggle Bookmarks

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

Next Bookmark

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

Previous Bookmark

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

Clear All Bookmarks

Clears all bookmarks in the current file.

4.2.5 Templates Toolbar Buttons

Define Template

Specify template text for subsequent insertion.

Insert Template

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

4.3 Standard File Operations

4.3.1 Creating a New File

➲ To create a new editing window:

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

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

4.3.2 Saving a File

➲ To save the contents of an editing window:

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

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

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

4.3.3 Saving all Files

➲ To save the contents of every open editor window:

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

4.3.4 Opening a File

➲ To open a file:

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

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

➲ To open a recently used file:
Select the **[File->Recent Files]** menu option and from this sub-menu select the desired file.

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

4.3.5 Closing Files

➲ To close individual files select one of the following methods:

1. Double click on the editor window's system menu (located at the top left of each window when not maximized).
2. Click on the editor window's system menu (located at the top left of each window when not maximized) and select the "Close" menu option.
3. Ensure that the window that you want to close is the active window and then press **CTRL+F4**.
4. Ensure that the window that you want to close is the active window and then select **[File->Close]**.
5. Click on the close button (located at the top right of each window when not maximized).

➲ To close all windows at once:

Select **[Window->Close All]**.

4.4 Editing a File

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

Table 4.1 Basic Editing Operations

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

4.5 Searching and Navigating through Files

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

4.5.1 Finding Text

➲ To search for text in the current file:

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

Figure 4.2: Find Dialog

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

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

4.5.2 Finding Text in Multiple Files

➲ To search for text in many files:

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

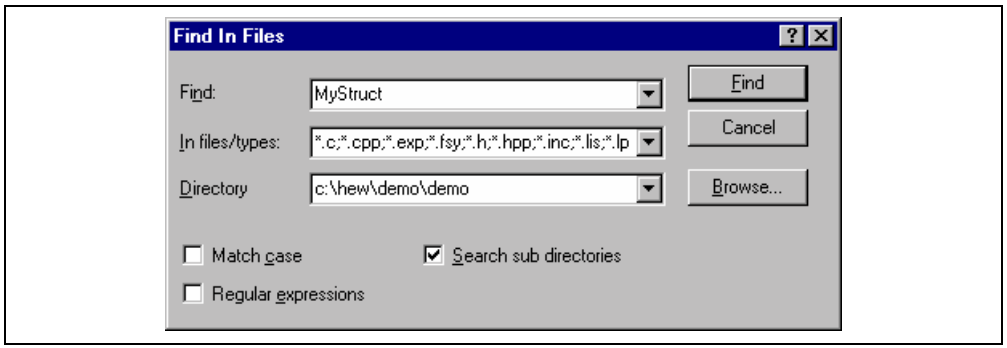


Figure 4.3: Find in Files Dialog

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

4.5.3 Replacing Text

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

➲ To replace text in a file:

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

5. Enter the text that you want to replace the search string with into the "Replace with" field, or select a previous replace string from the drop-down list box.

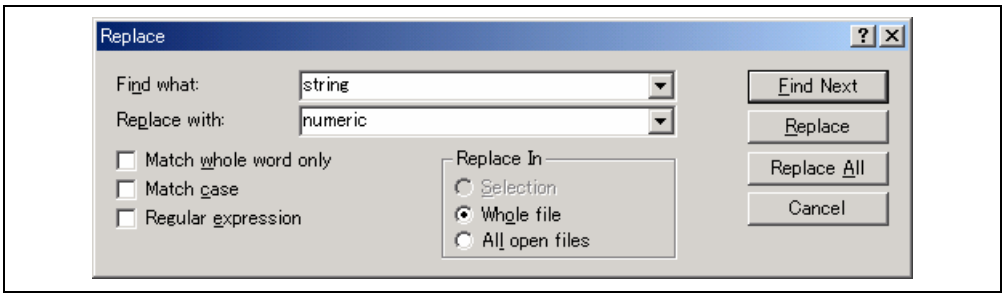


Figure 4.4: Replace Dialog

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

4.5.4 Jumping to a Specified Line

➲ To jump to a line in a file:

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

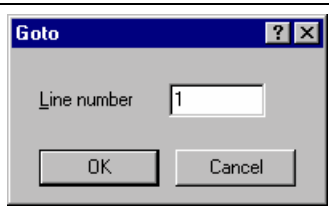


Figure 4.5: Goto Dialog

4.6 Bookmarks

When working with many large files at a time, it can become difficult to locate specific lines or areas of interest. Bookmarks enable you to specify lines that you want to jump back to at a subsequent time. One example of its

use is in a large C file where you may want to set a bookmark on each function definition. Once a bookmark has been set, it exists until it is removed or the file is closed.

➲ To set a bookmark:

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

➲ To remove a bookmark:

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

➲ To jump to the next bookmark in a file:

1. Ensure that the insertion cursor is somewhere within the file to be searched.
2. Select **[Edit->Bookmarks->Next Bookmark]**, press **F2** or select **[Bookmarks->Next Bookmark]** from the local menu or click the next bookmark toolbar button ().

➲ To jump to the previous bookmark in a file:

1. Ensure that the insertion cursor is somewhere within the file to be searched.
2. Select **[Edit->Bookmarks->Previous Bookmark]**, press **SHIFT+F2** or select **[Bookmarks->Previous Bookmark]** from the local menu or click the previous bookmark toolbar button ().

➲ To remove all bookmarks in a file:

1. Ensure that the window, whose bookmarks you want to remove is the active window.
2. Select **[Edit->Bookmarks->Clear All Bookmarks]** or select **[Bookmarks->Clear All Bookmarks]** from the local menu or click the clear all bookmarks toolbar button ().

4.7 Printing a File

➲ To print a file:

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

4.8 Configuring Text Layout

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

4.8.1 Page Set-up

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

➲ To set-up the page margins:

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

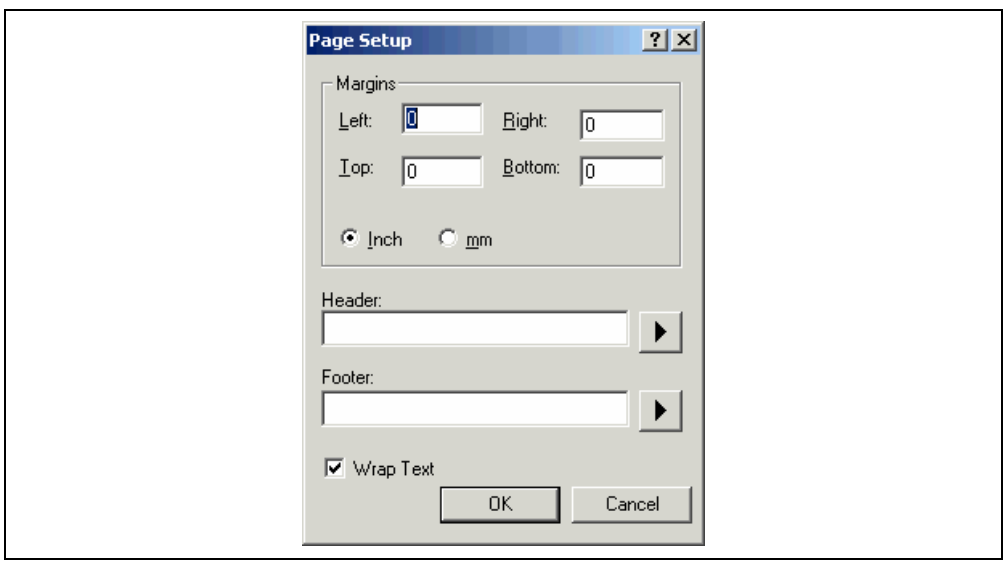


Figure 4.6: Page Setup Dialog

➲ To set-up the page header and footers:

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

➲ To set-up print wrapping

1. Select **[File->Page Setup...]**. The "Page Setup" dialog will be invoked (figure 4.6).

2. Click the wrap text check box. This switches on the wrap text facility when printing so no text is truncated and everything is visible.
3. Click "OK" for the new settings to take effect.

4.8.2 Changing Tabs

➲ To change tab size:

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

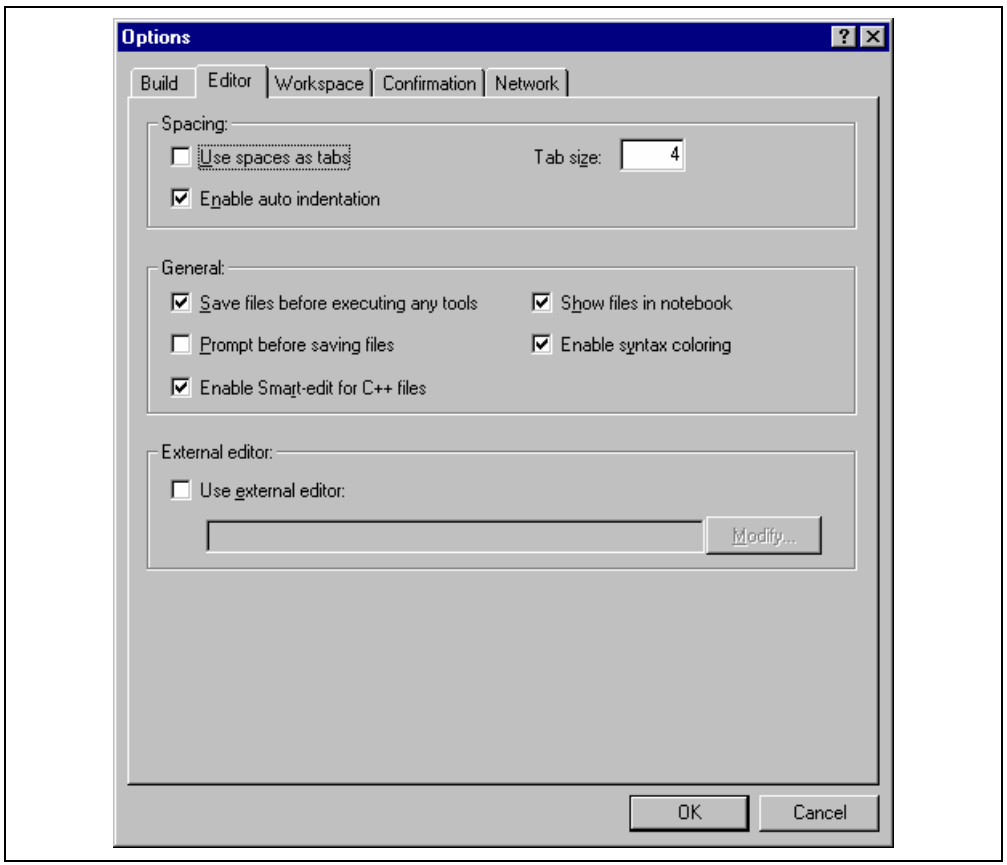


Figure 4.7: Options Dialog Editor Tab

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

➲ To use spaces as tabs:

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

4.8.3 Auto Indentation

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

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

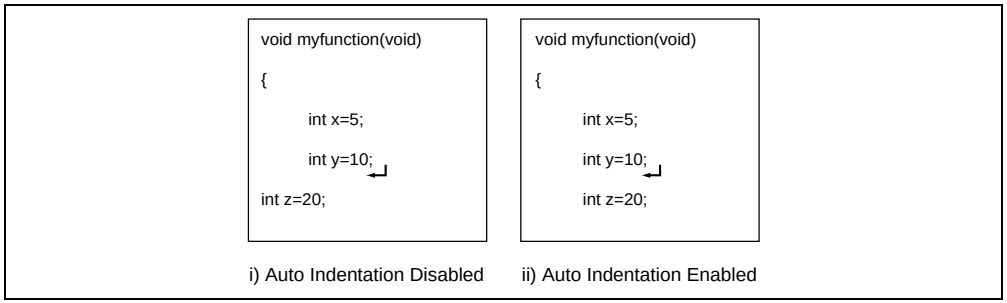


Figure 4.8: Effect of Auto Indentation

➲ To enable/disable Auto Indentation:

1. Select **[Tools->Options...]**. The "Options" dialog will be displayed. Select the "Editor" tab (figure 4.7).
2. Set the "Enable auto indentation" check box accordingly.
3. Click "OK" for the setting of the auto indentation check box to take effect.

4.9 Splitting a Window

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

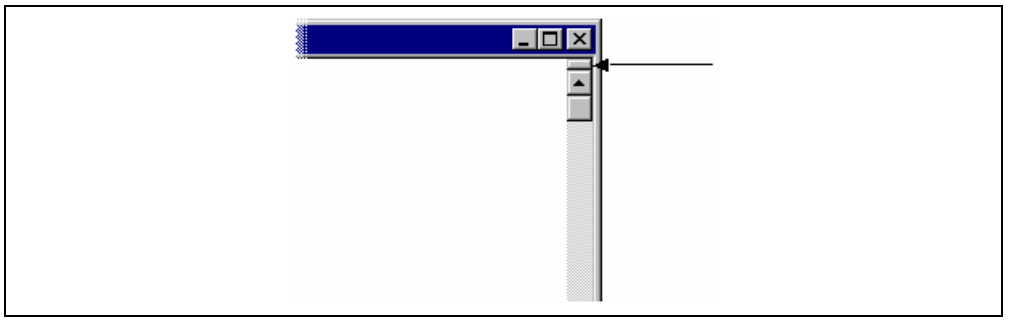


Figure 4.9: Split Bar Button

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

4.10 Configuring Text

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

4.10.1 Changing the Editor Font

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

➲ To change the editor font:

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

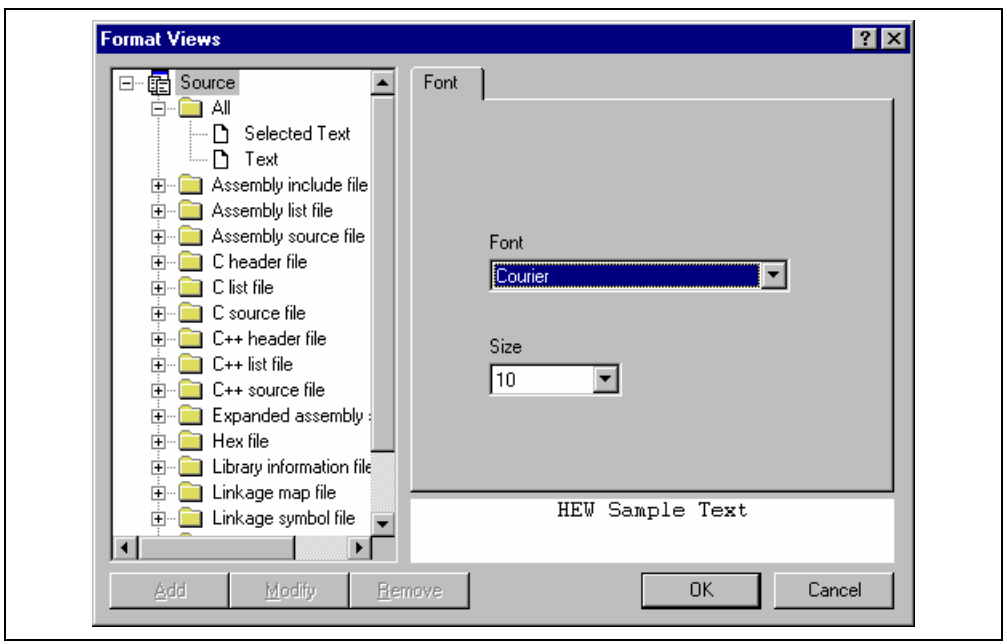


Figure 4.10: Format Views Dialog Font Tab

4.11 Syntax Coloring

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

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

➲

To change existing colors:

1. Select **[Tools->Format Views...]**. The "Format Views" dialog will be displayed.
2. Select the item underneath the icon in the tree you wish to modify the colour for. This should be the file type (e.g. C source file) and correct keyword group (e.g. identifier or pre-processor).
3. Select the "Colour" tab.
4. Modify the "Foreground" and "Background" color lists as desired. The color "System" refers to the current window foreground and background settings in control panel.
5. Click "OK" for the new colors to take effect.

➲ To create new keyword groups:

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

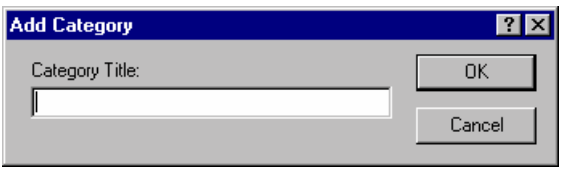


Figure 4.11: Add Category Dialog

➲ To create new keywords:

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

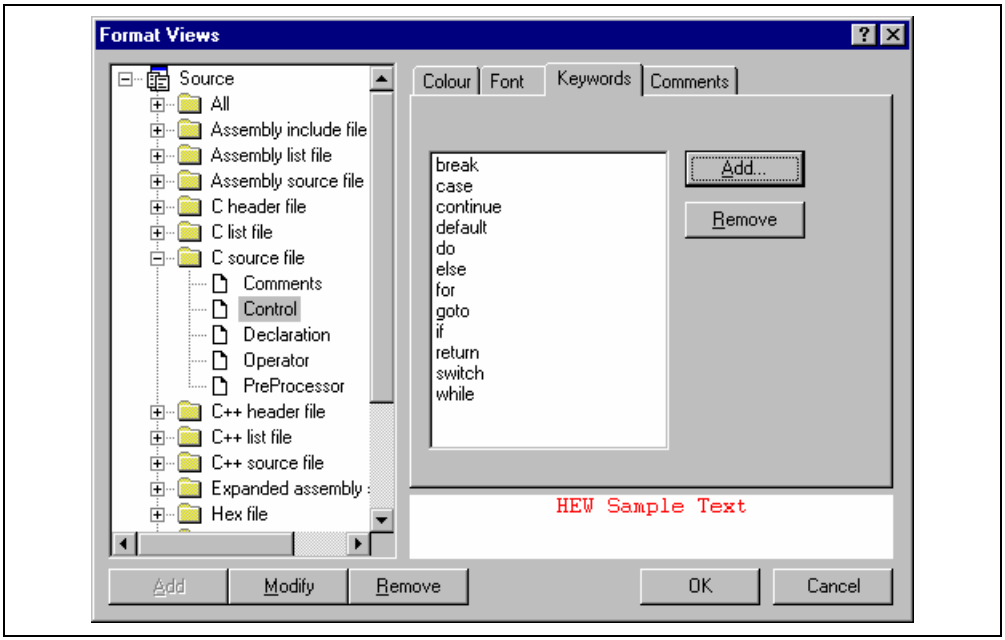


Figure 4.12: Format Views Dialog Keywords Tab

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

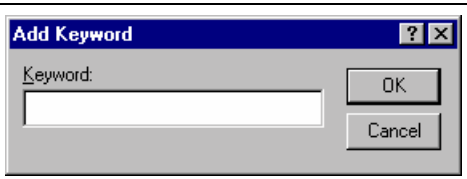


Figure 4.13: Add Keyword Dialog

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

➲ To disable/enable syntax coloring:

1. Select **[Tools->Options...]**. The "Options" dialog will be displayed. Select the "Editor" tab (figure 4.7).
2. Set the "Enable syntax coloring" check box as necessary and then click "OK".

4.12 Templates

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

4.12.1 Defining a Template

➲ To define a template:

1. Select **[Edit->Templates->Define Templates...]**, select **[Templates->Define Templates...]** from the local menu, press **CTRL+T** or click on the define template toolbar button (). The dialog shown in figure 4.14 will be displayed.
2. Click "Add". A dialog is displayed that asks you to enter your chosen template name. This name must be unique otherwise a duplicated template name message will be displayed and the template will not be added.
3. If you want to modify an existing template use the "Template name" drop down menu to select which template you want to modify.
4. Enter the desired text into the "Template text" text area. You can copy text from another editor window and then paste it into this dialog using **CTRL+V**.
5. There are 10 shortcut keys reserved for templates. If you want to designate one of these select the key in the drop list at the bottom of the edit template dialog. These range from CTRL+0 to CTRL+9.
6. Enter the following keywords to insert special information when the template is inserted:

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

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

User Input

When defining a template it is possible to define a user input field. Using the following placeholder specifies this $(USERINPUT<*n*>|<*"some text"*>). The n is a number, which identifies the user input identifier. These placeholders can be added manually but the user input button on the define templates dialog adds these placeholders in an automated manner. When the template is inserted into a file a dialog is displayed which allows you to enter some custom text for each of these fields. This text is then inserted instead of the placeholder. You can define ten of these user input fields.

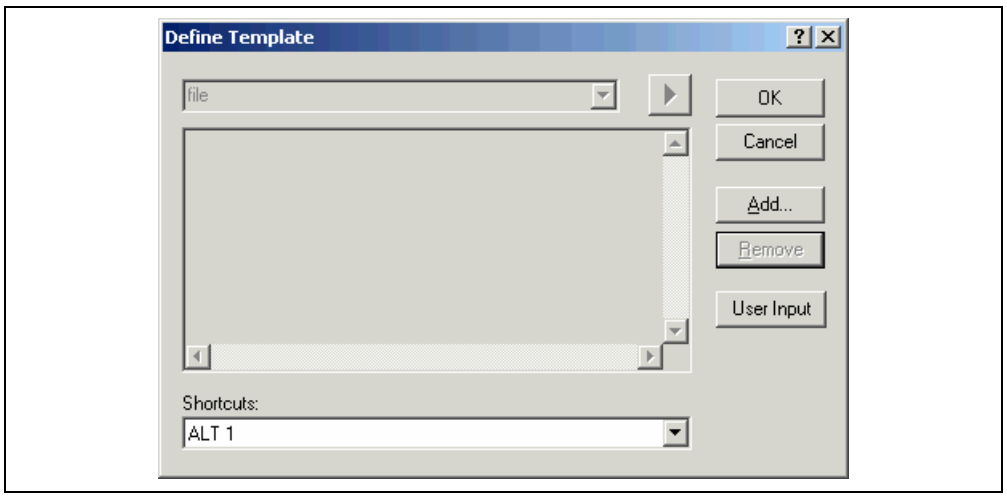


Figure 4.14: Define Template Dialog

A new addition to the HEW 3.0 is the templates view. This is shown in figure 4.15. This view is located with the workspace and navigation windows.

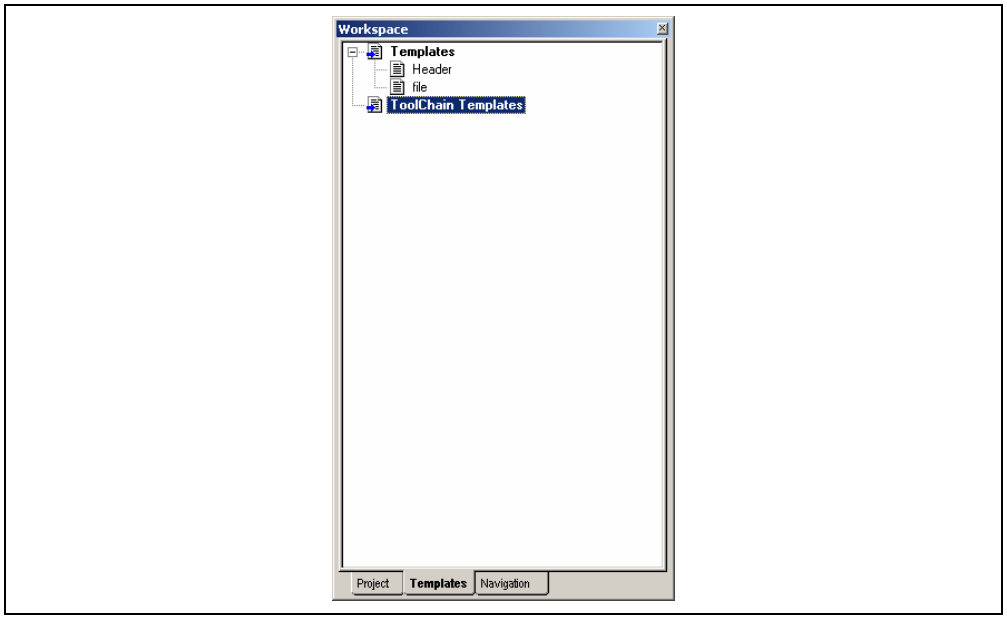


Figure 4.15: Templates view

Any new templates, which have been added to the HEW, are displayed under the templates folder. The toolchain templates folder is for templates, which are read only and have been provided for use in the HEW system by the current toolchain. Templates in this view can be dragged for insertion into an editor file. It is also possible to drag an area of text from the editor into the templates folder for quick template creation. Right clicking on this view displays a pop-up so that you can quickly add a new template, remove the current selection and edit the current selection.

4.12.2 Deleting a Template

➲ To delete a template:

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

4.12.3 Inserting a Template

➲ To insert a template:

1. Select a template in the toolbar, then click the insert template toolbar button ( ), select **[Edit-> Templates->Insert Template...]** or select **[Templates-> Insert Template...]** from the local menu. The dialog is dismissed and the chosen template is added to the current editor window.

Note: It is also possible to use the defined keyboard shortcut or drag the template from the templates view.

4.12.4 Brace Matching.

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

➲ To find a matching brace:

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

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

4.13 Editor Column management

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

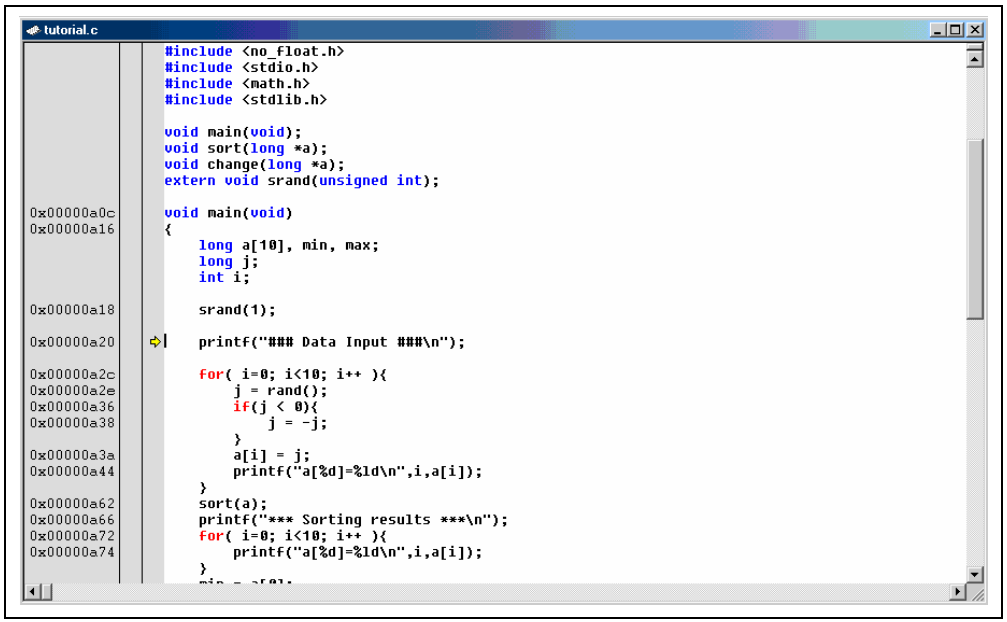


Figure 4.16: Editor columns

➲ To switch off a column in all source files:

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

➲ To switch off a column in one source files:

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

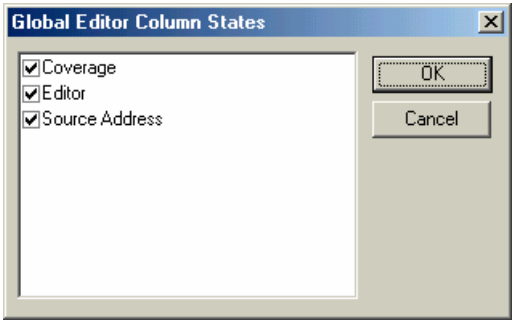


Figure 4.17: Global column state Dialog

5. Tools Administration

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

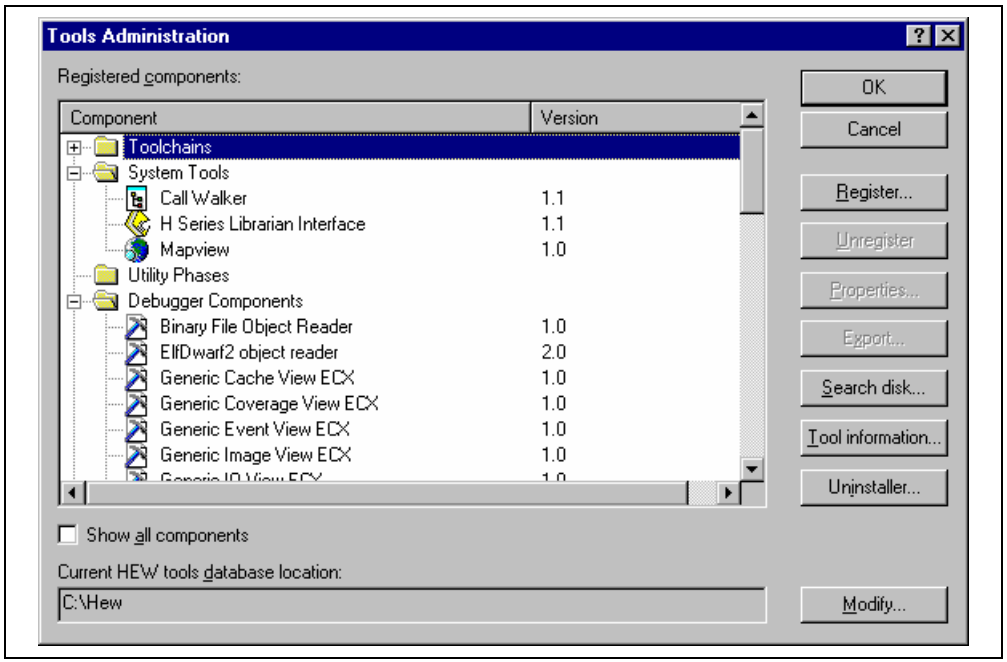


Figure 5.1: Tools Administration Dialog (Example)

There are five standard types of component:

- ***Toolchain*** - a set of build phases (e.g. compiler, assembler, and linker). These components provide the build capability.
- ***System Tool*** - an application (.EXE), which can be launched from the "Tools" menu. They are often provided as extra applications, which support the toolchain (e.g. an external debugger like the Hitachi Debugging Interface (HDI) or an interactive graphical librarian).
- ***Utility Phase*** - a "ready made" build phase which supports some specific build functionality (e.g. analyze complexity of source code, count lines of source code, etc.). These components provide added functionality to the build that is not toolchain specific.
- ***Debugger Component*** – a component that supports some specific debugger functionality (e.g. Target platform, Object reader, etc).
- ***Extension Component*** – a component that provides key functionality in a certain area of the HEW system. These components cannot be unregistered when installed (e.g. The HEW builder, debugger and flash support).

5.1 Tool Locations

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

5.2 HEW Registration Files (*.HRF)

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

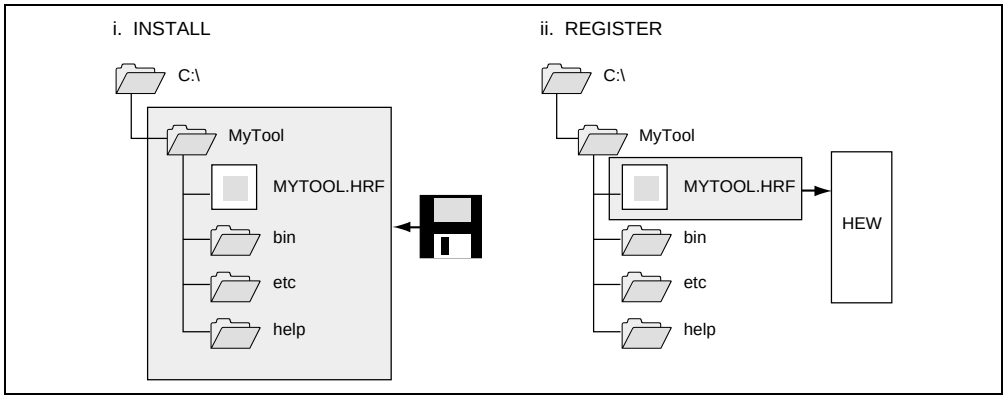


Figure 5.2: HRF File Location and Registration

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

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

➲ To change the tools location:

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

5.3 Registering Components

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

5.3.1 Searching Drives for Components

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

➲ To search for components:

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

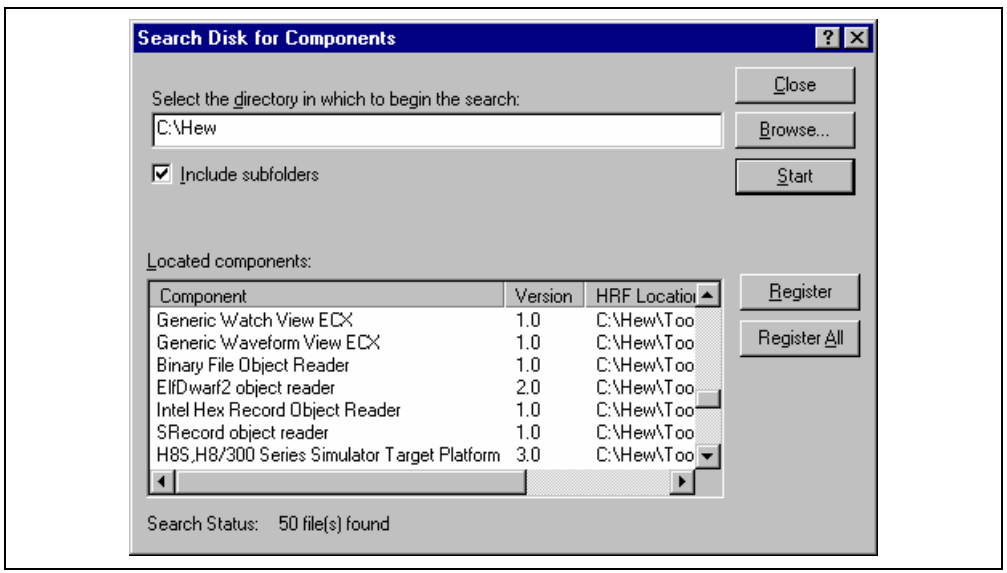


Figure 5.3: Search Disk for Components Dialog

2. Enter the directory in which you would like to search into the top field or browse to it graphically by clicking the "Browse…" button.
3. Check the "Include subfolders" check box if you would like to search the directory specified and all directories below it.
4. Click the "Start" button to begin the search. During the search, the "Start" button will change to a "Stop" button. Click the "Stop" button to halt the search at any time.
5. The results of the search are shown in the "Located components" list. Select a component and click "Register" to register an individual component or click "Register All" to register all located components.
6. Click "Close" to exit the dialog.

5.3.2 Registering a Single Component

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

➲ To register a component:

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

5.4 Unregistering Components

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

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

5.5 Viewing and Editing Component Properties

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

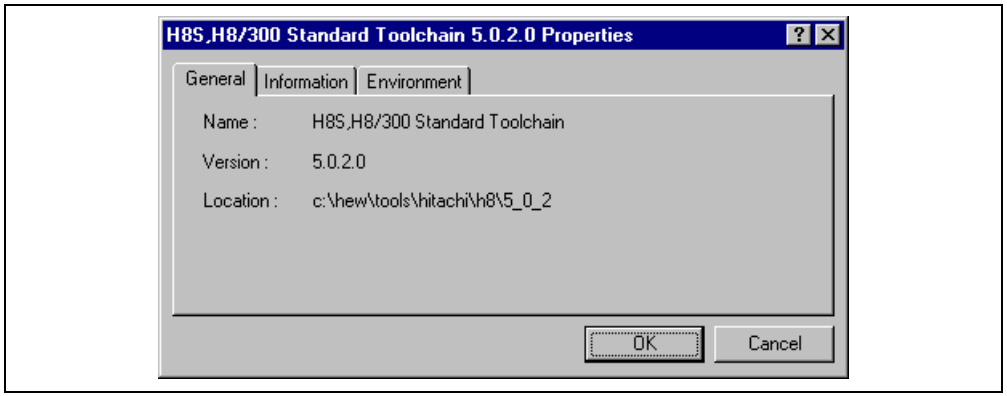


Figure 5.4: Properties Dialog General Tab

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

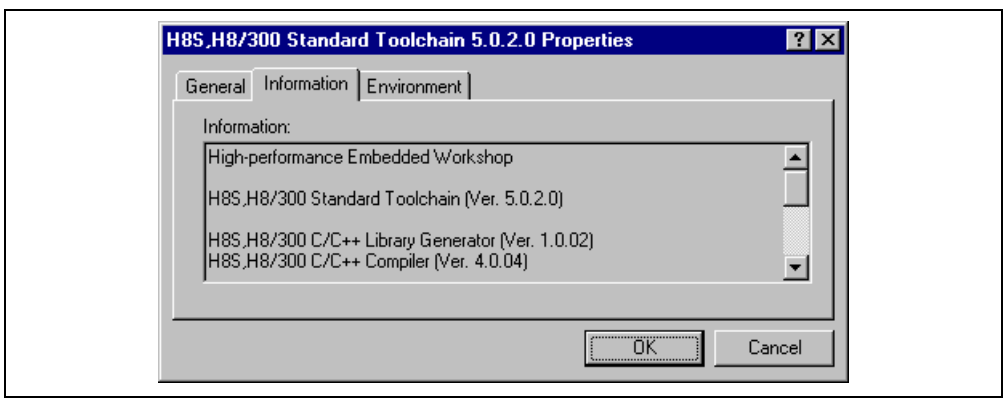


Figure 5.5: Properties Dialog Information Tab

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

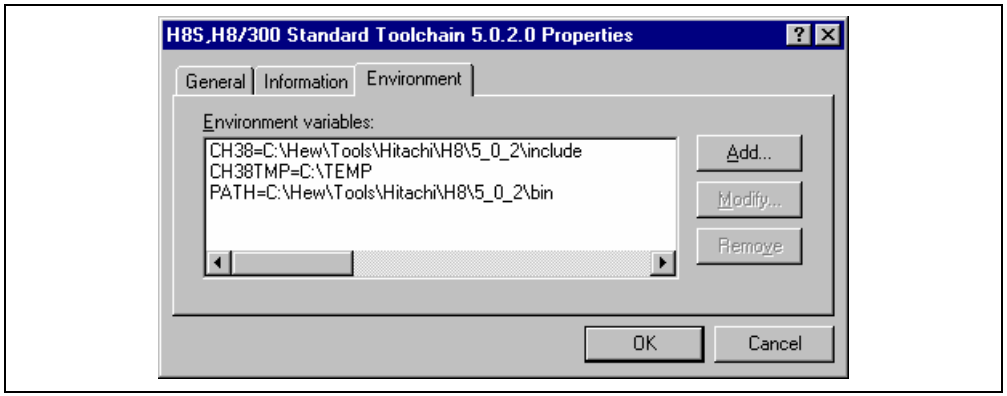


Figure 5.6: Properties Dialog Environment Tab

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

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

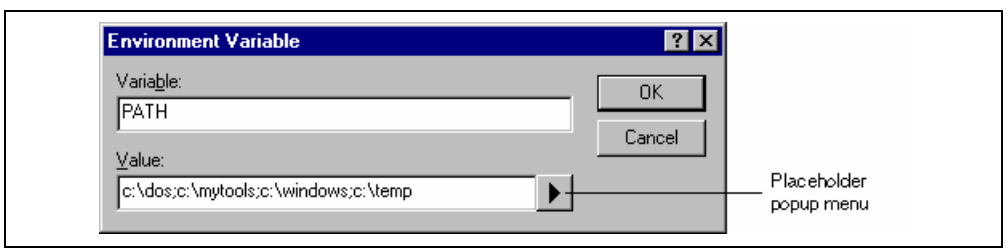


Figure 5.7: Environment Variable Dialog

5.6 Uninstalling Components

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

➲ To uninstall a component:

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

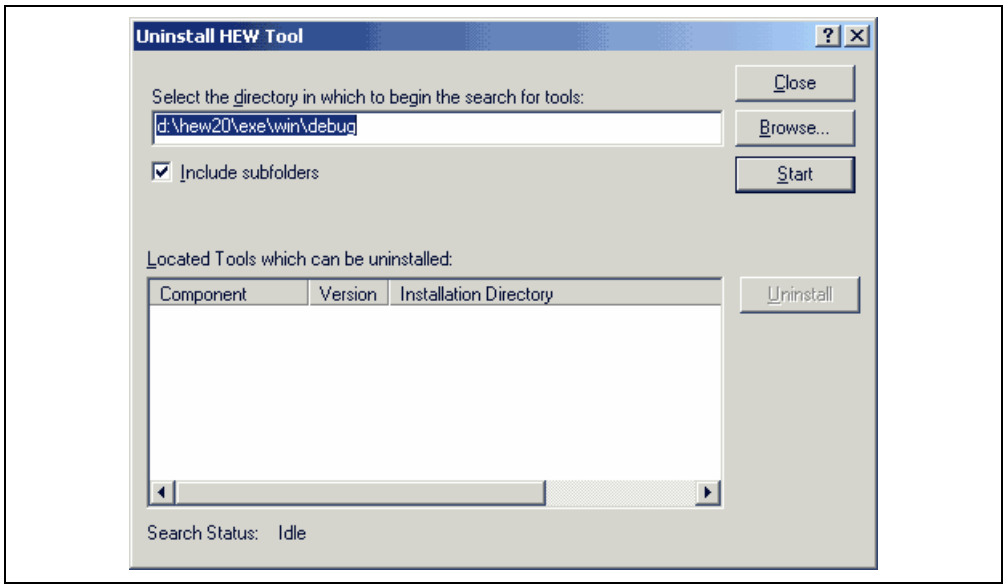


Figure 5.8: Uninstall HEW Tool

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

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

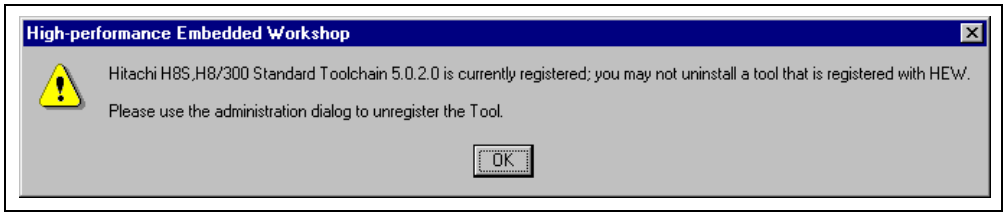


Figure 5.9: Unable to Uninstall Tool

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

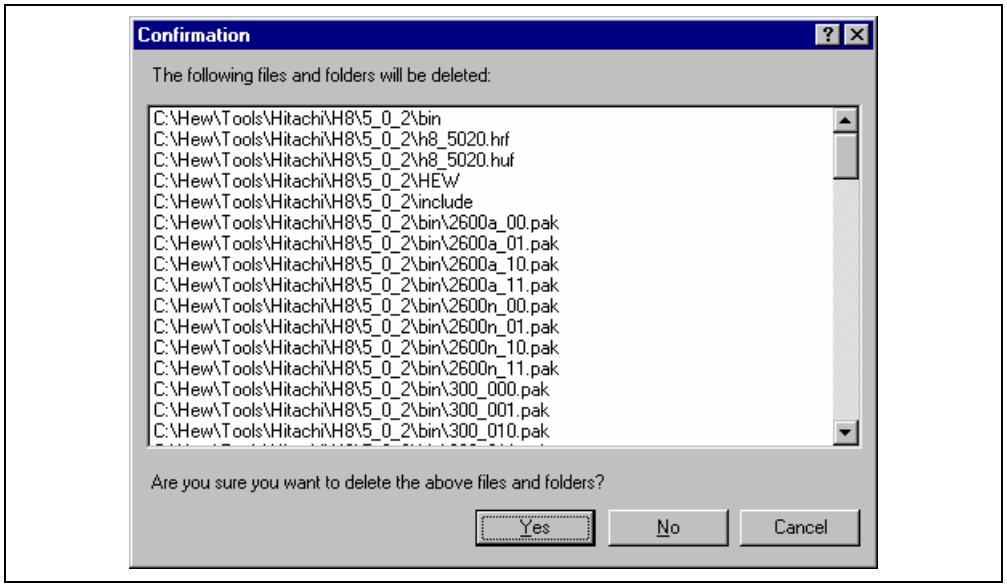


Figure 5.10: Confirmation Dialog

5.7 Technical Support Issues

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

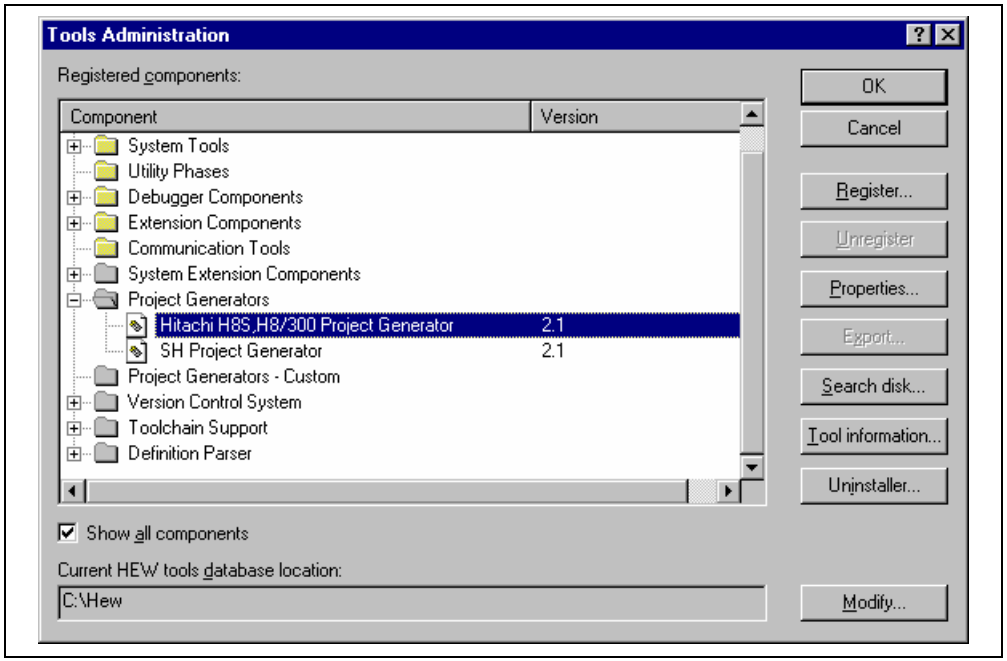


Figure 5.11: All Components Shown

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

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

➲ To output tool information:

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

If any of the components have problems these can be seen in the tools administration dialog. If the icon has an additional icon this explains the problem. There are two additional icons that can be displayed. If a component is found but cannot be used due to it being an old version or another dependent component is not available then

the icon in figure 5.12 is used to show this. If the component is not located where the registration file says it is then the icon in figure 5.13 is used to show this.

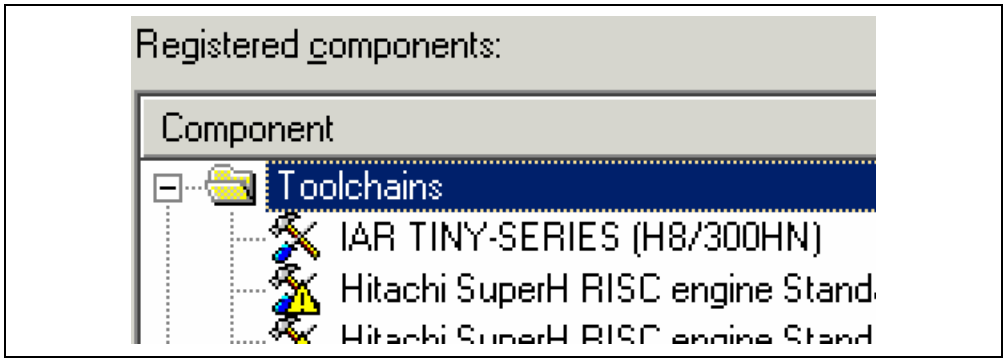


Figure 5.12: Incompatible component found icon

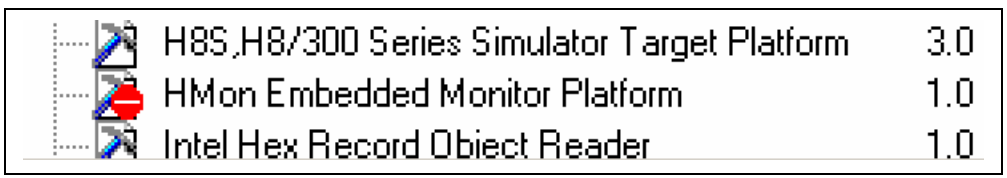


Figure 5.13: Component not found icon

5.8 On-Demand components

The HEW version 3.0 onwards has the concept of on-demand components. These components are not automatically loaded by the application or the debugger component. These components can be loaded by the user or as part of the project generation process.

➲ To load or unload an on-demand component manually:

1. Click the **[Project->Components…]** menu item.
2. The component gallery dialog is displayed. This is displayed in figure 5.14.
3. Select the component you wish to load. Click the load button. The components image should change to the loaded state.
4. If you wish to unload a component. Select the component. Click the unload button. The components image should change to the unloaded state.
5. Click OK to verify the changes.

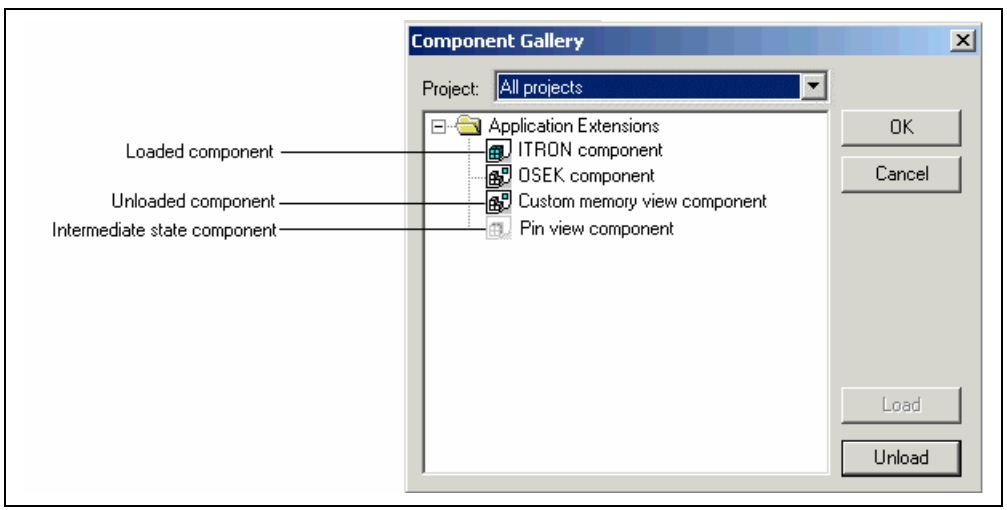


Figure 5.14: Component Gallery

Note: Each project in your workspace can have different components loaded and unloaded. If you have multiple projects you can use the "Multiple projects…" and "All projects" items to change a components load status over more than one project. If you select a combination which means the component is loaded in one project and not anther then the intermediate state icon is displayed.

5.9 Custom Project Types

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

6. Customizing the Environment

6.1 Customizing the Toolbar

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

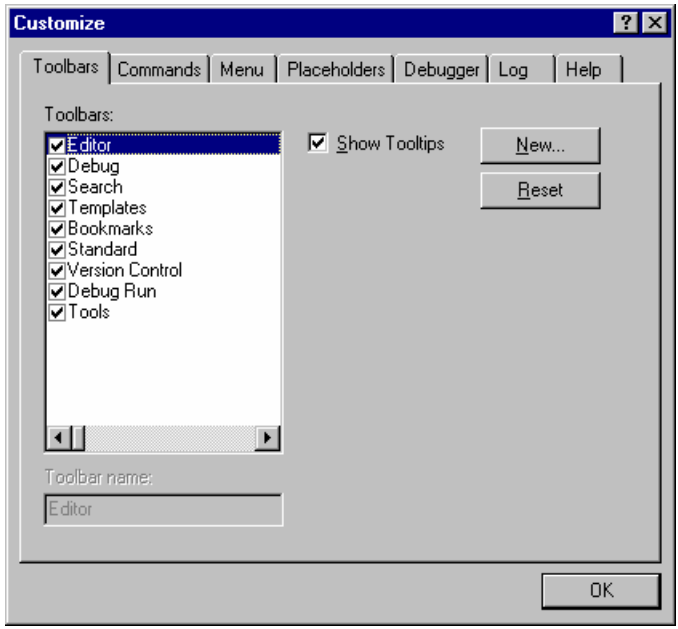


Figure 6.1: Customize Dialog Toolbars Tab

➲ To create a new toolbar:

1. Select **[Tools->Customize…]**. The dialog shown in figure 6.1 will be displayed.
2. Click the "New..." button. The dialog shown in figure 6.2 will be displayed.
3. Enter the name of the new toolbar into the "Toolbar name" field.
4. Click "OK" to create the new toolbar.

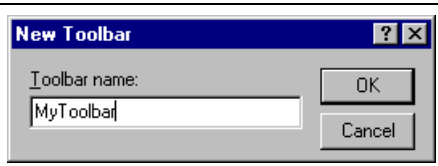


Figure 6.2: New Toolbar Dialog

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

➲ To add buttons to a toolbar:

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

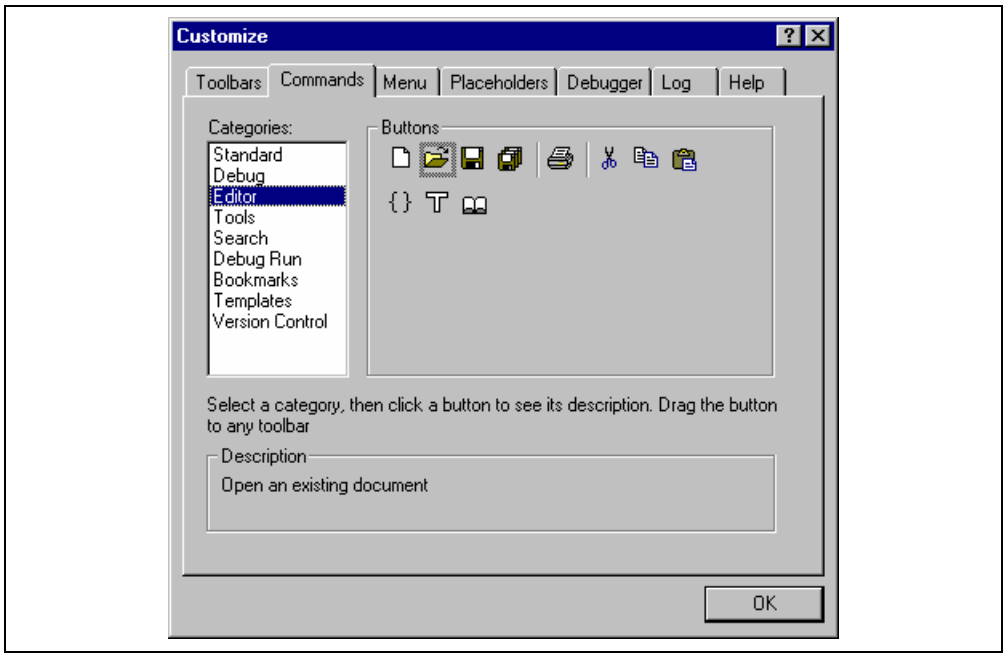


Figure 6.3: Customize Dialog Commands Tab

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

6.2 Customizing the Tools Menu

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

➲ To add a new menu option:

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

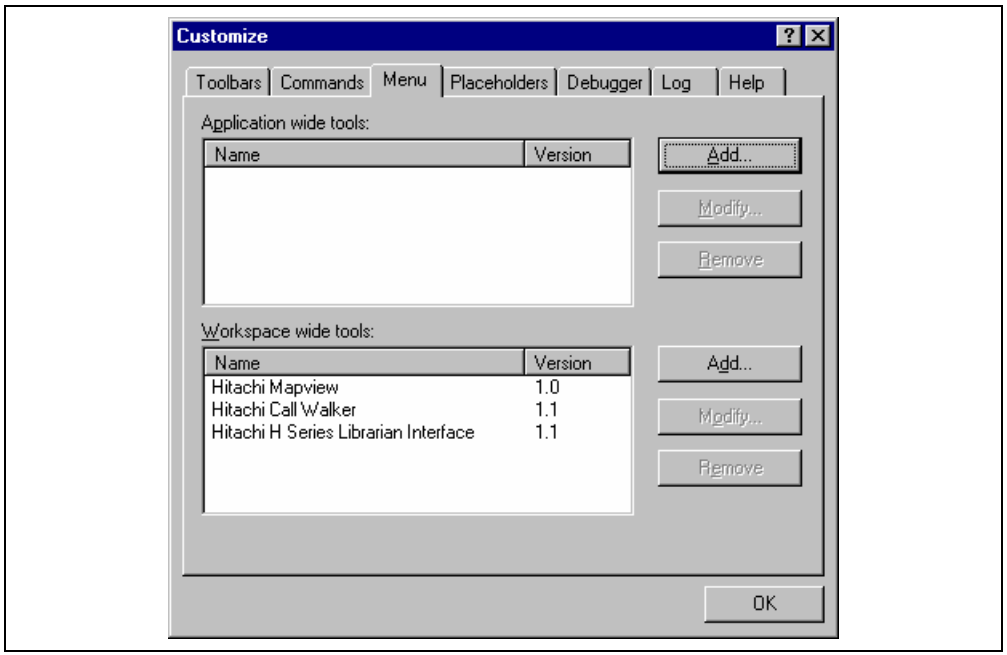


Figure 6.4: Customize Dialog Menu Tab

2. Click the "Add…" button (the dialog shown in figure 6.5 will be invoked). If you would like to add an existing system tool to the menu then select the "Select from existing system tools" radio button, choose the tool from the drop-down list and then click "OK". Alternatively, if you would like to add a tool of your own then follow the remaining steps.
3. Enter the name of the tool into the "Name" field.
4. Enter the command, excluding arguments, into the "Command" field.
5. Enter any arguments that you would like to pass to the command into the "Arguments" field.
6. Enter an initial directory in which you would like the tool to run, into the "Initial directory" field.
7. Click "OK" to add the menu option to the "Tools" menu.

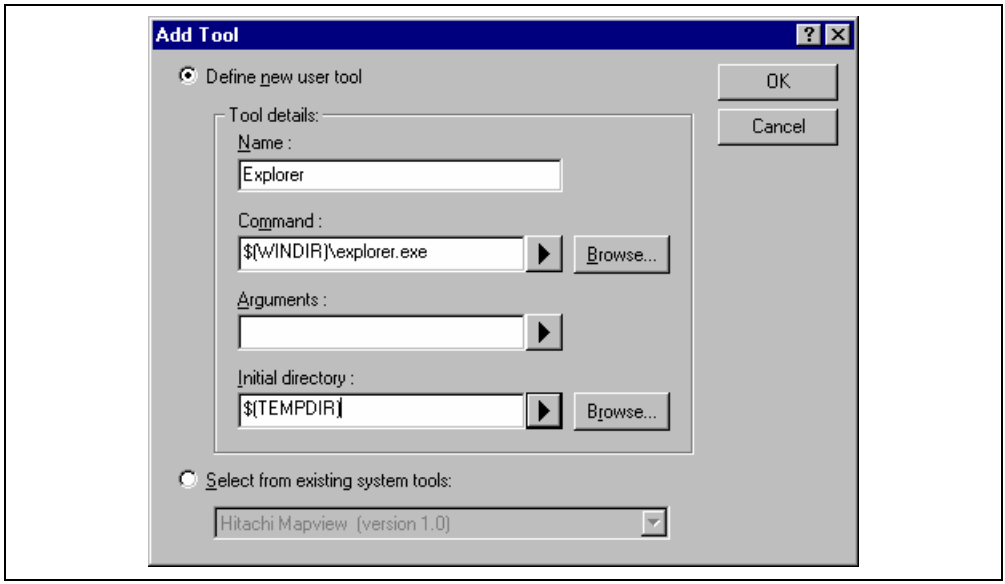


Figure 6.5: Add Tool Dialog

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

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

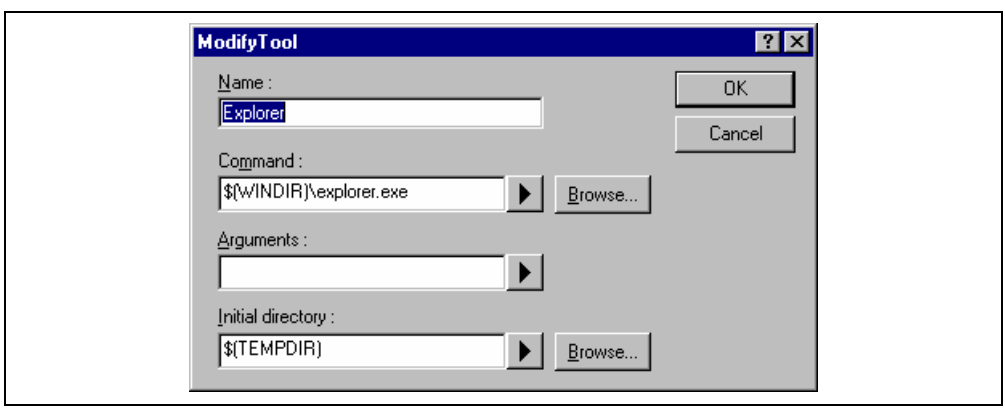


Figure 6.6: Modify Tool Dialog

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

6.3 Configuring the Help System

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

➲ To add a new help file:

1. Select **[Tools->Customize…]**. The dialog shown in figure 6.1 will be displayed. Select the "Help" tab (see figure 6.7).

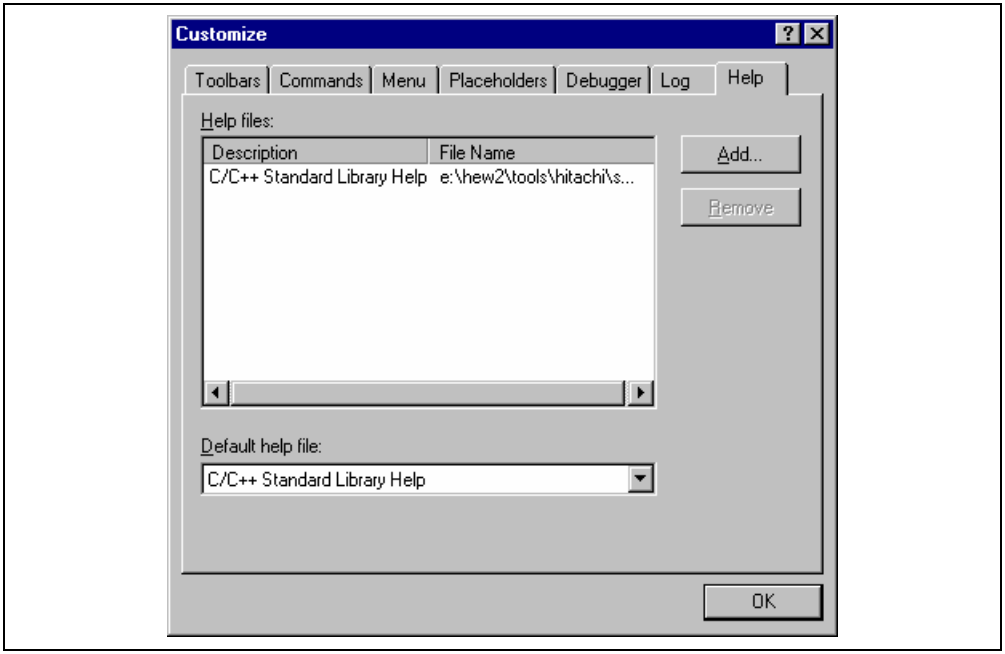


Figure 6.7: Customize Dialog Help Tab

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

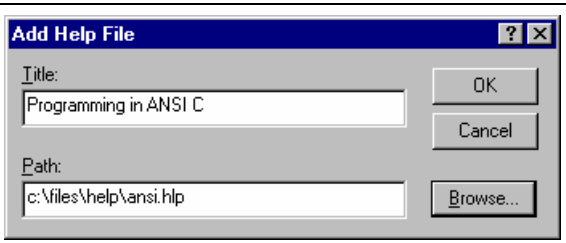


Figure 6.8: Add Help File Dialog

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

6.4 Specifying Workspace Options

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

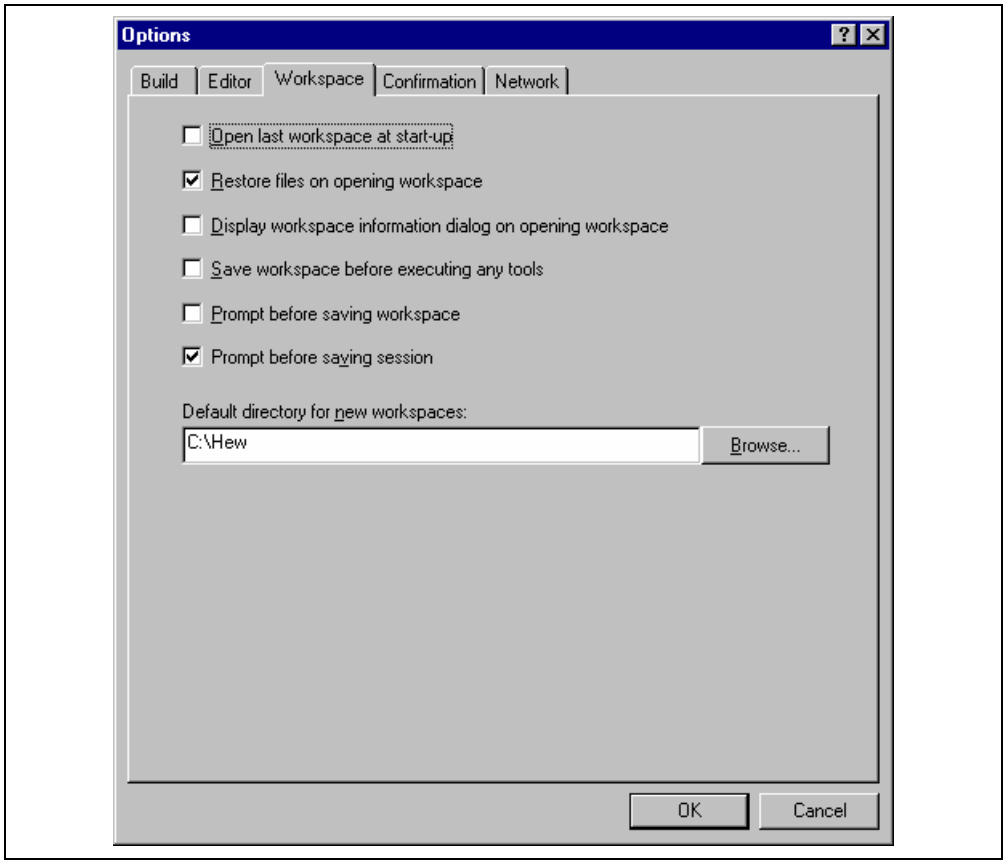


Figure 6.9: Options Dialog Workspace Tab

The following sections explain the options available on this tab.

6.4.1 Open last workspace at start-up

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

6.4.2 Restore the files on opening workspace

When you close a workspace, the HEW stores, which files were open. When you open a workspace, the HEW can restore (i.e. open) the same files so that you can continue your session in exactly the same state as when you

left it. If you would like the files associated with a workspace to be opened when you open a workspace then set this check box.

6.4.3 Display workspace information dialog on opening workspace

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

➲ To enter a workspace description:

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

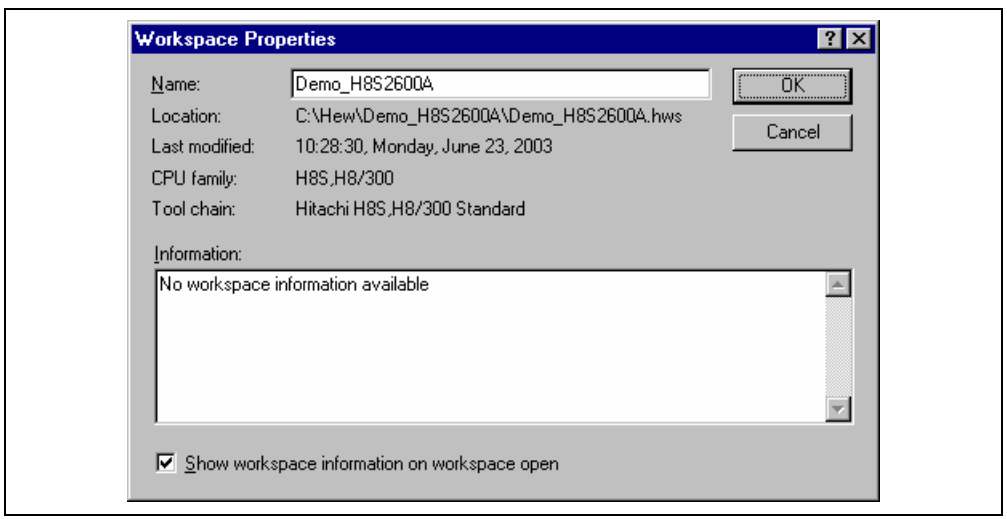


Figure 6.10: Workspace Properties Dialog

When a workspace is opened, the High-performance Embedded Workshop can display this information so that it is possible to determine whether the workspace is the desired workspace. To display this information on opening a workspace, set the "Display workspace information dialog on opening workspace" check box.

6.4.4 Save workspace before executing any tools

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

6.4.5 Prompt before saving workspace

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

6.4.6 Default directory for new workspaces

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

6.4.7 Prompt before saving session

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

6.5 Using an External Editor

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

- Double clicking on a file in the "Projects" tab of the "Workspace" window.
- Double clicking on an entry in the "Navigation" tab of the "Workspace" window.
- Double clicking on an error/warning in the "Build" tab of the "Output" window.
- Double clicking on an entry in the "Find in Files" tab of the "Output" window.
- Selecting the **[Open** <file>**]** option from the "Workspace" windows pop-up menu.
- Clicking the "Launch Editor" toolbar button.

➲ To specify an external editor:

1. Select **[Tools->Options…]**. The "Options" dialog will be displayed. Select the "Editor" tab (figure 6.11).

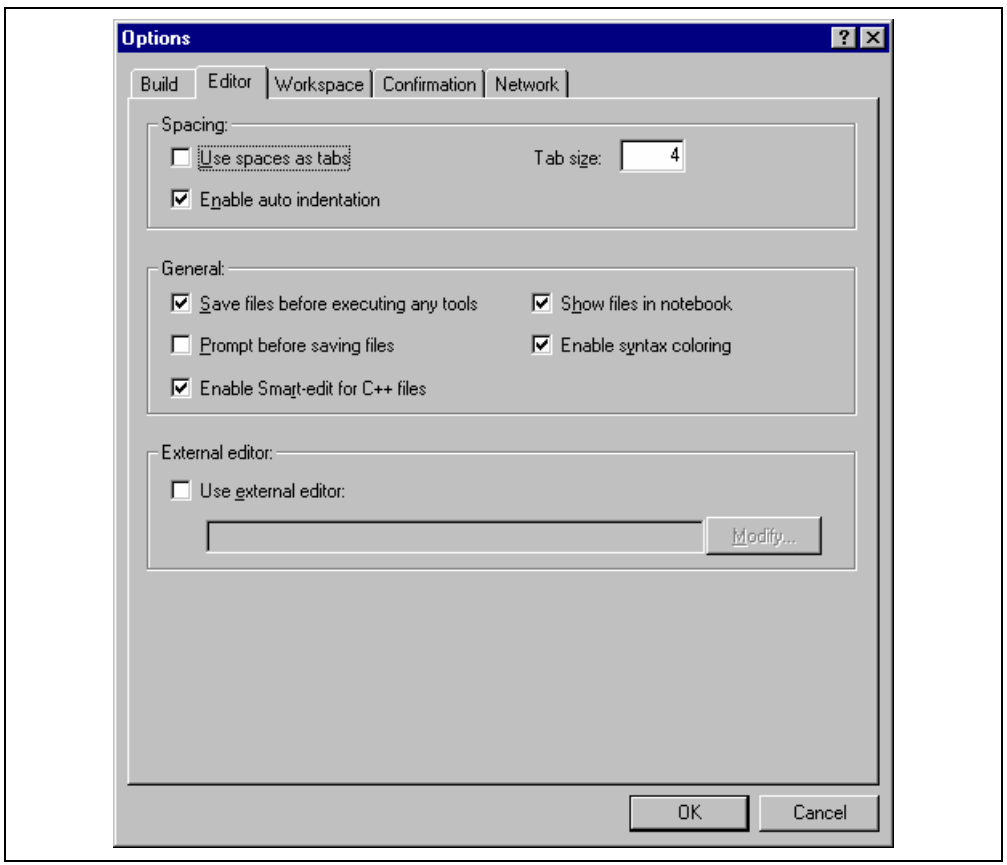


Figure 6.11: Options Dialog Editor Tab

2. Check the "Use external editor" check box.

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

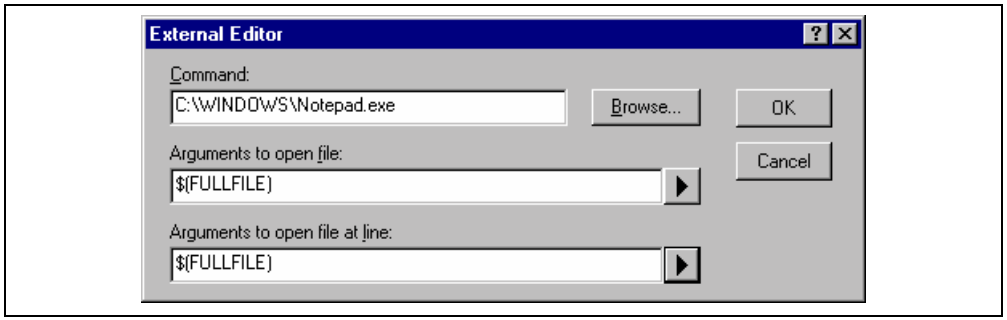


Figure 6.12: External Editor Dialog

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

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

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

6.6 Customizing File Save

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

The following sections explain the options related to file save.

6.6.1 Save files before executing any tools

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

6.6.2 Prompt before saving files

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

6.7 Using an External Debugger

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

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

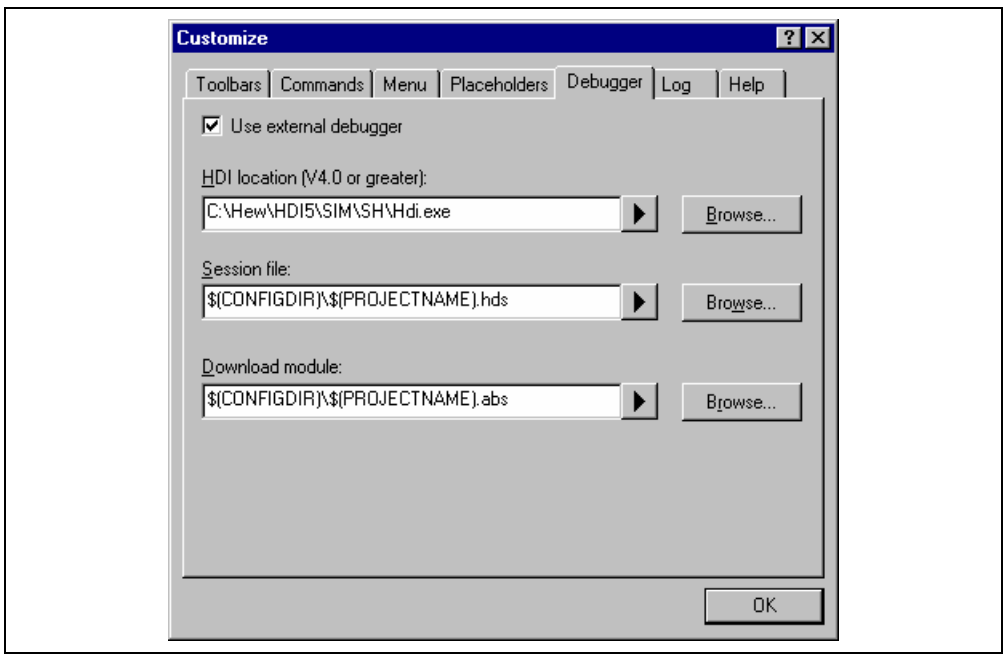


Figure 6.13: Customize Dialog Debugger Tab

To use an external debugger, check the "Use external debugger" checkbox and specify the items described below. There are three items of information, which need to be specified. Firstly, the location of the HDI executable must be specified. This must be version 4.0 or greater otherwise the behavior is not guaranteed. The second item of data is the session file. This tells HDI which session to load when it is launched. Finally, the location of the download module is required. This allows the HEW to automatically switch to HDI when the download module changes after a build. Click the "Launch External Debugger" toolbar button to invoke HDI with the specified session file.

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

6.8 Using Custom Placeholders

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

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

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

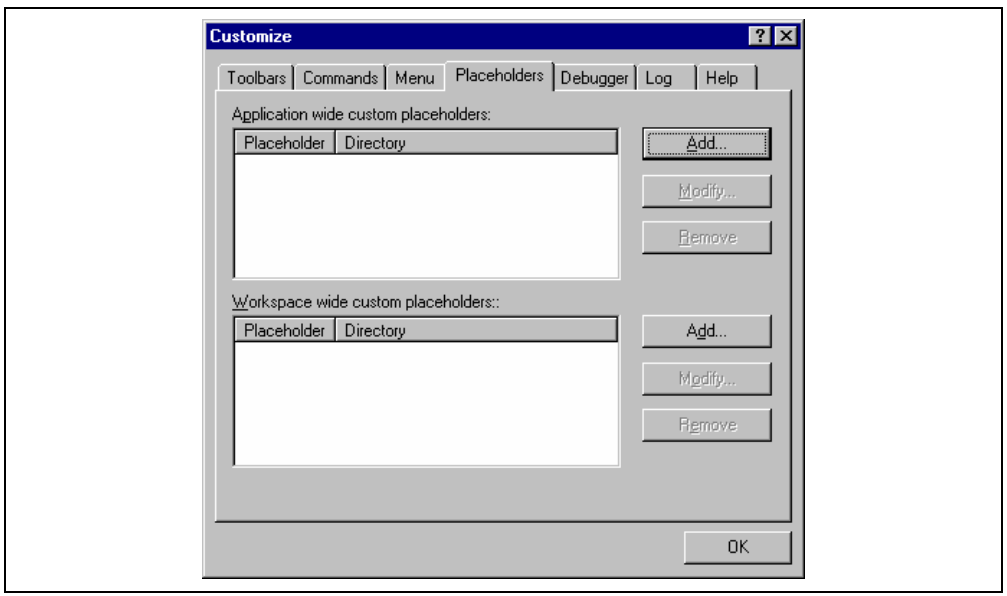


Figure 6.14: Customize Dialog Placeholder Tab

➲ To add a custom placeholder:

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

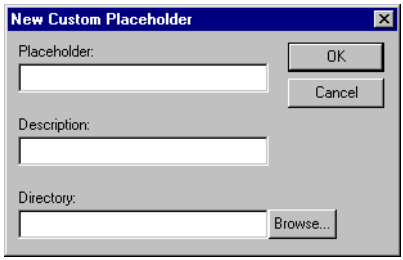


Figure 6.15: New Custom Placeholder Dialog

6.9 Using the workspace and project log facilities

The HEW 3.0 has workspace and project logging facilities integrated into the application. These facilities can be switched on via the log tab on the Tools customize dialog. This option is especially useful when the network database is in operation. This is because user names and changes are logged to this file. This dialog is shown in figure 6.16.

When the workspace log is clicked any workspace changes will be logged to a file with the same name as the workspace with a ".log" extension. This file will be located in the same directory as the workspace file.

When the generate log for projects log is clicked any projects in the current workspace that have changes made to them will be logged to a file with the same name as the project with a ".log" extension. This file will be located in the same directory as the project file.

The log file is updated when the workspace is saved.

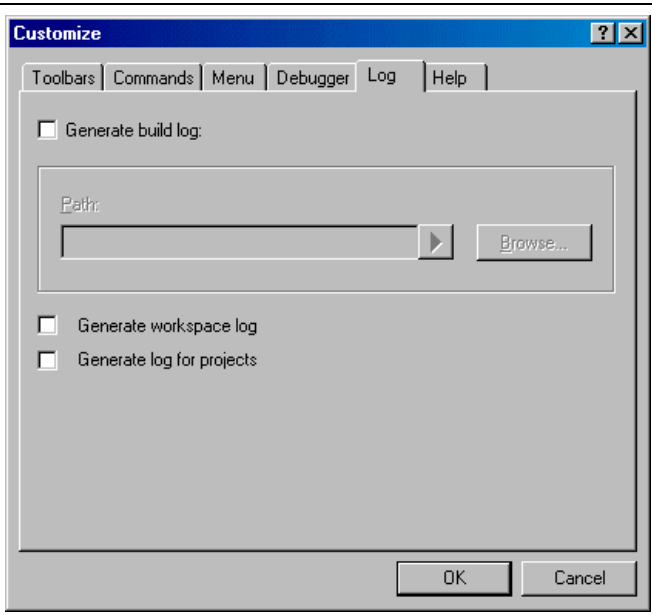


Figure 6.16: Tools customize log tab

7. Version Control

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

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

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

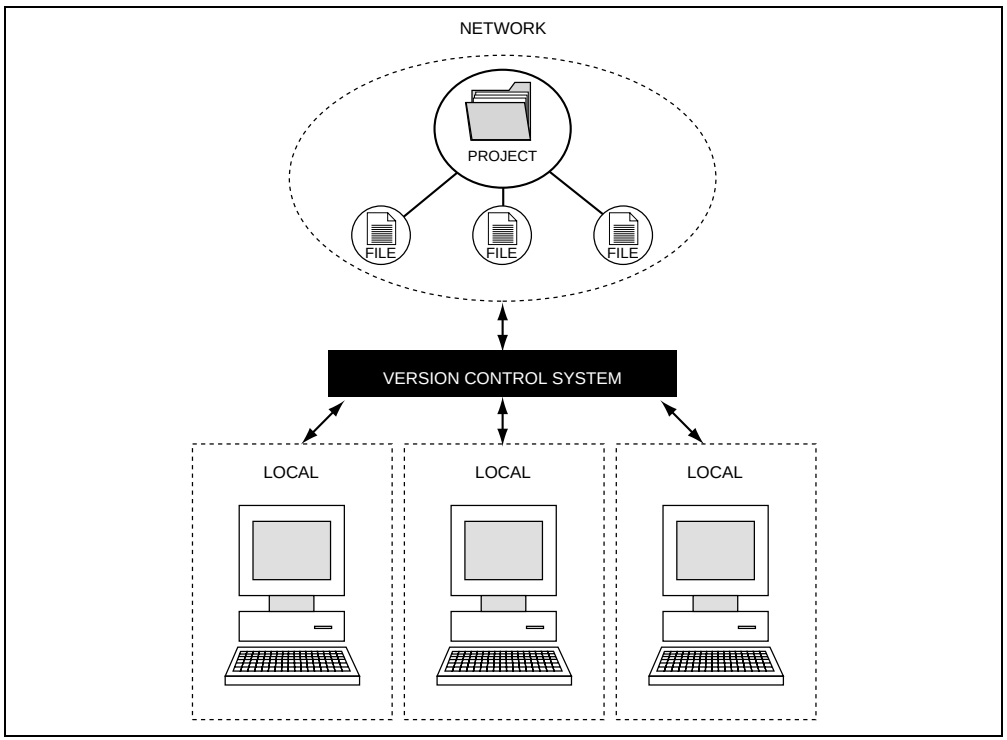


Figure 7.1: Version Control

7.1 Selecting a Version Control System

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

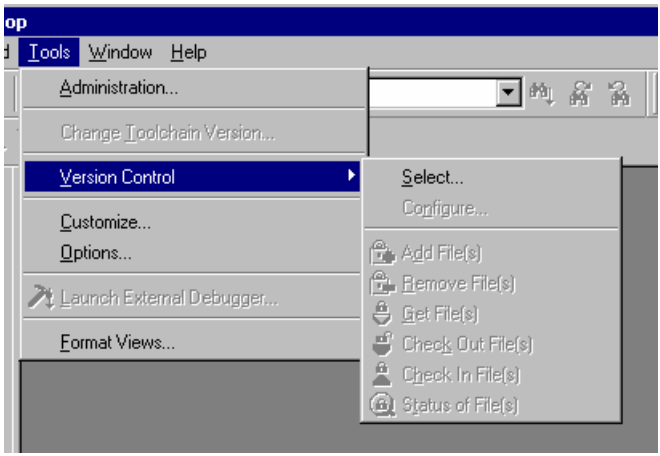


Figure 7.2: Version Control Sub-Menu

➲ To select a version control system:

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

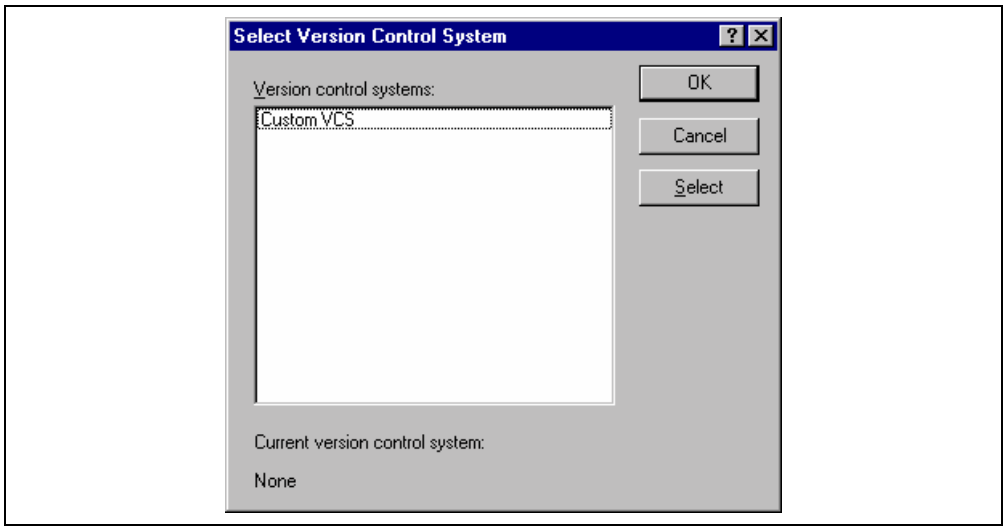


Figure 7.3: Select Version Control System Dialog

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

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

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

8. Using the Custom Version Control System

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

8.1 Defining Version Control Menu Options

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

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

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

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

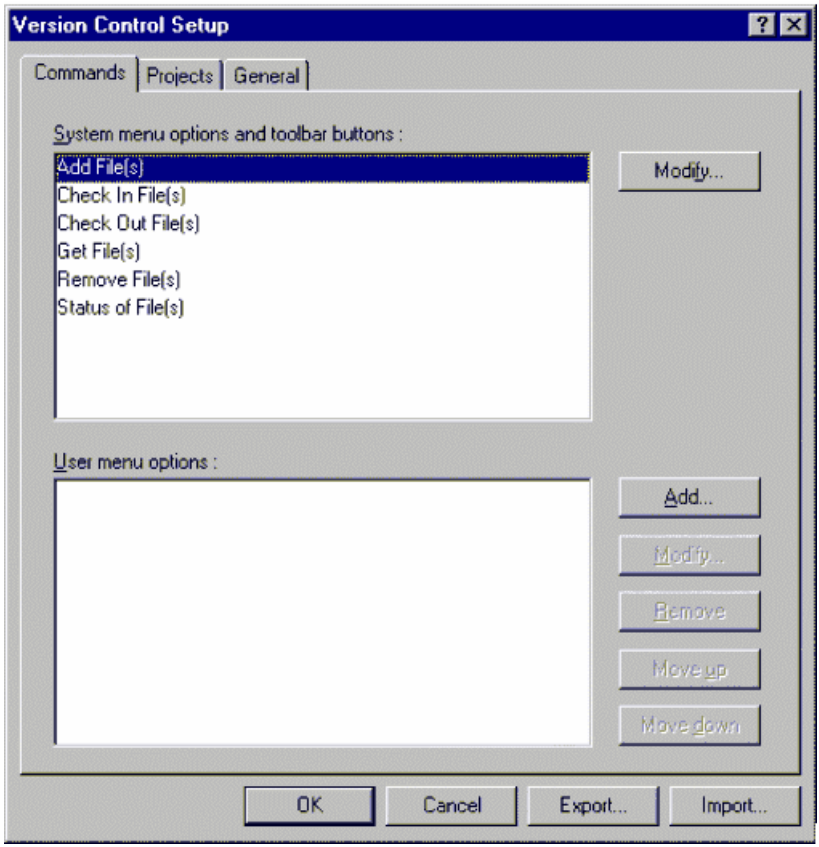


Figure 8.1: Version Control Setup Dialog Commands Tab

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

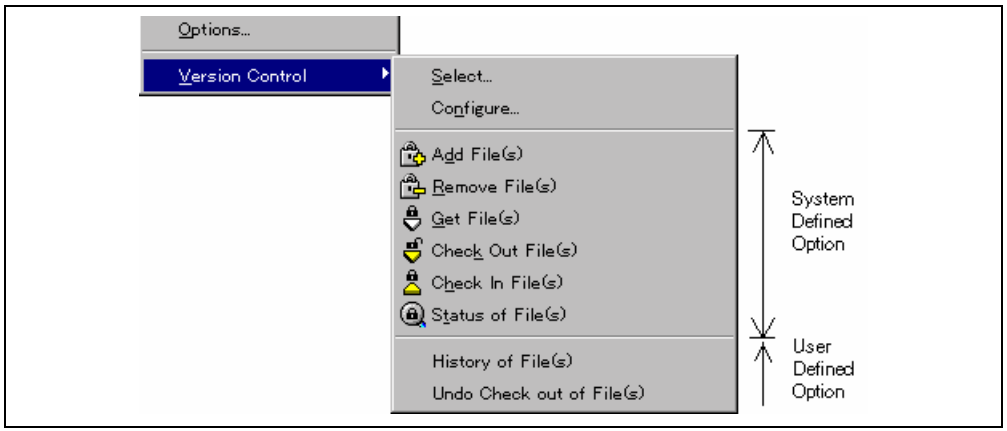


Figure 8.2: Version Control Sub-Menu

8.1.1 System menu options and toolbar buttons

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

Table 8.1: System Menu Option

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

➲ To modify a system menu / toolbar option:

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

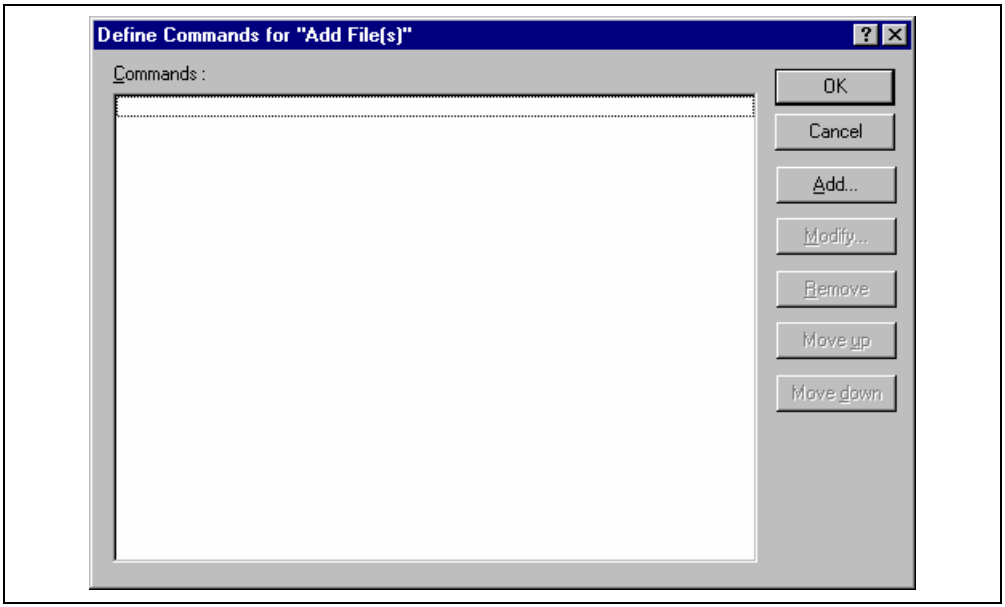


Figure 8.3: Modify System Menu Option (Example)

8.1.2 User menu options

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

➲ To create a new version control menu option:

1. Select **[Version Control->Configure…]**. The dialog shown in figure 8.1 will be displayed.
2. Click the "Add.." button. The dialog shown in figure 8.4 will be displayed.
3. Enter the name of the menu option into the "Option" field.
4. Commands are added to the menu option via the "Add…" button. See the section, "*Defining Version Control Commands*", later in this chapter for further information.
5. Close the "Add Menu Option" dialog by clicking "OK".
6. Close the "Version Control Setup" dialog by clicking "OK".

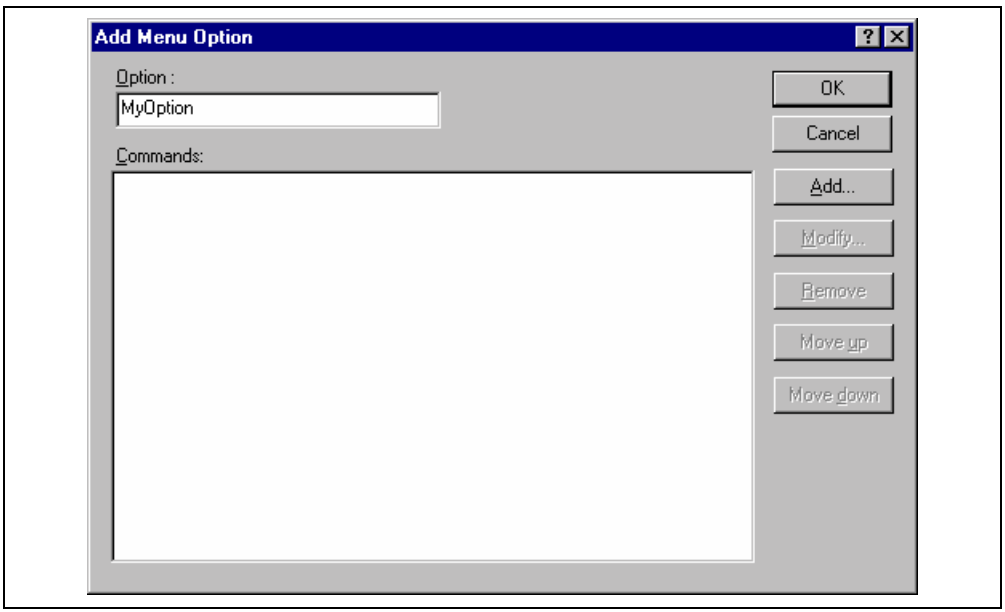


Figure 8.4: Add Menu Option Dialog

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

8.2 Defining Version Control Commands

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

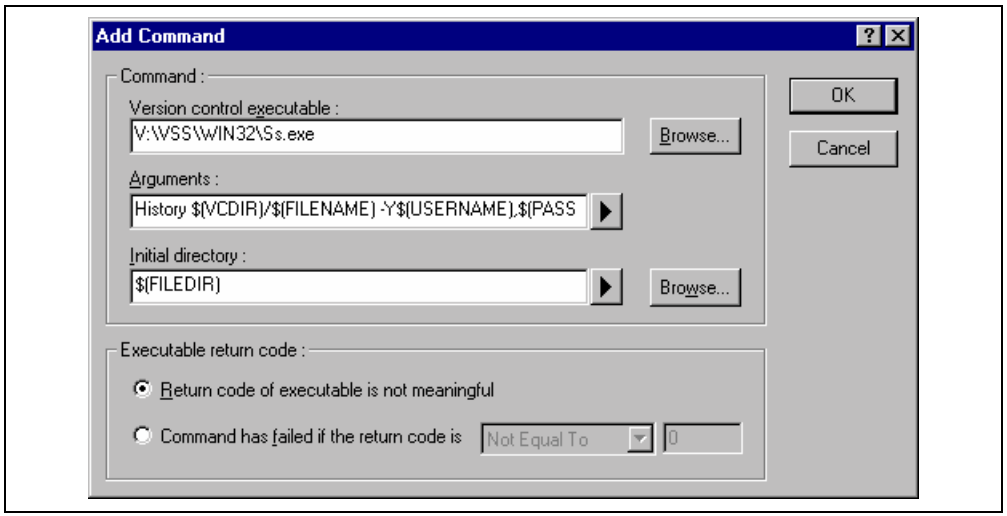


Figure 8.5: Add/Modify Command Dialog

➲ To define a command:

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

8.2.1 Executable return code

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

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

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

8.3 Specifying Arguments

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

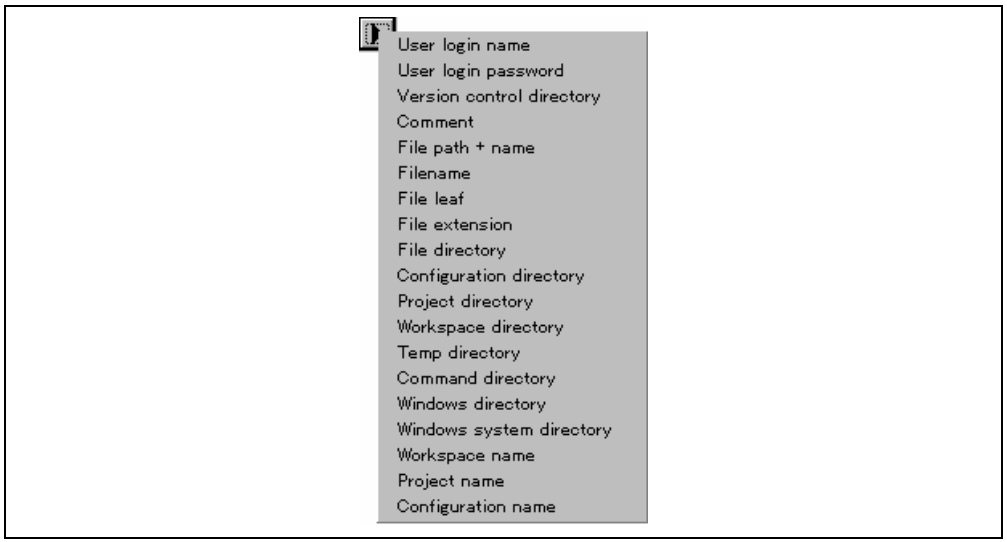


Figure 8.6: Arguments Field Placeholder Pop-up Menu

Table 8.2: Arguments Field Placeholders

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

8.3.1 Specifying File Locations

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

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

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

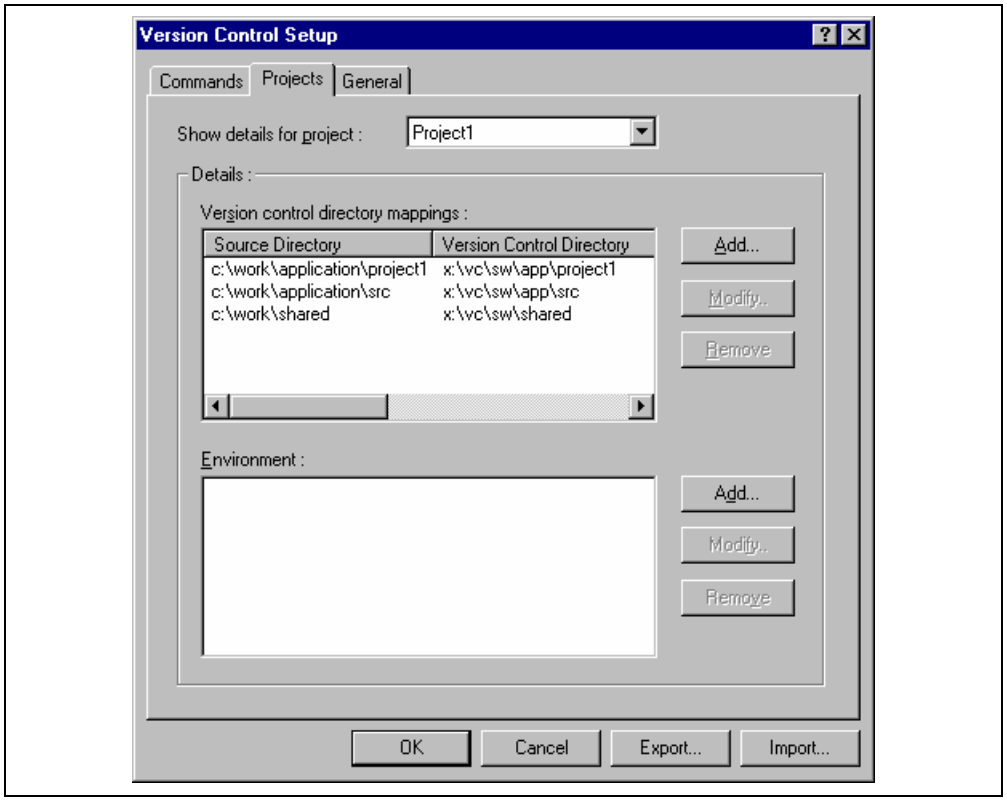


Figure 8.7: Version Control Setup Dialog Projects Tab

➲ To define a new mapping:

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

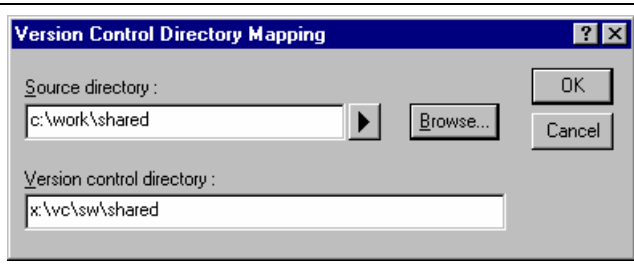


Figure 8.8: Version Control Directory Mapping Dialog

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

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

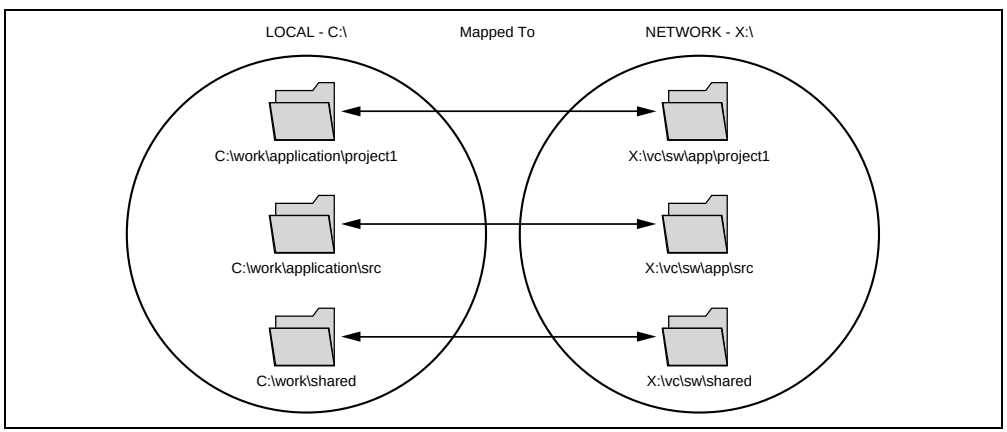


Figure 8.9: Example Mappings

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

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

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

For example, suppose FILE1.C is located at:

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

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

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

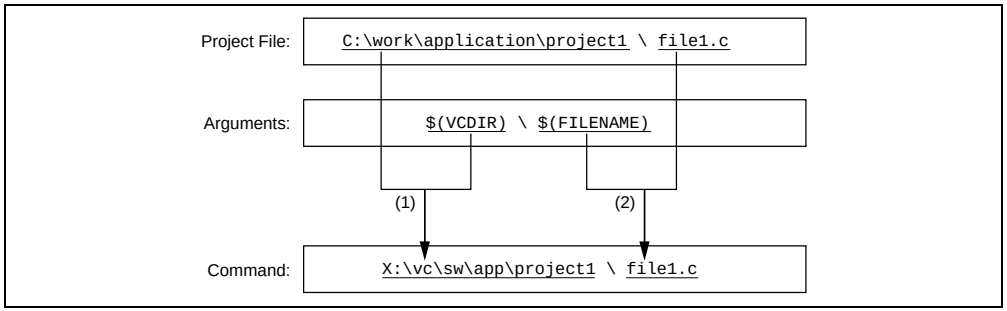


Figure 8.10: Example of Substitution

8.3.2 Specifying Environment

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

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

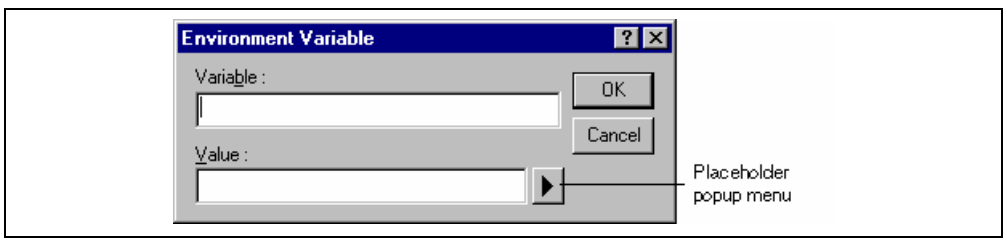


Figure 8.11: Environment Variable Dialog

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

8.3.3 Specifying Comments

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

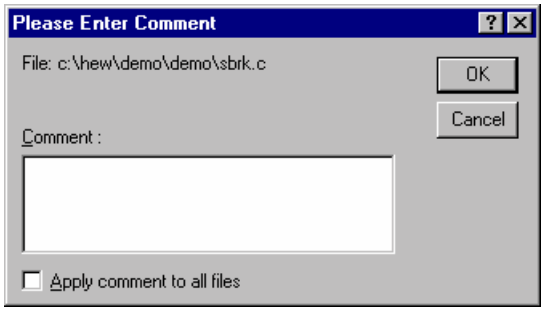


Figure 8.12: Please Enter Comment Dialog

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

8.3.4 Specifying a User Name and Password

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

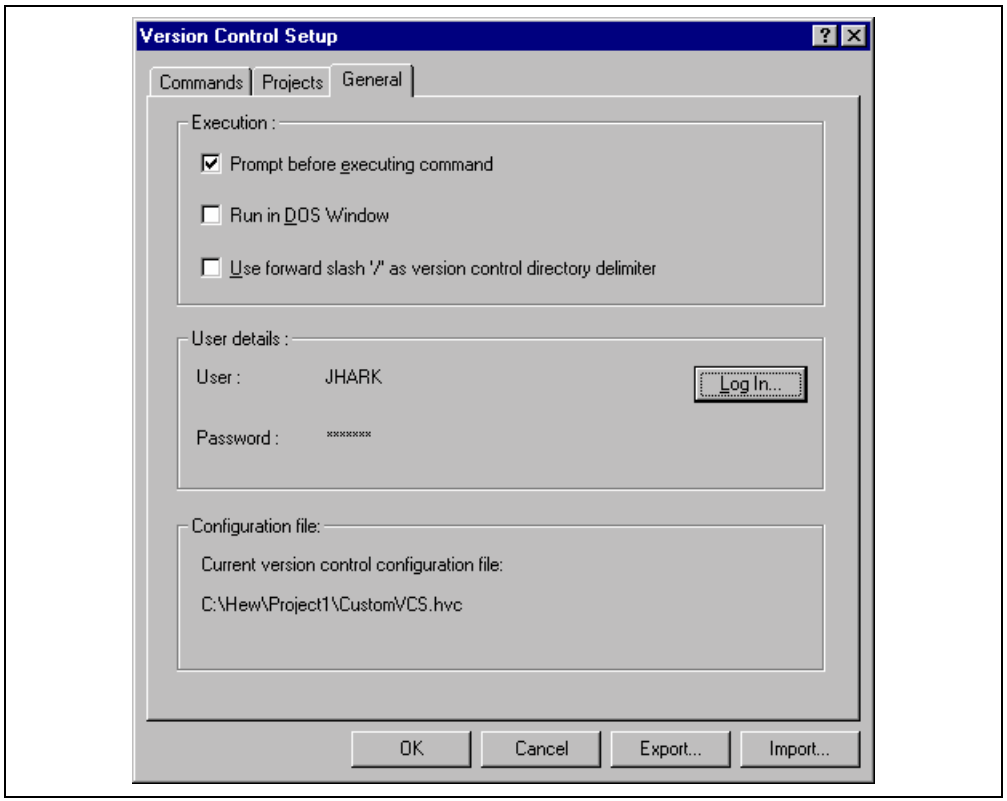


Figure 8.13: Version Control Setup Dialog General Tab

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

➲ To login (i.e. specify a user name and password):

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

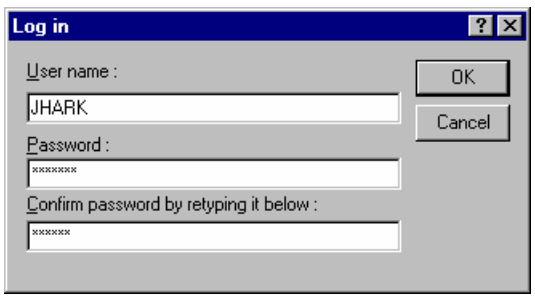


Figure 8.14: Log in Dialog

8.4 Controlling Execution

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

8.4.1 Prompt before executing command

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

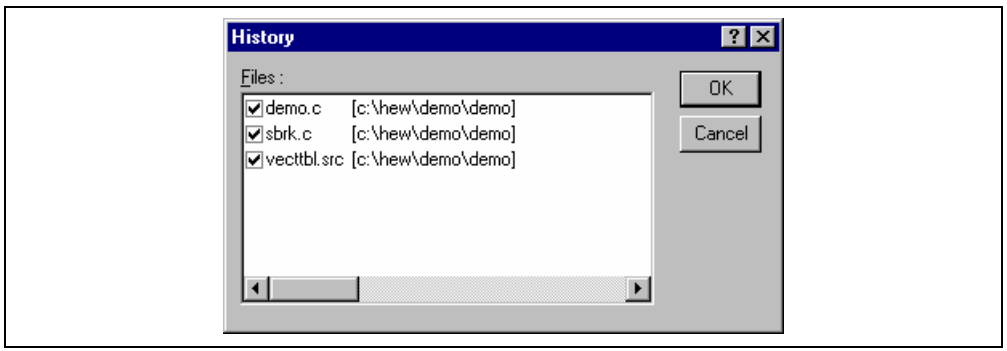


Figure 8.15: Command Prompt Dialog (Example)

8.4.2 Run in DOS Window

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

8.4.3 Use forward slash '/' as version control directory delimiter

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

8.5 Importing and exporting a Set-up

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

➲ To export a version control set-up:

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

➲ To import a version control set-up:

1. Select **[Version Control->Configure…]**. The dialog shown in figure 8.1 will be displayed.
2. Click the "Import…" button. A standard file open dialog will be displayed. Browse to the *.HVC file which you would like to import.
3. Select the file and then click "OK".

9. Using Visual SourceSafe

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

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

9.1 Attaching Visual SourceSafe to a Workspace

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

9.1.1 Selecting Visual SourceSafe

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

➲ To use Visual SourceSafe 5.0 or 6.0:

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

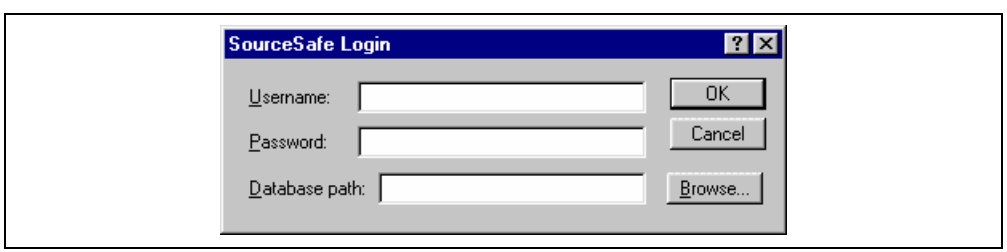


Figure 9.1: SourceSafe Login Dialog

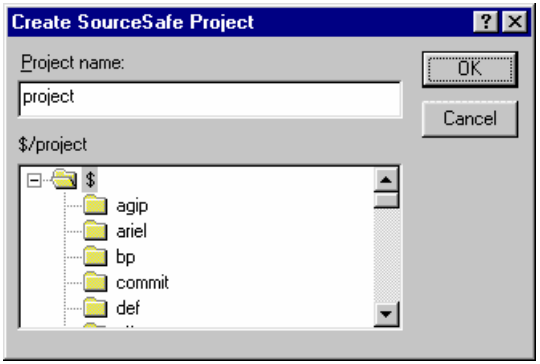


Figure 9.2: Create SourceSafe Project

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

9.1.2 Adding files to Visual SourceSafe

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

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

➲ To add a file or files to Visual SourceSafe:

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

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

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

9.2 Visual SourceSafe commands

The following 8 operations are available:

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

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

9.2.1 Removing a File from Version Control

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

➲ To remove a file or files from Visual SourceSafe:

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

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

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

➲ To get a read only copy of a file or files from Visual SourceSafe:

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

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

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

➲ To check out a writable copy of a file or files from Visual SourceSafe:

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

3. When the operation is finished the file has a red tick next to its name. This means you as the current user of HEW has checked it out.

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

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

➲ To check in edits made to a file or files in Visual SourceSafe:

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

9.2.5 Undoing a Check Out Operation

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

➲ To undo a check out of a file or files from Visual SourceSafe:

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

9.2.6 Viewing the Status of a File

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

➲ To view the status of a file or files in Visual SourceSafe:

1. Select the file(s) whose status you would like to view. You may also select a file folder, project folder, a workspace folder or combination thereof.
2. Click the Status of Files toolbar button () or select the **[Tools->Version Control->Status of Files]** menu option.
3. If a file has a blue tick next to it a user different to you has checked it out.

9.2.7 Viewing the History of a File

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

➲ To view the history of a file or files in Visual SourceSafe:

1. Select the file(s) whose history you would like to view. You may also select a file folder, project folder, a workspace folder or combination thereof.

2. Select the **[Tools->Version Control->Show History]** menu option.

9.3 Visual SourceSafe Integration Options

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

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

10. Network Facilities

10.1 Overview

The High-performance Embedded Workshop version 3 is capable of sharing workspaces and projects across a network. This allows users to concurrently work on shared projects and see each other's changes as they happen. This system can be used in conjunction with version control. The major difference with using this system is that each user can modify and update the workspace and project without making all of the other users reload their project and potentially lose all their changes.

This system is implemented by making one of the machines attached to the network the server machine. All other client machines then use the service this machine is providing. So if one of the client machines adds a new file, the server machine is notified. The server then notifies all other clients the action has taken place. This structure is shown below in figure 10.1.

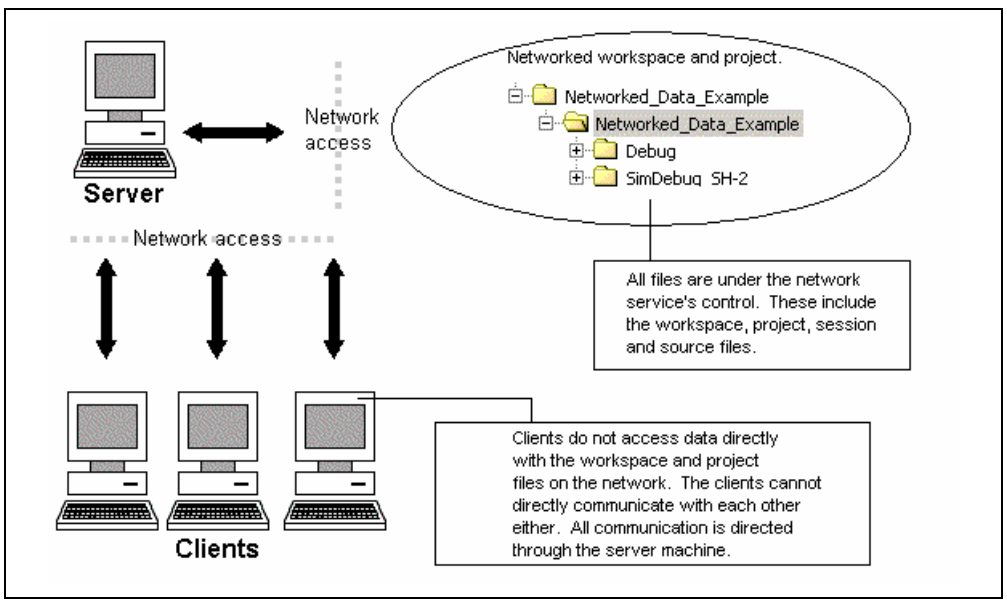


Figure 10. 1: Network database access structure

The network system allows users to be given access rights to files. This allows the project administrator to make sure the only people who can modify the project and source files are allowed to do so. This might allow the administrator to limit each user to only have write capabilities for their own area of the project, other areas would be read only. This could limit any potential conflict or damage one user could do to other areas of the project. These limitations can be set to a number of different levels. This is outlined later in this section.

Note: Certain operations are locked when other clients are carrying them out. This means that if one machine is currently changing the toolchain options all of the other machines can only see read only versions of this data.

Note: The performance of HEW does suffer when using the network facilities. If working in a small team it might be more suitable to use the single user mode and version control.

10.1.1 Enabling network access

➲ To use network access:

1. Select **[Tools->Options]**. The HEW tools options dialog is displayed.
2. Select the network tab. This is displayed in figure 10.2.
3. Click the enable network access checkbox. This should add an administrator to the system without a password. The administrator is the only user that can add additional users to the system and change user access rights. The administrator has the highest level of access.
4. Before leaving the network dialog the administrator must set their password. It is not possible to leave this dialog until this is completed. This is described below.

10.1.2 Setting the administrator user's password

➲ To set the password:

1. Continue from the previous sections steps.
2. Click the password button. This should have been enabled when the network data access was enabled.
3. The password dialog is displayed. This is shown in figure 10.3.
4. The user name is read only in the top field. In this case it should be Admin.
5. Type the new password into both of the fields and click OK.
6. This should set the user and password on the [Tools->Options] network tab.
7. It is now possible to leave the [Tools->Options] dialog.
8. When the dialog is closed you are asked if you want to save the workspace and then re-open it. This is because the workspace must be re-opened in the shared access mode. If the changes are not saved then they will be lost.
9. When the workspace is re-opened a dialog is displayed which asks you to log back into the system. Once you have logged in a dialog is displayed which shows your current access rights. For example if you are the admin user the level will be administrator. When this dialog is closed the HEW server window is opened and the network facilities are enabled.

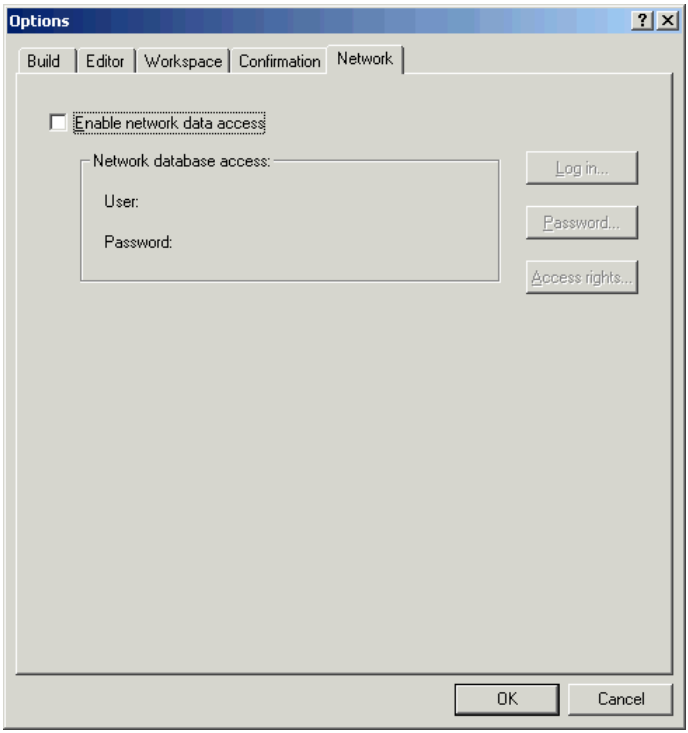


Figure 10. 2: Initial setup of network tab

Figure 10. 3: Password dialog

10.1.3 Adding new users to the system

The initial setting of the network database adds an administrator user and a guest user to the system. The following levels of access are possible in the HEW system:

- **Administrator:**

  Full access to every aspect of HEW. The user can add and remove users from the projects and change access rights. The administration user can change the workspace and project files and also the source files.

- **Full read/write access**:

  The workspace and project files can be modified, as can the source files. But it is not possible to change user access rights from this access level.

- **Read/write file access:**

  Only the source files can be modified. All project settings can only be viewed not modified.

- **Read only:**

  All source files and project files can only be viewed as read only. Nothing can be modified.

When any user opens a network-enabled project they must type in their user name and password. Until this is done no access can be granted. Once entered the user is given one of the levels of access as seen above.

➲ To add a new user to the system:

1. Log in with a user who has administrator access rights. The process for doing this is described above.
2. Select **[Tools->Options]**. The HEW tools options dialog is displayed.
3. Select the network tab. This is displayed in Figure 10.2.
4. Click the access rights button. The dialog displayed in Figure 10.4 is displayed.
5. Click the add button. The log in dialog is displayed. This allows you as the administration user to add a new log in name and password. Normally the password should be set to some default text or left blank. Then click OK.
6. Once OK is clicked the user is added with read only rights. To change the access level select the user you wish to modify and then click the required radio button. Then click OK to save the access rights changes.

➲ To remove an existing user to the system:

1. Log in with a user who has administrator access rights. The process for doing this is described above.
2. Select **[Tools->Options]**. The HEW tools options dialog is displayed.
3. Select the network tab. This is displayed in Figure 10.2.
4. Click the access rights button. The dialog displayed in Figure 10.4 is displayed.
5. Select the user you wish to remove in the users list.
6. Press the remove button.
7. Then click OK to save the access rights changes.

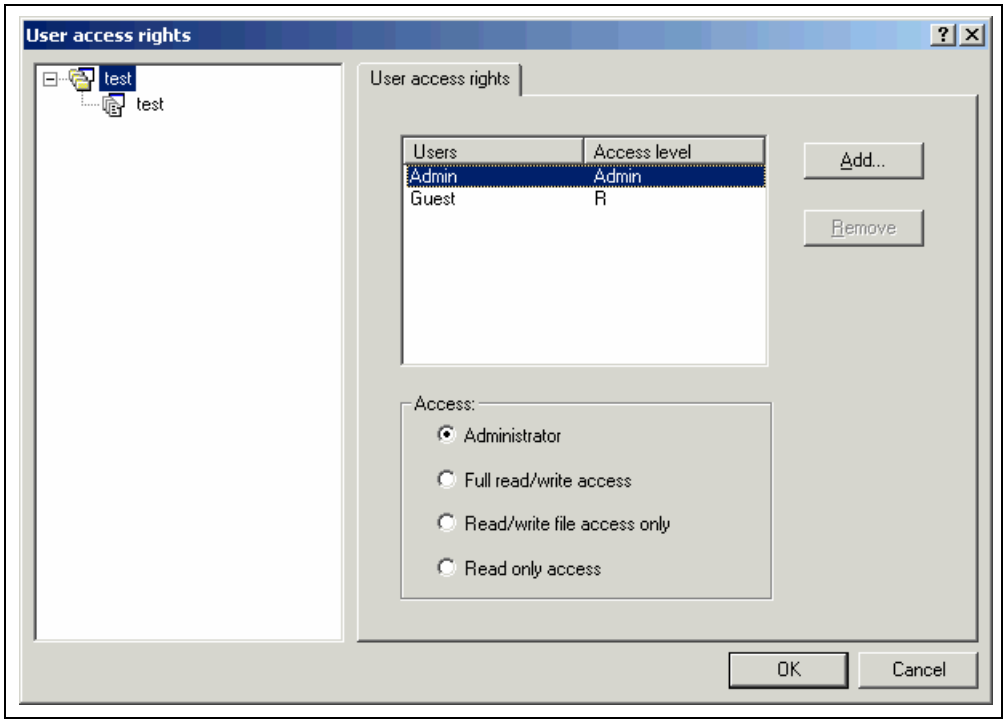


Figure 10. 4: Access rights dialog

10.1.4 Changing your password

To change your password:

1. Log into the HEW network database you are changing your password for. Select **[Tools->Options]**. The HEW tools options dialog is displayed.
2. Select the network tab. This is displayed in Figure 10.2.
3. Click the password button.
4. Enter your new password and confirm it in the second edit box.
5. Click OK.
6. Then click OK to save the password change.

10.1.5 Using the network HEW service

When you connect to a networked project for the first time the HEW automatically connects you to the correct network HEW service. This is defined using machine name. If the service cannot be found using the machine name in the workspace then the dialog displayed in figure 10.5 is shown. Simply type or browse to the machine where the service is located and click OK. If you want to be the server machine then leave the radio button on its default selection, use local machine.

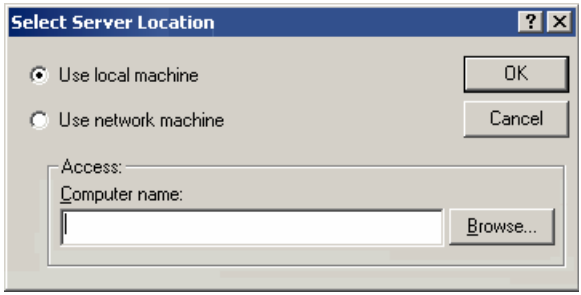


Figure 10. 5: Select machine name

If you have previously been the server of a workspace then the following message will be displayed when you attempt connection to another machine. This dialog is displayed in figure 10.6. Clicking OK then connects your machine to the new location.

Figure 10. 6: Select machine name

Note: If the network is running multiple HEW workspaces with the network service enabled then a user can only access one of them at one time. The only instance when this is not the case is if the same machine is serving all of the network workspaces.

11. Difference View

The High-performance Embedded Workshop now has an integrated difference view. This view allows detailed difference comparisons to be made with local files on the drive and also with files in the version control system. In the HEW version 3 onwards the Visual SourceSafe component has this facility.

The difference window can be invoked two ways. The first is from the show differences menu item on the tools menu. The second is via the workspace window pop-up with the same name. When using the pop-up menu the current file selection is automatically added to the edit box. Clicking the show differences menu item displays the dialog shown below in figure 11.1.

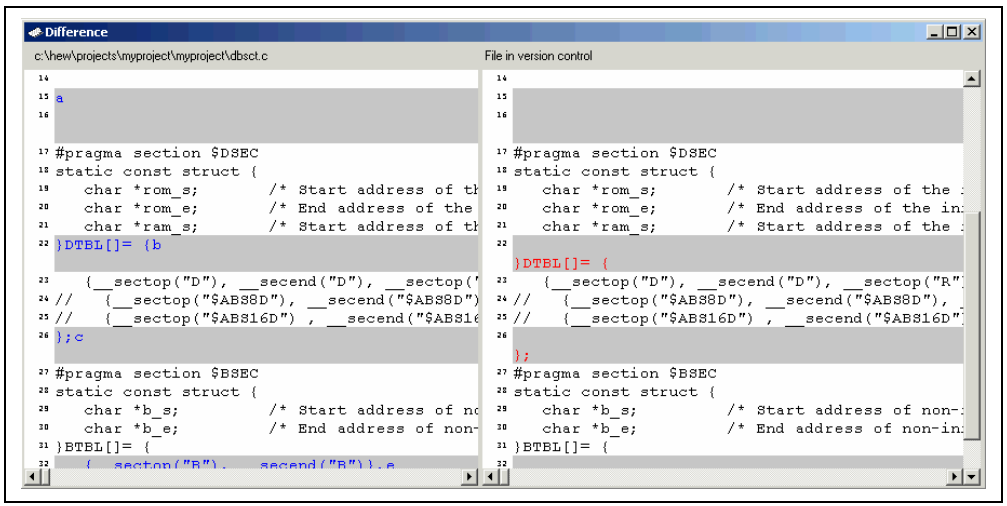


Figure 11. 1: Difference view

➲ To perform a difference comparison with two files on your local drive:

1. Select **[Tools->Show Differences…]**. The difference compare files dialog is displayed. This is shown in figure 11.2.
2. Ensure the compare with file on drive radio button is enabled.
3. Enter the first and second file to compare. You can either select a previous difference comparison or browse to a new file.
4. Clicking the advanced button displays the dialog in 11.3. This allows you to perform the difference comparison without taking white space into account. Click OK when you are finished with this options dialog.
5. Click compare.
6. The difference view is displayed. The two files being compared are loaded into each side of the split view. Their names are at the top of each window.

➲ To perform a difference comparison with a local file and a file in SourceSafe:

1. Ensure the SourceSafe component is enabled. Also note that the file must be have been added into the version control system.
2. Select **[Tools->Show Differences…]**. The difference compare files dialog is displayed. This is shown in figure 11.2.
3. Ensure the compare with version control radio button is enabled.

4. Enter the first file to compare. You can either select a previous difference comparison or browse to a new file.
5. Clicking the advanced button displays the dialog in 11.3. This allows you to perform the difference comparison without taking white space into account. Click OK when you are finished with this options dialog.
6. Click compare.
7. The difference view is displayed as shown in figure 11.1.

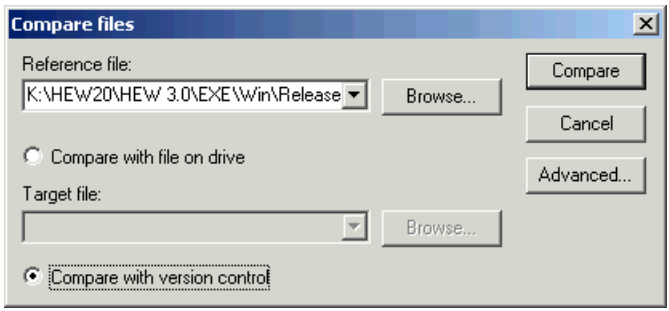


Figure 11. 2: Difference compare files dialog

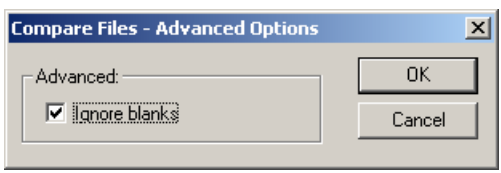


Figure 11. 3: Difference advanced options dialog

The difference view functionality can be accessed in two different ways. The difference view has its own toolbar this is shown in figure 11.4. This functionality can also be accessed via the difference window right mouse button pop-up. This is shown in figure 11.5.

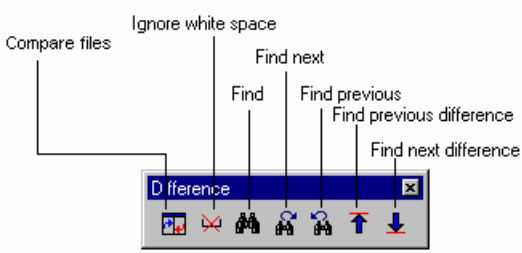


Figure 11. 4: Difference toolbar

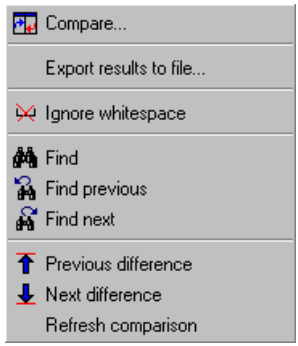


Figure 11. 5: Difference popup menu

The functionality for each menu item is described below:

- Compare: This opens a new compare window so that some new files can be comapred and the differences displayed.
- Export results to file: This opens a dialog which allows you to choose a file to export the current difference results to a textual format.
- Ignore white space: The ignore white space option which is on the advanced options dialog can be toggled via this menu item.
- Find: Displays a standard find dialog. This uses the same find dialog as the HEW editor.
- Find next: Finds the next string that meets the find requirements.
- Find previous: Finds the next previous string that meets the find requirements.
- Previous Difference: Automatically jumps the view to the next previous difference.
- Next Difference: Automatically jumps the view to the next difference.
- Refresh comparison: Refreshs the view to manually run the difference comparison again. This can be used if either file has been modified since the last comparison.

The difference window also allows the colours in the view to be customised. This procedure is outlined below:

➲ To change the colors viewed in the difference window:

1. Select **[Tools->Format Views…]**. The format views dialog is displayed.
2. Expand the difference view. The dialog is shown and looks the same as figure 11.6.
3. Select the category you want to change. The color display tab is displayed and allows modification. The categories available are Left hand side moved lines, Left hand side different lines, Right hand side moved lines and Right hand side different lines,
4. Click OK to keep the changes.

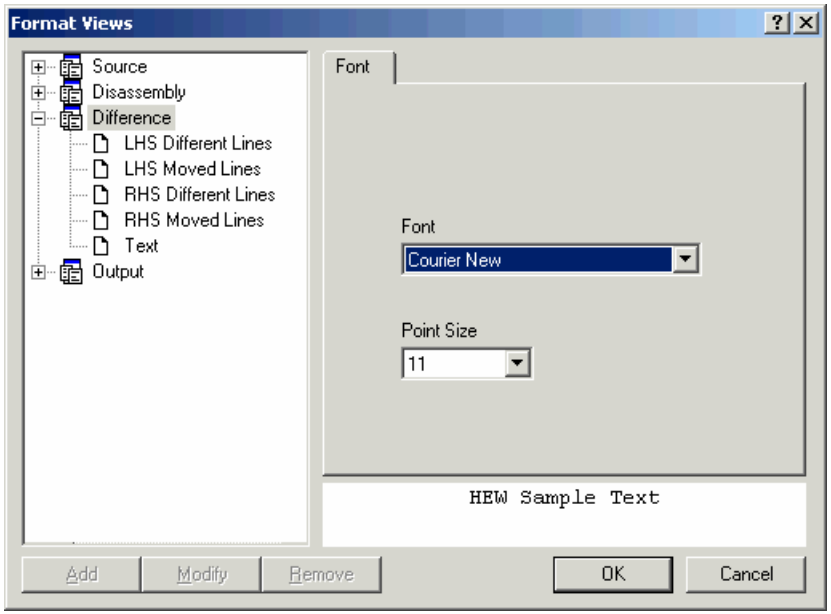


Figure 11. 6: Format views dialog

12. Technical Support

The High-performance Embedded Workshop has a number of integrated technical support features.

➲ To check for HEW product updates or service packs:

1. Select **[Help->Technical support->Check website for updates].**
2. Your default web browser is invoked and defaults to the HEW download page for your region.
3. Browse this area for HEW updates to fix bugs or add new features.

Occasionally you may experience some unforeseen problems with the HEW application. If a problem does occur that results in a application crash the HEW bug tracking program will be invoked automatically. This allows you to compile a bug report and this can then be sent to your technical support contact in a variety of ways. It is also possible to invoke this tracker program manually. This is described below:

➲ To create and send a HEW bug report:

1. Select **[Help->Technical support->Create Bug Report].**
2. Detailed information is generated from your HEW system. This may take some time. The bug report dialog is then displayed. This is shown in figure 12.1.
3. It is possible to then add additional information concerning the exact issue you have found in the large edit box.
4. Once you are happy with your report, you can choose the method of sending the report in the submit drop list. This has e-mail entries and an entry for printing the report.
5. Then click submit. This will send the report.

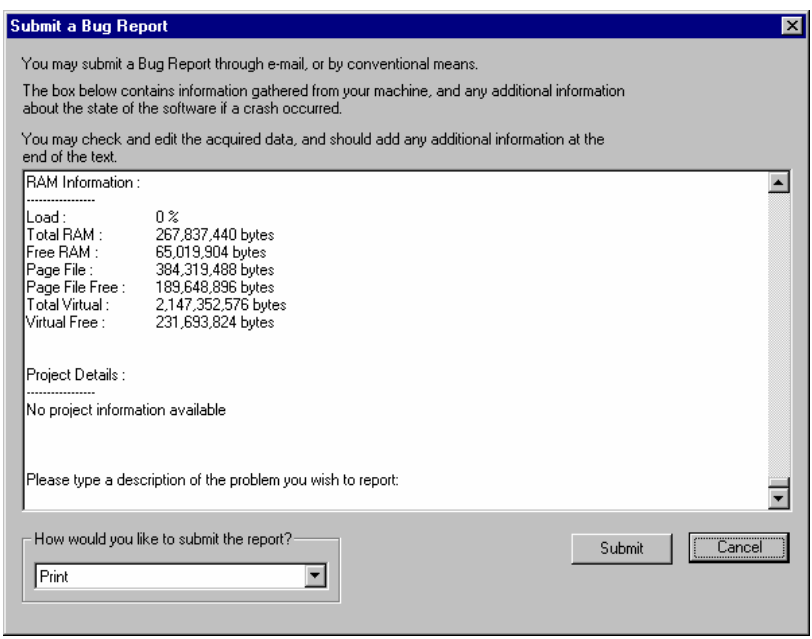


Figure 12. 1: Submit bug report dialog

13. Navigation facilities

The High-performance Embedded Workshop 3.0 has a number of new integrated navigation facilities. The navigation window is located alongside the project and templates window. This view is shown below in Figure 13.1.

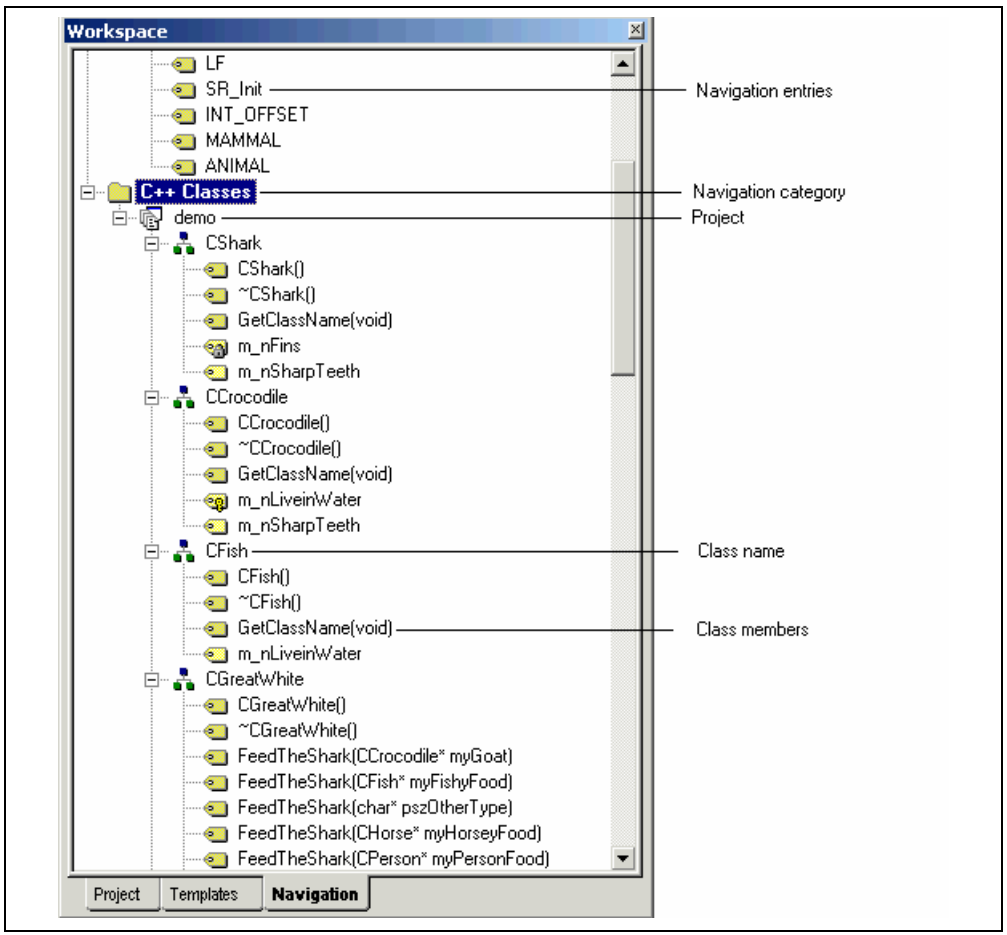


Figure 13.1: Navigation view

The navigation view contains categories for all supported navigation types. In HEW 3.0 the following navigation components are supported as standard:

- C++ Classes: All classes, functions and members are displayed for C++ source files.
- C Defines: All #defines for C and C++ source files are displayed.
- C Functions: All ANSI C standard functions are for C source files displayed.

Each category is displayed in the top level of the view. Underneath each category each project in the workspace is displayed. Then the items belonging to that particular project are displayed below the project icon.

It is possible to disable scanning for certain navigation categories if you do not require the information.

➲ To switch off a navigation category:

1. Right click on the navigation window. The pop-up menu displayed in figure 13.2 is shown.
2. Select the "Select Categories" menu item.
3. The dialog in figure 13.3 is displayed.
4. Uncheck any categories you are not interested in seeing definitions for.
5. Click OK.

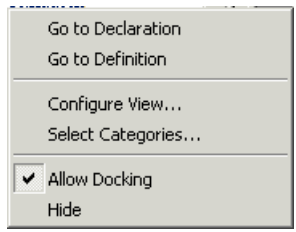


Figure 13. 2: C++ Navigation pop-up menu

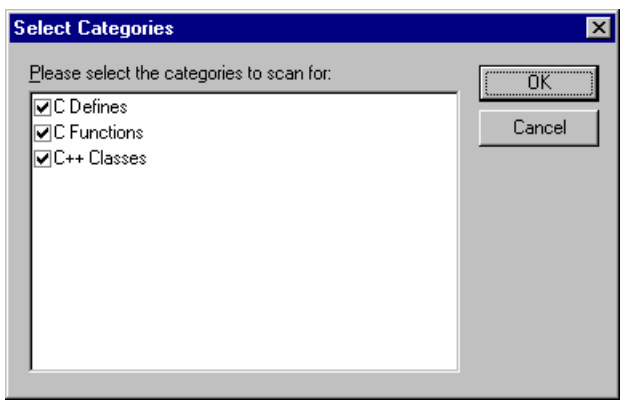


Figure 13. 3: C++ Navigation select categories dialog

Note: The navigation items are displayed gradually as the files are scanned. This means it may take some time if there are many files to fully complete the navigation view update.

Files are rescanned when they are saved. This means that navigation information will not be available for new classes and functions until the file or files are saved.

13.1 C++ Navigation component

The C++ navigation component is the most complicated of the three supported navigation components. It supports the following structures in the view for C++ source files. The basic structure of the information is shown below in figure 13.2.

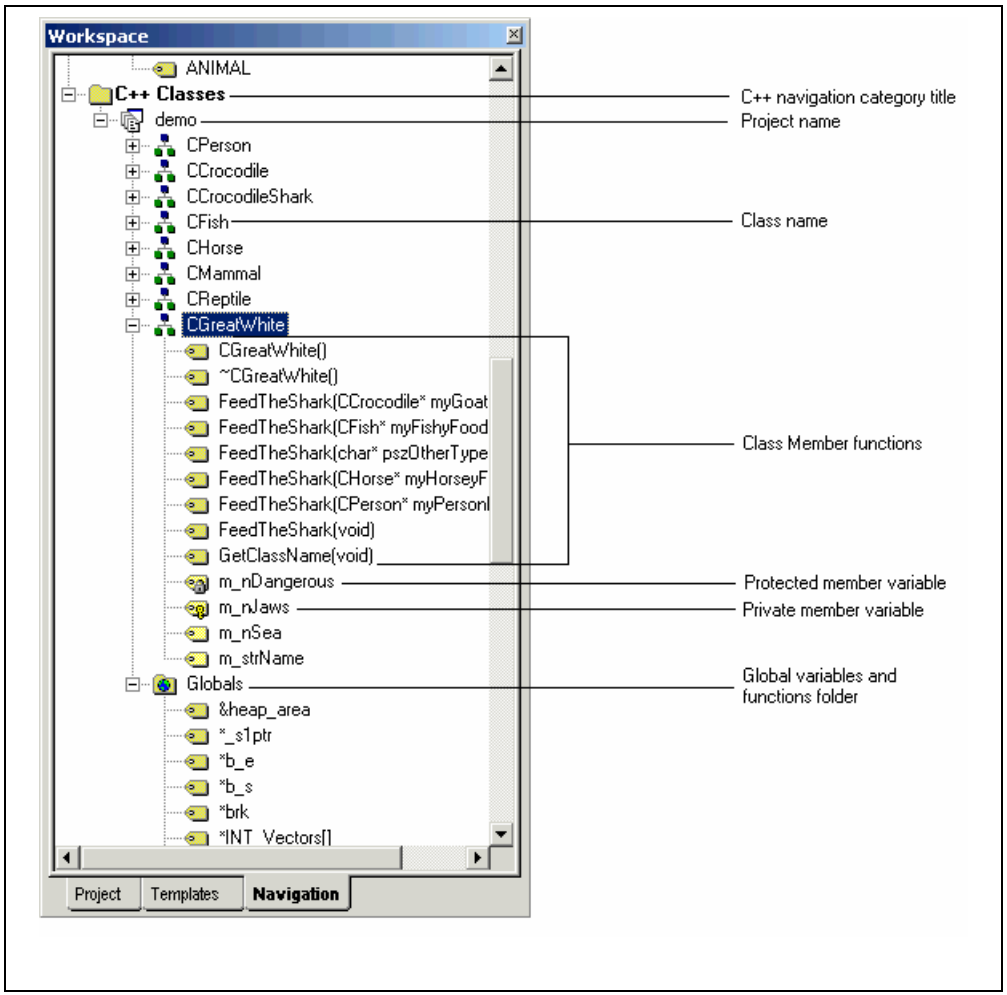


Figure 13. 4: C++ Navigation information

The C++ navigation view uses a number of icons to describe the type of function or variable the icon belongs too. These are listed in the table below:

Icon	Description
	Public member function
	Private member function
	Protected member function
	Private member variable
	Protected member variable
	Public member variable

Figure 13. 5 C++ Navigation Icons

The navigator view allows you to move around your source code quickly and efficiently. Double clicking on a navigation item by default jumps you to the associated navigation items definition. Normally the definition is found in the source file and the declaration is found in the header file.

This default behavior can be modified for this action. This can be achieved via the configure view dialog. The configure view dialog modifies the way data in the navigation view is displayed.

➲ To configure the C++ navigation views data:

1. Right click on a C++ navigation item in the navigation window. The pop-up menu displayed in figure 13.2 is shown.
2. Select the "Configure view" menu item.
3. The dialog in figure 13.5 is displayed.
4. Decide what items you wish to modify and click OK to accept the changes.

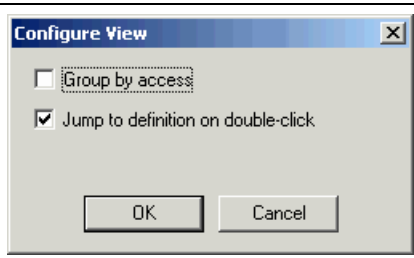


Figure 13. 6 C++ Configure view dialog

The following items are available on the configure view dialog.

- Group by access: This option groups the display of public, protected and private member variables and functions together in the navigation view.

- Jump to definition on double click: By default this option is checked. If this option is switched off then a dialog is displayed which asks you where you wish to jump. You rather than HEW resolve the ambiguity. This dialog is shown in figure 13.6.

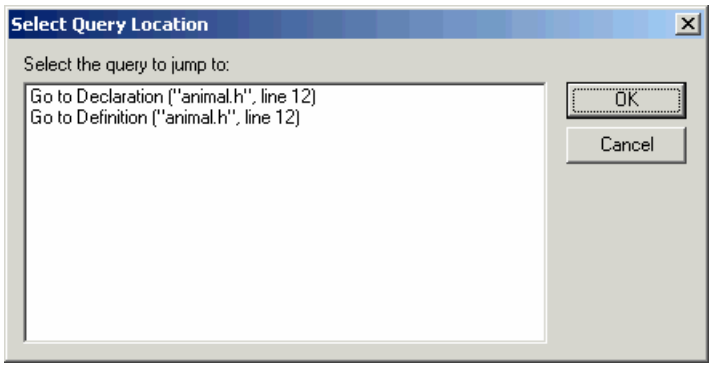


Figure 13. 7 C++ Configure view dialog

Another useful facility is the capability of viewing the base or derived classes for a certain selection.

➲ To view the Base or derived classes

1. Select the class that interests you in the navigation view.
2. Right click and view the pop-up menu.
3. To see the derived classes for the selection click the "Show Derived Classes" menu item. To see the base classes for the selection click the "Show Base Classes" menu item.
4. Depending on the selection a dialog is displayed which shows the class structure selected in an expanded tree format.
5. Click OK to close this dialog once you have the information you require.

13.2 C Function and #defines navigation components

These components simply add the function and #define definitions to the navigation view. It is then possible to jump to these definitions by double clicking on the label you wish to view.

14. Smart Editor

Another feature of the High-Performance Embedded Workshop is its smart edit facility. This is enabled by default for all C++ source files. This feature allows the HEW editor to access C++ navigation information and provide auto-completion help when using C++ classes and member functions.

➲ To view the Smart edit status:

1. Click on the **[Tools->Options]** menu item.
2. Select the "Editor" tab of the tools options dialog.
3. The dialog in figure 14.1 is displayed. The "Enable Smart-edit for C++ files" should be checked.
4. Click OK.

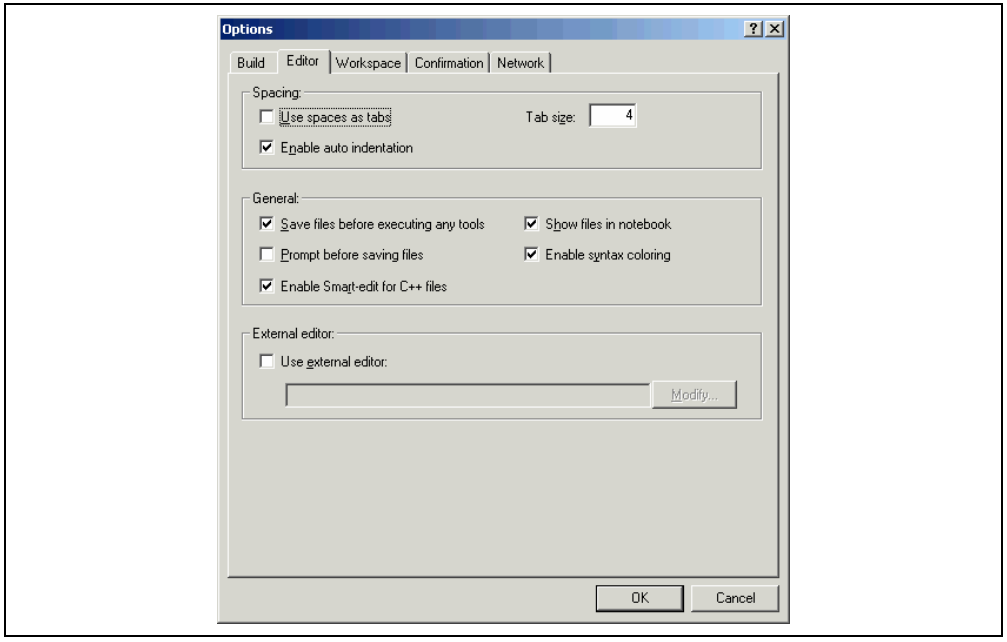


Figure 14.1: Navigation view

With this option switched on if you are working on a C++ file the smart-edit capability should be enabled.

Note: If the C++ Class navigation category has been switched off the Smart-edit capability of HEW will not be active.

During normal usage the following editor operations will make the smart edit facilities visible.

➲ If you are using an object and are trying to access the members using the '.' or '->'. If you do this a pop-up will be displayed which may help you select the correct member more efficiently than typing. Whilst typing the pop-up will keep track of the keys you have pressed to help your selection. If you press return then the currently selected member will be added. This pop-up is also used when using the '::' method and it is displayed in figure 14.2.

➲ If you are trying to use a C++ function then the dialog in Figure 14.3 is displayed when the first open bracket is entered. This dialog allows you to see what functions are available for the current object. Selecting the function automatically enters the remaining parameters for you.

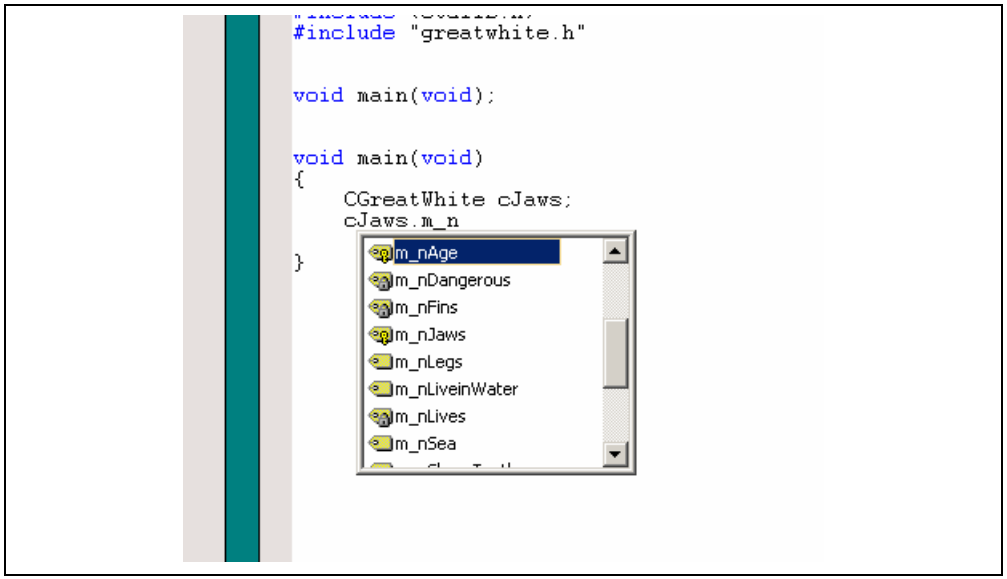


Figure 14.2: Smart-edit member selection

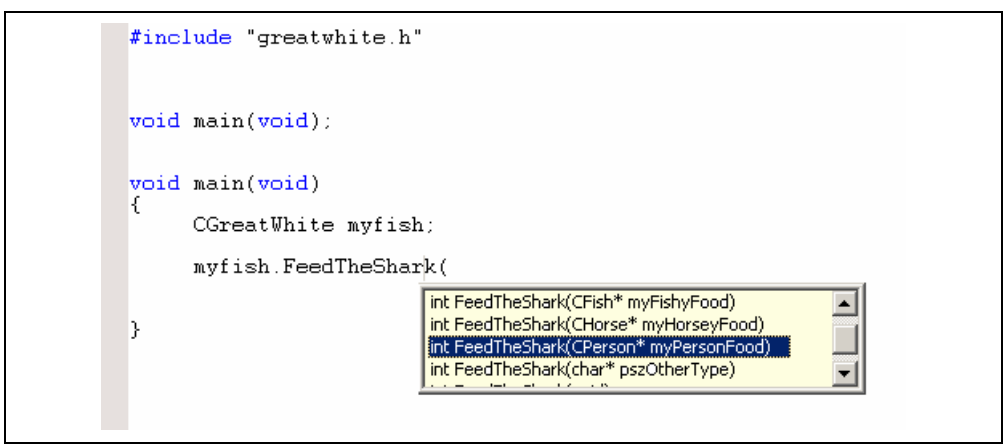


Figure 14.3: Smart -edit function selection

Note: Smart-edit auto completion may not pop-up for a variety of reasons. HEW can only provide help when the file has been scanned by the navigation component. For this to work the file must have been saved.

Another issue is that #defines are not considered so this means that redundant areas of code may still be visible.

Another important consideration is if the syntax of your code is incorrect the smart editor may not be able to parse your code correctly and the smart edit functionality will fail. In this case no pop-up will be displayed.

Please note that Smart-edit does not support macro usage.

Simulator/Debugger Part

Section 1 Overview

This section describes the fundamental concepts of the High-performance Embedded Workshop (HEW) necessary to use the simulator/debugger. It is intended to give users who are unfamiliar with the Windows® operation, the details that are required by subsequent sections.

1.1 Workspaces, Projects, and Files

Just as a word processor allows you to create and modify documents, the HEW allows you to create and modify workspaces.

A workspace can be thought of as a container of projects. Similarly, a project can be thought of as a container of project files. Thus, each workspace contains one or more projects and each project contains one or more files. Figure 1.1 shows this configuration.

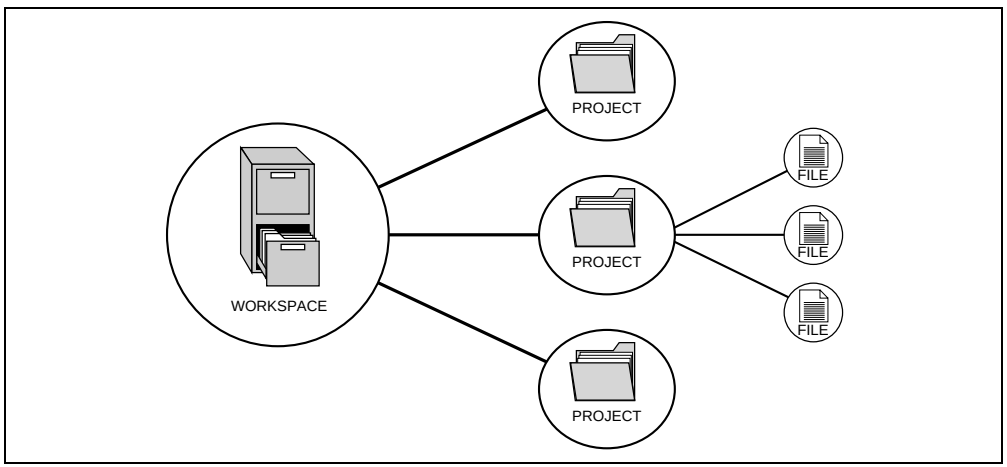


Figure 1.1 Workspaces, Projects, and Files

Workspaces allow you to group related projects together. This is useful when you have an application that needs to be built for different processors, when you are developing an application and library at the same time, or in other cases. 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, first you need to add a project to a workspace and then add files to that project before you can actually do anything.

1.2 Launching the HEW

To initiate the HEW, open the [Start] menu of Windows®, select [Programs], select [Renesas High-performance Embedded Workshop], and then select the shortcut of the HEW. The [Welcome!] dialog box shown in figure 1.2 will be displayed by default.

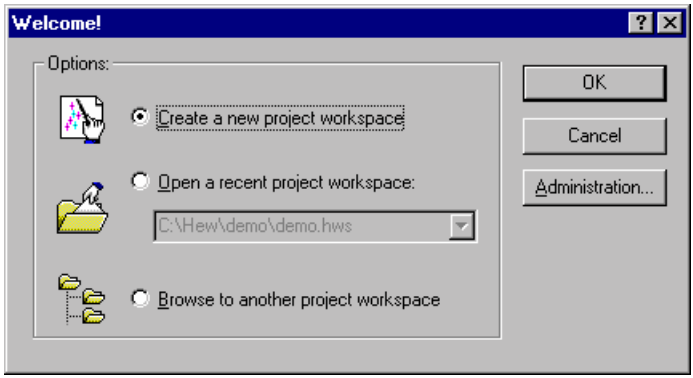


Figure 1.2 Welcome! Dialog Box

To create a new workspace, select the [Create a new project workspace] button, and click the [OK] button. To open one of the 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 content as that seen in the workspace most recently used file list. This list also 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 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 box (figure 1.2) and click the [OK] button or select [File -> New Workspace...]. The [New Project Workspace] dialog box will be displayed (figure 1.3).

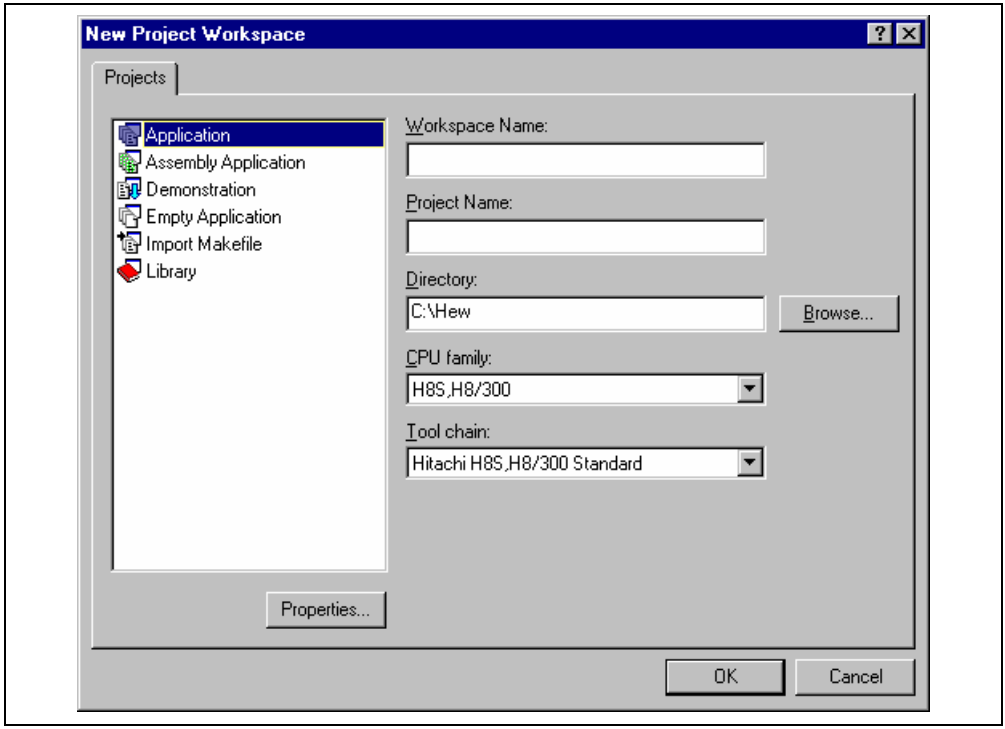


Figure 1.3 New Project Workspace Dialog Box

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. To select the directory in which you would like to create the workspace, use the [Browse…] button or type the directory into the [Directory] field manually. This allows the workspace and project name to be different.
3. Select the CPU family and toolchain upon which you would like to base the workspace. Note that these cannot be changed once the workspace has been created.
4. When a new workspace is created, the HEW will also automatically create a project with the name specified in the project name field and place it inside the new workspace. The [Project types] list displays all of the available project types (application, library, etc.). Select the project type you want to create from this list. The project types displayed will be all valid types for the current pair of CPU family and toolchain. The project types are classified in three classes: toolchain-only, debugger-only, and full project generator that configures both the debugger and toolchain aspect of the 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 box (figure 1.2) and click the [OK] button or select [File -> Open Workspace…]. The [Open Project Workspace] dialog box will be displayed.
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, the [Workspace Properties] dialog box will be displayed (figure 1.4). Otherwise, the workspace will be opened.

   Note that whether the [Workspace Properties] dialog box is shown depends on the setting of either the [Show workspace information on workspace open] check box in the [Workspace Properties] dialog box or the [Display workspace information dialog on opening workspace] check box on the [Workspace] sheet of the [Tools Options] dialog box. Click the [OK] button in the [Workspace Properties] dialog box to open the workspace. Click the [Cancel] button to stop opening the workspace.

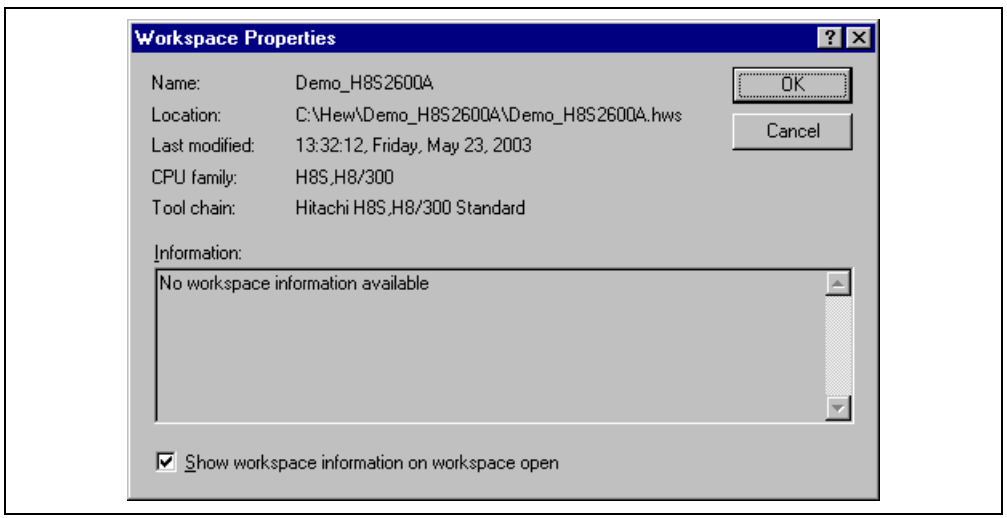


Figure 1.4 Workspace Properties Dialog Box

The HEW keeps track of the last five workspaces that you have opened and adds them to the [File] menu under the [Recent Workspaces] submenu. This gives you a shortcut to opening workspaces, which you have used recently.

➲ To open a recently used workspace

Select [Open a recent project workspace] in the [Welcome!] dialog box, select the name of the workspace from the drop-down list, and then click the [OK] button.

Another way is to select [File -> Recent Workspaces], and then from this submenu select the name of the workspace.

Note: The HEW 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] saves a HEW workspace.

1.6 Closing a Workspace

Selecting [File -> Close Workspace] closes a HEW workspace. If there are any 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 when the workspace is opened. A back-up version of the initial workspace or project file must be saved in the current directory of the file that has been upgraded. The session files used in the Hitachi Debugging Interface will not be upgraded.

1.8 Exiting the HEW

The HEW can be exited by selecting [File -> Exit], pressing the ALT + F4 accelerator, or 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 currently open, the same workspace closing procedure is followed as described in section 1.6, Closing a Workspace.

1.9 Debugger Sessions

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

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

Each session's data should be stored in a separate file in the HEW project. For details, refer to section 3.4, Debugger Sessions.

Section 2 Simulator/Debugger Functions

This section describes the functions of the H8S, H8/300 series simulator/debugger.

2.1 Features

- Since the simulator/debugger runs on a host computer, software debugging can start without using an actual user system, thus reducing overall system development time.
- The simulator/debugger performs a simulation to calculate the number of instruction execution cycles for a program, thus enabling performance evaluation without using an actual user system.
- The simulator/debugger provides pseudo-interrupt and I/O-simulation functions for simple system-level simulation.
- The simulator/debugger offers the following functions that enable efficient program testing and debugging.
  1. The ability to handle all of the H8S, H8/300 series CPUs
  2. Functions to stop or continue execution when an error occurs during user program execution
  3. Profile data acquisition and function-unit performance measurement
  4. A comprehensive set of break functions
  5. Functions to set or edit memory maps
  6. Functions to display function call history
  7. Coverage information is displayed in the C/C++ or assembly-source level
  8. Visual debugging functions provided through the display of images or waveforms
- The breakpoints, memory map, performance, and trace can be set through the dialog boxes under Windows®. Environments corresponding to each memory map of the H8S, H8/300 series microcomputers can be set through the dialog box.
  1. Intuitive user interface
  2. Online help
  3. Common display and operationality

2.2 Target User Program

Load modules in the Elf/Dwarf2 format can be symbolically debugged with the simulator/debugger. Load modules in other formats can be downloaded, and their instructions can be executed, however, they cannot be symbolically debugged. For details, refer to section 4.10, Elf/Dwarf2 Support.

2.3 Simulation Range

The simulator/debugger provides simulation functions for the H8/300, H8/300L, H8/300H, H8S/2600, and H8S/2000 series microcomputers.

The simulator/debugger supports the following H8S, H8/300 series microcomputer functions:

- All CPU instructions
- Exception processing
- Registers
- All address areas
- CPU modes shown in table 2.1

Table 2.1 Platforms and CPU Modes

Names of Debugging Platforms	CPU Modes
H8/300 Simulator	H8/300
H8/300L Simulator	H8/300L
H8/300HA Simulator	H8/300H: advanced mode
H8/300HN Simulator	H8/300H: normal mode
H8S/2600A Simulator	H8S/2600: advanced mode
H8S/2600N Simulator	H8S/2600: normal mode
H8S/2000A Simulator	H8S/2000: advanced mode
H8S/2000N Simulator	H8S/2000: normal mode

The simulator/debugger does not support the following H8S, H8/300 series MCU functions. Programs that use these functions must be debugged with the H8S, H8/300 series emulator.

- Dual-port RAM
- Timer
- Pulse-width modular
- Serial communication interface (SCI)
- A/D converter
- I/O ports
- Interrupt controller

2.4 Memory Management

Memory Map Specification: A memory map is used to calculate the number of memory access cycles during simulation. The following items can be specified:

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

For details, refer to section 4.21.2, Modifying the Memory Map and Memory Resource Settings. The user program can be executed in all areas except for the internal I/O area.

Memory Resource Specification: A memory resource must be specified to load and execute a user program. The following items can be specified:

- Start address
- End address
- Access type

The access type is readable/writable, read-only, or write-only.

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

Unlike the other memory, EEPROM is writable by the EEPMOV instruction even if its access type is read-only. Other instructions than EEPMOV are not available.

For details on memory resource setting, refer to section 3.3.2, Memory Resource.

2.5 Instruction-Execution Reset Processing

The simulator/debugger resets the instruction execution counts and the number of instruction execution cycles when:

- The program counter (PC) is modified after the instruction simulation stops and before it restarts.
- The Run command to which the execution start address has been specified is executed.
- Initialization is performed.

2.6 Exception Processing

The simulator/debugger detects the generation of TRAPA instructions (H8/300H, H8S series only) and trace exceptions (H8S series only) and simulates exception processing. Accordingly, simulation can be performed even when an exception occurs.

The simulator/debugger simulates exception processing with the following procedures.

1. Detects an exception during instruction execution.
2. Saves the PC and SR in the stack area. EXR is also saved when the enabled bit of EXR is on. If an error occurs when saving, the simulator/debugger suspends exception processing, displays the generation of the exception processing error, and returns to the command-wait state.
3. Sets the I bits in CCR to 1.
4. Reads the start address from the vector address corresponding to the vector number. If an error occurs when reading, the simulator/debugger suspends exception processing, displays the generation of the exception processing error, and returns to the command-wait state.
5. Starts instruction execution from the start address. If the start address is 0, the simulator/debugger stops exception processing, displays that an exception processing error has occurred, and enters the command input wait state.

2.7 H8S/2600 CPU Specific Functions

MAC Instruction: The multiplication/accumulation operation (MAC instruction) can be performed on the H8S/2600 CPU. In this instruction, a saturation and non-saturation operations are selectable. The simulator/debugger determines the operation with a value of bit 7 (hereafter called as the MACS bit) in the SYSCR register of internal I/O.

MACS bit 0: Non-saturation operation
MACS bit 1: Saturation operation

EXR Register: The EXR register can be used on the H8S/2600 CPU. It is also possible to set the enable or disable of this register. The simulator/debugger determines the operation with a value of bit 5 (hereafter called as the EXR bit) in the SYSCR register of internal I/O.

EXR bit 0: EXR disabled
EXR bit 1: EXR enabled

The SYSCR address is set in [SYSCR Address] in the [Simulator System] dialog box.

Note: Set the SYSCR address in the internal I/O. Note that the simulator/debugger determines the MACS bit as 0 (non-saturation) and the EXR bit as 0 (EXR disabled) when the SYSCR address is set other than the internal I/O.

For details, refer to section 4.21.1, Simulator System Dialog Box.

2.8 Control Registers

The H8S/2600 series support the SYSCR (system control register) as the control register mapped to the memory. It enables multiplication/accumulation operation, user program simulation for EXR accessing, and debugging.

The SYSCR address is set in [SYSCR Address] in the [Simulator System] dialog box. For modifying or displaying the control register, use the [IO] window.

For details, refer to section 4.22.1, Modifying the Simulator System Settings, or section 4.5, Viewing the I/O Memory.

2.9 Trace

The simulator/debugger writes the execution results of each instruction into the trace buffer. The conditions for trace information acquisition can be specified in the [Trace Acquisition] dialog box. Right-clicking on the [Trace] window displays the pop-up menu. Choose [Acquisition...] from the pop-up menu to display the [Trace Acquisition] dialog box. The acquired trace information is displayed in the [Trace] window.

The trace information can be searched. The search conditions can be specified in the [Trace Search] dialog box. Right-clicking on the [Trace] window displays the pop-up menu. Choose [Find...] from the pop-up menu to display the [Trace Search] dialog box.

For details, refer to section 4.14, Viewing the Trace Information.

2.10 Standard I/O and File I/O Processing

The simulator/debugger enables the standard I/O and file I/O processing listed in table 2.2 to be executed by the user program. When the I/O processing is executed, the [I/O Simulation] window must be open.

Table 2.2 shows the supported I/O functions. The function codes have 16-bit address, 24-bit address, and 32-bit address versions. Select the version according to the CPU to be used.

Table 2.2 I/O Functions

No.	Function Code	Function Name	Description
1	H'01 (16-bit address) H'11 (24-bit address) H'21 (32-bit address)	GETC	Inputs one byte from the standard input device
2	H'02 (16-bit address) H'12 (24-bit address) H'22 (32-bit address)	PUTC	Outputs one byte to the standard output device
3	H'03 (16-bit address) H'13 (24-bit address) H'23 (32-bit address)	GETS	Inputs one line from the standard input device
4	H'04 (16-bit address) H'14 (24-bit address) H'24 (32-bit address)	PUTS	Outputs one line to the standard output device
5	H'05 (16-bit address) H'15 (24-bit address) H'25 (32-bit address)	FOPEN	Opens a file
6	H'06	FCLOSE	Closes a file
7	H'07(16-bit address) H'17 (24-bit address) H'27 (32-bit address)	FGETC	Inputs one byte from a file
8	H'08 (16-bit address) H'18 (24-bit address) H'28 (32-bit address)	FPUTC	Outputs one byte to a file
9	H'09 (16-bit address) H'19 (24-bit address) H'29 (32-bit address)	FGETS	Inputs one line from a file
10	H'0A (16-bit address) H'1A (24-bit address) H'2A (32-bit address)	FPUTS	Outputs one line to a file
11	H'0B	FEOF	Checks for end of the file
12	H'0C	FSEEK	Moves the file pointer
13	H'0D	FTELL	Returns the current position of the file pointer

For details on I/O functions, refer to section 4.22, Standard I/O and File I/O Processing.

2.11 Calculation of the Number of Instruction Execution Cycles

The simulator/debugger calculates the number of instruction execution cycles by using the expression, the data-bus width of the memory map, and the number of access cycles, which is described in the H8S, H8/300 Series Programming Manual. However, note that the number of instruction execution cycles may differ when the number of instruction execution cycles that is calculated on the simulator/debugger and the one that the program is executed on the system.

MOVFPE and MOVTPE Instructions: The number of cycles for transferring data of the E-clock synchronization instruction is the value between 9 to 16. The simulator/debugger calculates as '11 + number of operand access cycles'. The number of operand access cycles can be calculated from the data bus width of the memory and the number of access cycles.

EEPMOV Instruction: The number of cycles for the EEPROM-write instruction is the sum of the number of cycles for reading instructions and cycles for transferring data.

SLEEP Instruction: The simulator/debugger does not add the number of cycles considering the case when the SLEEP instruction is used for halting program.

Standard I/O and file I/O processing: The standard I/O and file I/O processing are specific functions for the simulator/debugger, and they are not added to the number of cycles. Note that the standard I/O and file I/O processing are the period while the branch to the position specified with the system-call address of the BSR and JSR instructions is completed to return to the called source after the I/O processing.

2.12 Break Conditions

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

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

Break Due to Satisfaction of a Break Command Condition: There are five break commands as follows:

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

If [Stop] is specified as the action for when a break condition is satisfied, user program execution stops when the break condition is satisfied. For details, refer to section 4.15, Using the Simulator/Debugger Breakpoints.

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

Table 2.3 Processing When a Break Condition is Satisfied

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

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

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

Break Due to Error Detection during User Program Execution: The simulator/debugger detects simulation errors, that is, program errors that cannot be detected by the CPU exception generation functions. The [Simulator System] dialog box specifies whether to stop or continue the simulation when such an error occurs. Table 2.4 lists the error messages, error causes, and the action of the simulator/debugger in the continuation mode.

Table 2.4 Simulation Errors

Error Message	Error Cause	Processing in Continuation Mode
Address Error	Odd PC value	Operates in the same way as the actual device operation.
	Instruction was fetched from the internal I/O area	
	Access in words from odd-number addresses	
	Access in longwords from odd-number addresses	
Memory Access Error	Access to a memory area that has not been allocated	On memory write, nothing is written; on memory read, all bits are read as 1.
	Write to a memory area having the write protect attribute	
	Read from a memory area having the read disable attribute	
	Access to an area where memory does not exist	
	Write to the EEPROM by instructions other than EEPMOV	
Illegal Instruction	Execution of a code that is not an instruction	Always stops
	Execution of MOV.B Rn, @-SP or MOV.B @SP + Rn	Continues simulation but the result is not guaranteed
Illegal Operation	Incorrect relation between the values in the C flag or H flag of the CCR in the DAA or DAS instruction and the values before adjustment	Continues simulation but the result is not guaranteed
	Zero division executed by the DIVXU or DIVXS instruction, or overflow	

When a simulation error occurs in the stop mode, the simulator/debugger returns to the command input wait state after stopping instruction execution and displaying the error message. Table 2.5 lists the states of the program counter (PC) at simulation error stop. The condition code register (CCR) value does not change at simulation error stop.

Table 2.5 Register States at Simulation Error Stop

Error Message	PC Value
Address Error, Memory Access Error	• When an instruction is read: The start address of the instruction that caused the error. • When an instruction is executed: The instruction address following the instruction that caused the error.
Illegal Instruction	The start address of the instruction that caused the error.
Illegal Operation	The instruction address following the instruction that caused the error.

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

1. First execute the program in the stop mode and confirm that there are no errors except those in the intended locations.
2. After confirming the above, execute the program in the continuation mode.

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

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

```
Trace Buffer Full
```

Break Due to Execution of the SLEEP Instruction: When the SLEEP instruction is executed during instruction execution, the simulator/debugger stops execution. The following message is displayed in the [Output] window when execution is stopped.

```
Sleep
```

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

Break Due to the [Halt] Button: Users can forcibly terminate execution by clicking the [HALT] button during instruction execution. The following message is displayed on the status bar when execution is stopped.

```
Stop
```

Execution can be resumed with the GO or STEP command.

2.13 Floating-Point Data

Floating-point numbers can be used for the following real-number data, which makes floating-point data processing easier. The following data can be specified for floating-point data:

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

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

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

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

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

2.14 Display of Function Call History

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

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

- When simulation stops due to the break conditions described in section 2.16, Break Conditions.
- When register values are modified while simulation stops due to the above break conditions.
- While single-step execution is performed.

For details, refer to section 4.13, Viewing the Function Call History.

2.15 Performance Measurement

The simulator/debugger has the profiler function and performance analysis function for performance measurement of the user program.

2.15.1 Profiler

The profiler function displays the memory address and size allocated to functions and global variables, the number of function calls, and the profile data for the entire user program. The profile data displayed differs according to the CPU.

Profile information is displayed in list, tree, and chart formats.

Profile information is useful in optimizing user programs by reducing the size and putting the most frequently called functions in-line.

When using the profile information saved in a file, it is possible to optimize user programs based on dynamic information using the optimizing linkage editor.

For details, refer to section 4.12, Viewing the Profile Information, and section 4.2.3, Optimize Option Profile in the Optimizing Linkage Editor User's Manual.

2.15.2 Performance Analysis

The performance analysis function displays the number of execution cycles and function calls for the specified function in the user program. Since performance data for only the specified function is acquired, simulation can be done faster. For details, refer to section 4.16, Analyzing Performance.

2.16 Pseudo-Interrupts

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

1. Pseudo-interrupts generated by break conditions
   A pseudo-interrupt can be generated using a break command to specify [Interrupt] as the action when a break condition is satisfied. For details, refer to "Specifying the Action at Break" in section 4.15.2, Setting a Breakpoint.
2. Pseudo-interrupts generated from the [Trigger] window
   A pseudo-interrupt can be generated by clicking a button in the [Trigger] window. For details, refer to section 4.20, Generating a Pseudo-Interrupt Manually.

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

Note: In the pseudo-interrupts, whether an interrupt is accepted is determined by interrupt information [Priority], not the vector number. Note that when H'8 is specified as the priority level of an interrupt, that interrupt is always accepted. The simulator/debugger does not simulate the operation of the interrupt controller.

2.17 Coverage

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

In the measurement range, addresses are directly specified, and all functions in a file whose name has been specified are set.

The status of each instruction execution can be monitored through the instruction coverage information. In addition, this information can be used to determine which part of a program has not been executed.

The [Coverage] window displays the acquired instruction coverage information:

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

For the address range or function to be measured, the coverage statistical information is displayed in a percentage format, which is helpful to know the amount of program that has been executed.

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

For details, refer to section 4.17, Acquiring Code Coverage.

Section 3 Preparations for Debugging

This section describes the preparations for debugging your program. You will learn how to select and configure a debugging platform with which to debug, how to load the user program, and what the debugger sessions are.

3.1 Building before Debugging

To debug your program in the 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. If either of these files is used for this purpose, it is usually called a debug object file. When you create your project, the initial setup will normally be configured for debugging.

Note: Make sure you have the [Debug] option enabled in your compiler and linker, when you generate an object file for debugging.
Even if your debug object file does not contain any debugging information (for example, the S-record format), you can still load it into the debugging platform, but you will only be able to debug it in the assembly-language level.

3.2 Selecting a Debugging Platform

Selecting the debugging platform is very dependent on the installation method of the HEW. If the HEW has a toolchain installed, the application project generator will be able to set up both the toolchain and the debugger targets simultaneously. This allows the options for the targets and toolchain to be matched closely so no inconsistencies occur. If there is no toolchain installed, you will only be able to select debug-only project types. By default, the HEW will display the debug-only project generation type for each CPU family in the [New Project Workspace] dialog box.

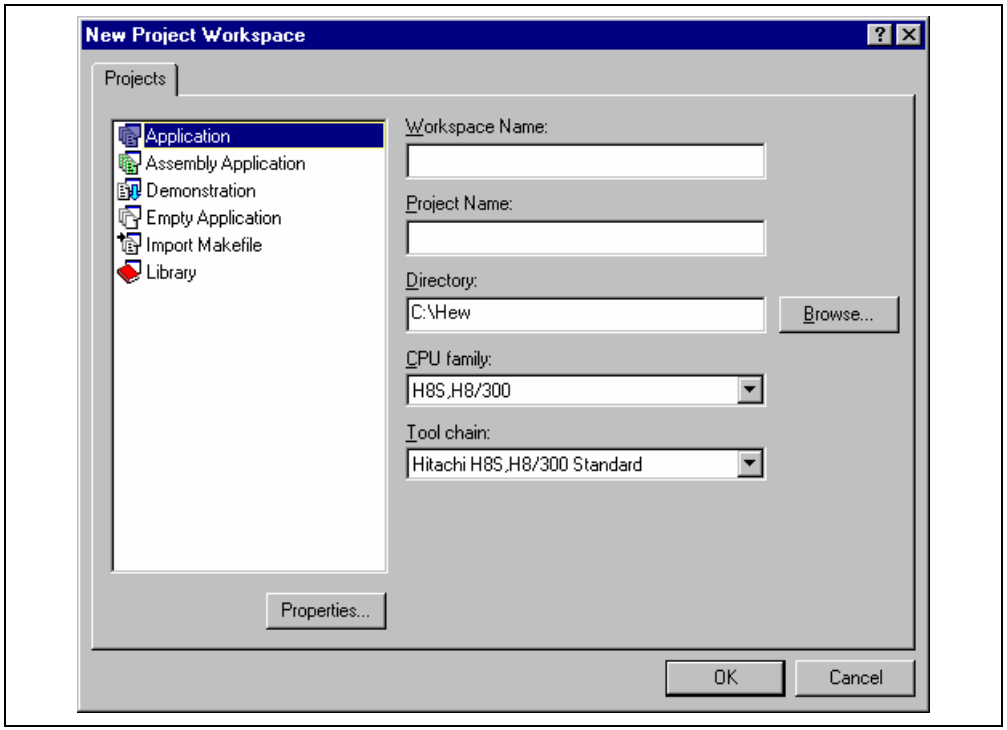


Figure 3.1 New Project Workspace Dialog Box

The dialog box shown in figure 3.1 allows you to select a project generation type that matches your target CPU.

[Application]	Project for generating an execution program that includes the initial routine file written in the C/C++ language.
[Assembly Application]	Project for generating an execution program that includes the initial routine file written in the assembly language.
[Demonstration]	Project for generating a demonstration program written in the C language.
[Empty Application]	Project for only setting the toolchain environment (no generation file).
[Import Makefile]	A project to create an executable program by importing an existing makefile.
[Library]	Project for generating a library file (no generation file).

Note: Select [Empty Application] to generate a debug-only project.

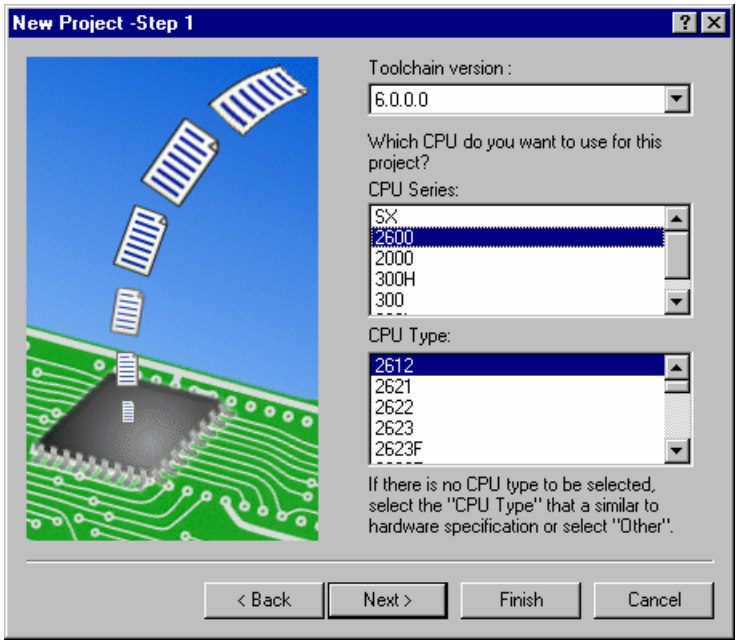


Figure 3.2 CPU Selection Display (Step 1)

1. Select the CPU and Toolchain version in Step 1. The CPU types ([CPU Type]) are classified according to the CPU series ([CPU Series]). Select the CPU corresponding to the program to be developed because the generation file differs according to the [CPU Type] and [CPU Series] settings. If there is no corresponding CPU, select a CPU with similar hardware specifications or [Other].

   The following buttons at the bottom of the dialog box are the same as those in the [New Project] wizard dialog box.

   [Next>] Moves to the next display.

   [<Back] Returns to the previous display.

   [Finish] Opens the [Summary] dialog box (selections followed by this button are default).

   [Cancel] Returns to the [New Project Workspace] dialog box.

   To move to Step 2, click the [Next>] button in Step 1.

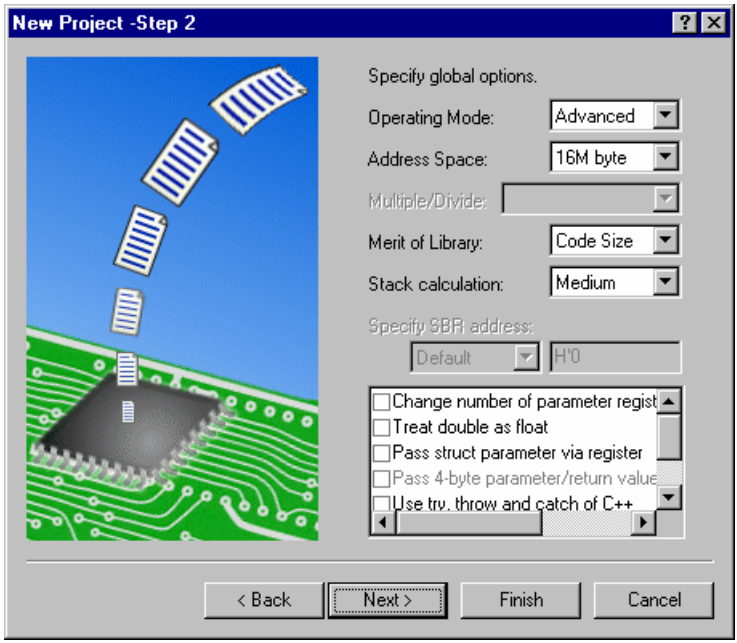


Figure 3.3 Option Setting Display (Step 2)

2. Specify the options common to all project files in Step 2. The specifiable items depend on the CPU selected in Step 1.

List Box Settings:

[Operating Mode]		Specifies the operating mode of the object to be created.
	[Normal]	Normal mode.
	[Advanced]	Advanced mode.
[Address Space]		Specifies the address space of the object to be created.
	[1M byte]	Specifies 1-Mbyte address space.
	[16M byte]	Specifies 16-Mbyte address space.
	[256M byte]	Specifies 256-Mbyte address space.
	[4G byte]	Specifies 4-Gbyte address space.
[Merit of Library]		Specifies whether the priority of the standard library is speed or code size.
	[Code Size]	Reduces size.
	[Speed]	Execution speed.
[Stack calculation]		Specifies the range of the stack address.
	[Small]	1-byte calculation of the stack address.
	[Medium]	2-byte calculation of the stack address.
	[Large]	4-byte calculation of the stack address.

Check Box Settings:

[Change number of parameter registers from 2(default) to 3]	Three registers for storing parameters. There are two registers by default.

[Treat double as float]	Generates an object making the value of the double-precision floating-point type as the single-precision floating-point type.
[Pass struct parameter via register]	Specifies whether or not the parameter of the structure is allocated to the register. When checked, the parameter is passed with the register. When not checked, the parameter is passed with the memory, not with the register.
[Pass 4-byte parameter/return value via register]	Specifies whether or not the 4-byte parameter or the return value is allocated to the register. When checked, the parameter is passed with the register. When not checked, the parameter is passed with the memory, not with the register.
[Use try, throw and catch of C++]	Checking enables the C++ exception processing functions (try, catch, and throw).
[Enable/disable runtime type information]	Enables or disables the type information at execution. When checked, dynamic_cast and typeid are available. When not checked, dynamic_cast and typeid are unavailable. For details, refer to the compiler manual.

To move to Step 3, click the [Next>] button in Step 2.

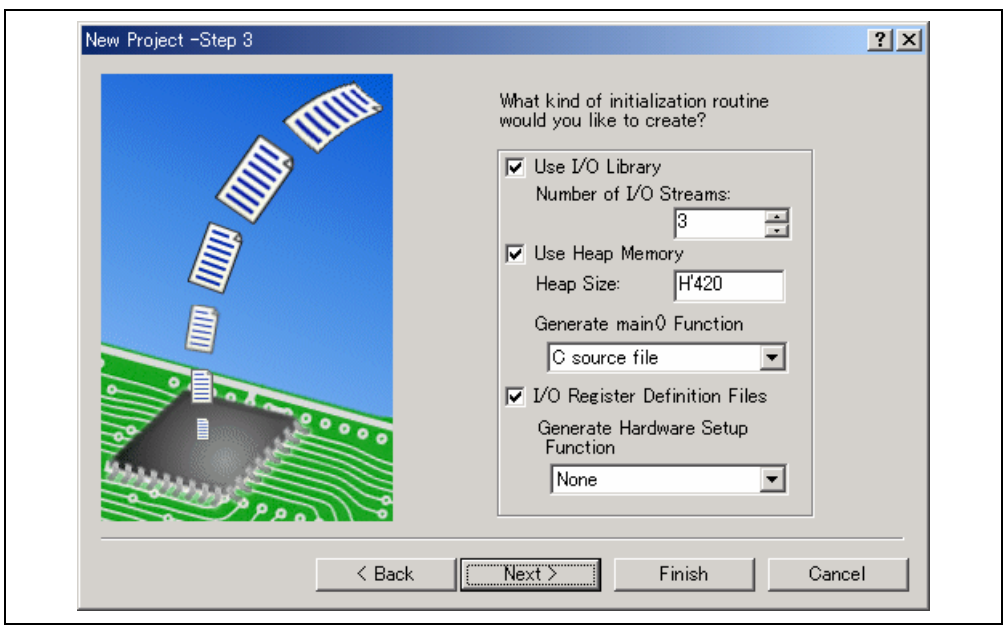


Figure 3.4 Generation File Setting Display (Step 3)

3. Specify the generation file in Step 3.

[Use I/O Library]	Checking enables use of standard I/O libraries.	
	[Number of I/O streams]	Specifies the number of I/O streams that can be used simultaneously.
[Use Heap Memory]	Checking enables use of the heap area management function sbrk( ).	
	[Heap Size]	Specifies the unit of the size of the heap area to be managed.

[Generate main( ) Function] Selects generation of a model main function.
Generates a main function file [(Project name).c(cpp)].

[I/O Register Definition Files] Checking generates an I/O register definition file (iodefine.h) written in the C language.

[Generate Hardware Setup Function] Selects generation of a model I/O register initial setting program.
Generates a hardware setting file (hwsetup.c(cpp) or hwsetup.src).

Note: To include a main function that has already been made, select [None] in [Generate main( ) Function] and after making the project, add the file containing the main function to the project. Note that if the name of the function to be included is different, the function calling section in resetprg.c must be modified. Be sure to refer to the hardware manual of the CPU for actual values of the sample file contents, such as the vector table definition and I/O register definition, which are generated by the project generator.

To move to Step 4, click the [Next>] button in Step 3.

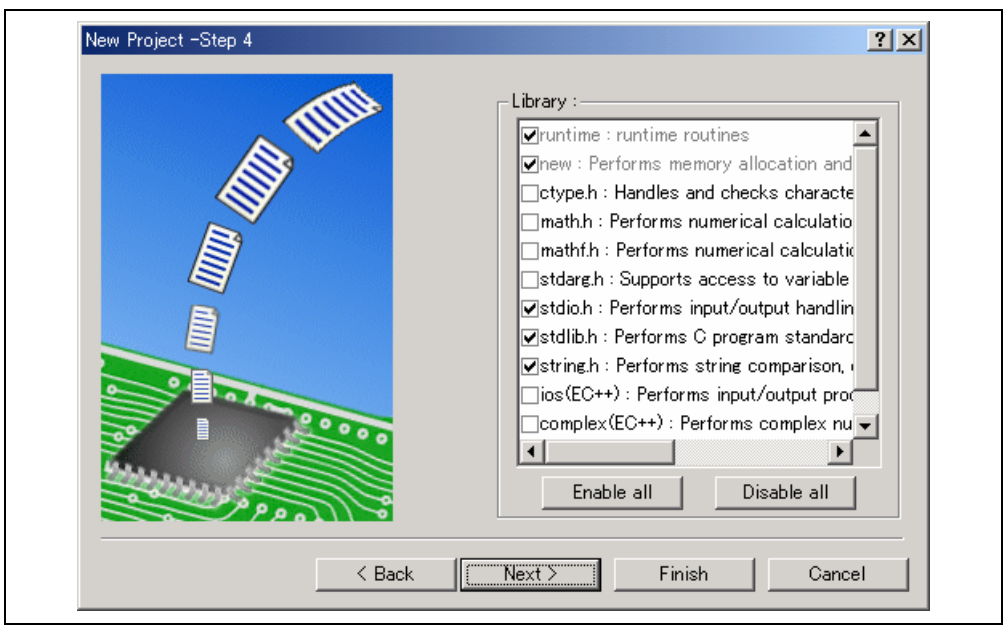


Figure 3.5 Standard Library Setting Display (Step 4)

4. Specify the configuration of the standard libraries used by the C/C++ compiler in Step 4.
   The functions defined in the checked items and the runtime function are included.

   [Enable all] Selects all standard library functions.

   [Disable all] Does not select all standard library functions.
   Note that only the minimum required functions, runtime and new, are selected.

   To move to Step 5, click the [Next>] button in Step 4.

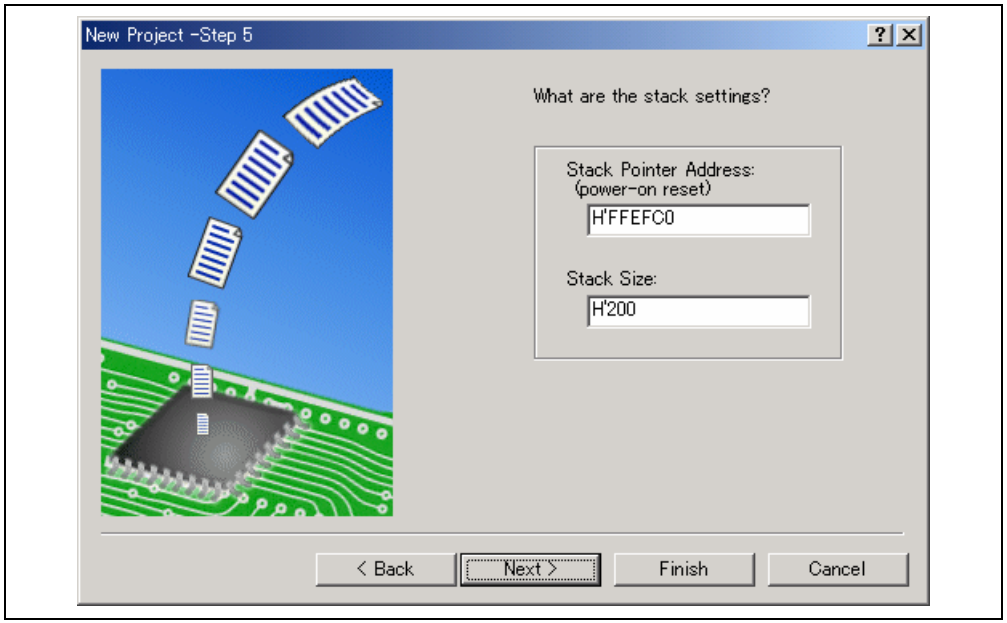


Figure 3.6 Stack Area Setting Display (Step 5)

5. Specify the stack area in Step 5. This is done by setting the initial value of the stack pointer and the stack size. The initial value of the stack areas depends on the CPU selected in Step 1.

   Note: The stack area is defined by stacksct.h which is generated by the HEW. If stacksct.h has been modified by an editor, it cannot be modified from [Project -> Edit Project Configuration] in the HEW.

   To move to Step 6, click the [Next>] button in Step 5.

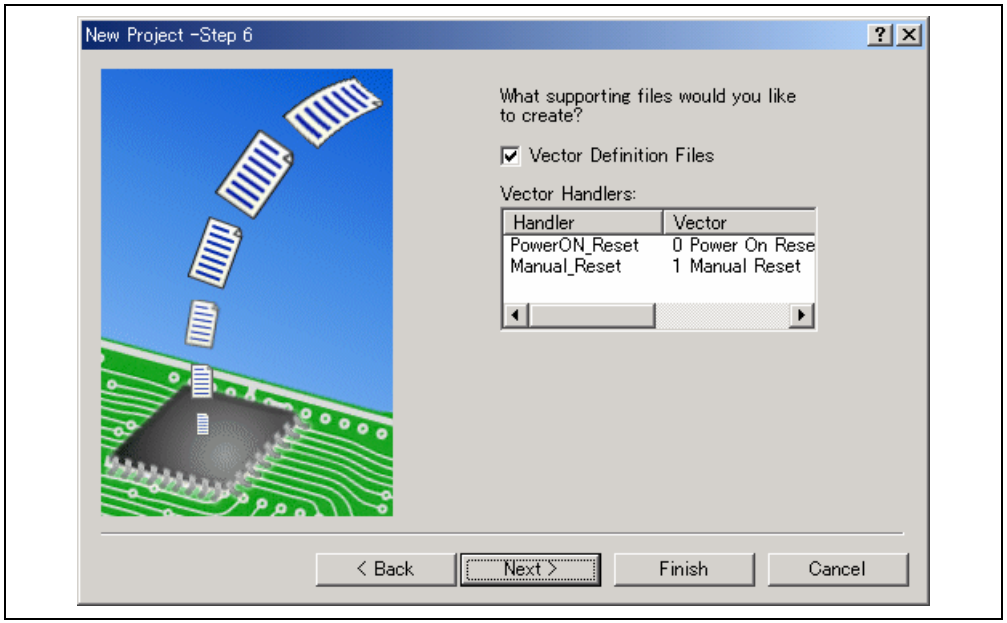


Figure 3.7 Vector Setting Display (Step 6)

6. Specify the vector in Step 6.

   [Vector Definition Files] Checking generates a vector definition file and a vector table setting function definition file.

   [Vector Handlers] [Handler] Displays the handler program name of the reset vector. To modify the handler program, after selecting the handler program name by clicking on it, enter the new handler program name. Note that if the handler program is modified, a reset program (resetprg.c) is not generated.

   [Vector] Displays a description of the vector.

   Note: Since the generated reset program and interrupt functions are samples, be sure to refer to the CPU hardware manual.

   To move to Step 7, click the [Next>] button in Step 6.

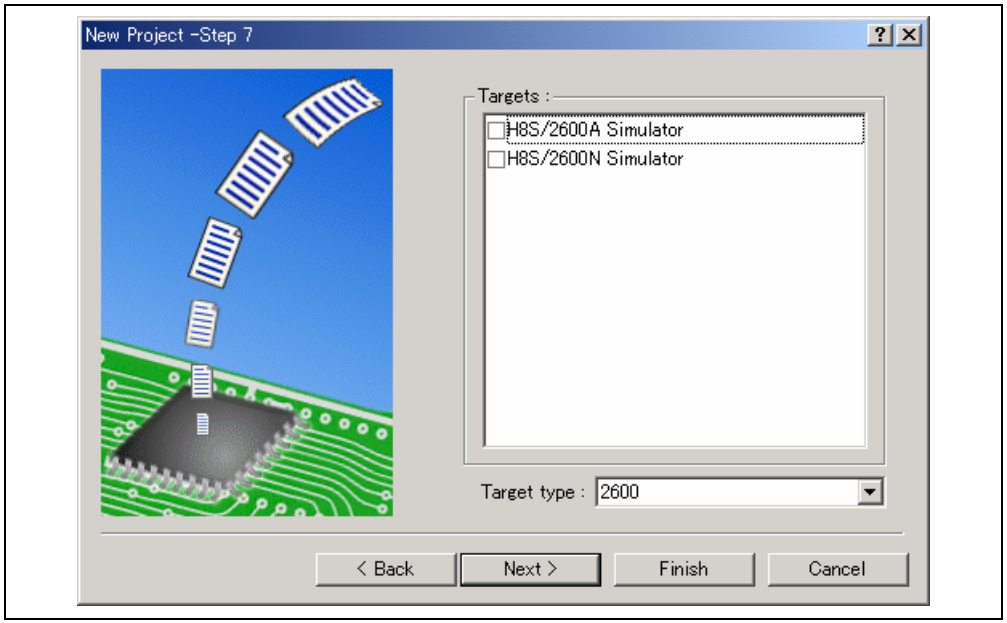


Figure 3.8 Debugger Target Setting Display (Step 7)

7. Specify the debugger targets in Step 7.

   [Targets] Sets the debugger targets. Select (by checking) the debugger targets. No selection or a selection of more than one target is possible.

   [Target Type] Specifies the type of the targets displayed in [Targets].

   To move to Step 8, click the [Next>] button in Step 7.

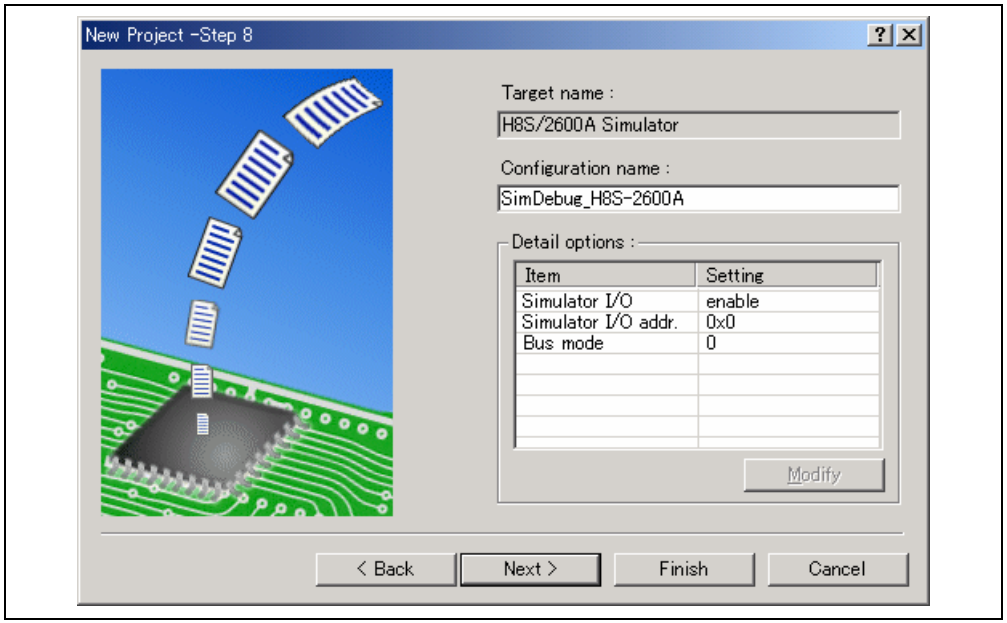


Figure 3.9 Debugger Option Setting Display (Step 8)

8. Set the options for the debugger targets selected in Step 8.

   [Configuration name] By default, the HEW generates two configurations: [Release] and [Debug]. If a debugger target is selected, a configuration for the selected target is also generated (an abbreviation including the target name). This configuration name can be changed in [Configuration name].

   [Detail options] Sets the debugger target options. To modify an option, select [Item] and click [Modify]. If the selected item cannot be modified, [Modify] remains gray even when [Item] is selected.

   - [Simulator I/O] System call for standard I/O or file I/O from the user program is enabled ([Enable]) or disabled ([Disable]).
   - [Simulator I/O addr.] Address for above system call.
   - [Bus mode] Currently cannot be used by the simulator/debugger.

   To move to Step 9, click the [Next>] button in Step 8.

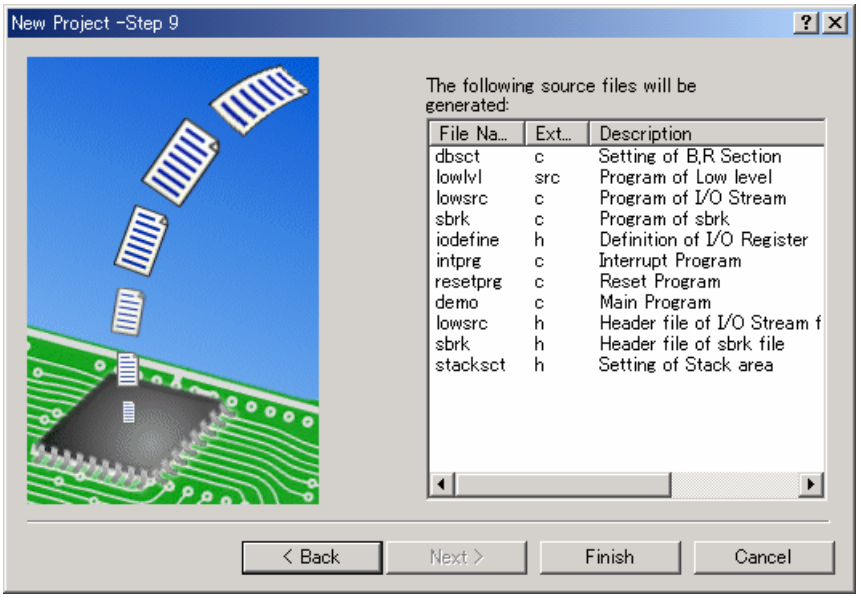


Figure 3.10 Generation File Name Modification Display (Step 9)

9. The files to be generated by the HEW based on the settings made so far is displayed as a list in Step 9.

   [File Name] File name
   To change a file name, after selecting the file name by clicking on it, enter the new file name.

   [Extension] File extension

   [Description] Description of the file

   Clicking the [Finish] button in Step 9 displays the [Summary] dialog box.

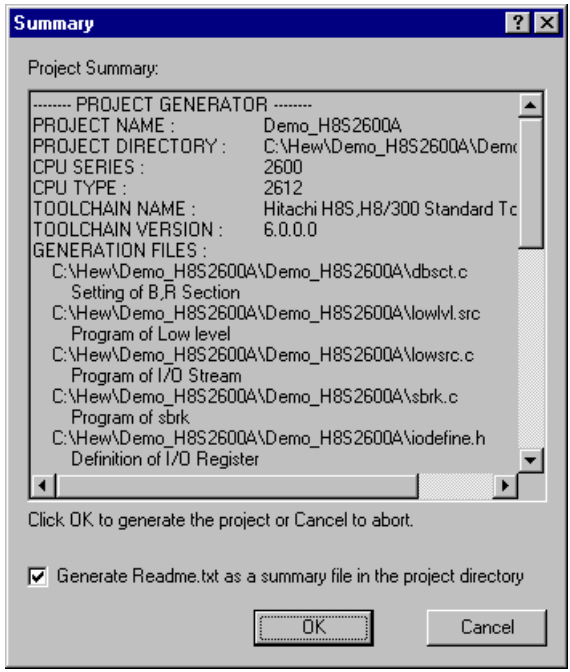


Figure 3.11 Summary Dialog Box

10. The project generator displays information on the project to be generated in the [Summary] dialog box. After confirming the display contents, click [OK].

    Checking [Generate Readme.txt as a summary file in the project directory] will save the project information displayed in the [Summary] dialog box as a text file named Readme.txt in the project directory.

3.3 Configuring the Debugging Platform

Before loading a program into your debugging platform, the platform must be set up to match your application system. The items that must be set are typically the device type, operating mode, clock speed, and memory map. It is particularly important to set the memory map. In the HEW, the project generation process will have completed much of this work. However if you are using a board with a configuration different from the standard types, then some customization will be required.

3.3.1 Memory Map

The debugging platform needs to know which areas in the device's address space are RAM, ROM, on-chip registers, or areas where memory is not allocated. Therefore, the memory map must be set up.

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.

When external memory is used, the debugging platform requires to be informed.

When using the simulator, the memory map settings can be modified in the [Memory] tab of the [Simulator System] dialog box. For details, refer to section 4.21.2, Modifying the Memory Map and Memory Resource Settings.

In this case, the settings can be referred to in the [Simulator] sheet of the [Standard Toolchain] dialog box. Clicking the [View] button of the [Standard Toolchain] dialog box (figure 3.12) displays the [CPU hardware map] dialog box (figure 3.13).

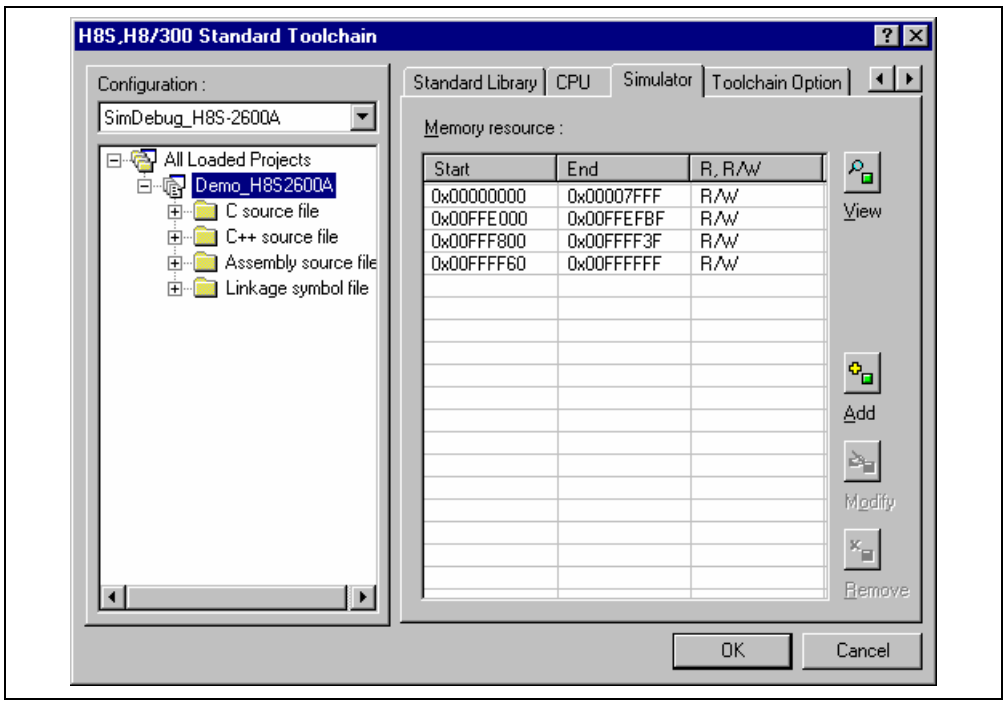


Figure 3.12 Standard Toolchain Dialog Box

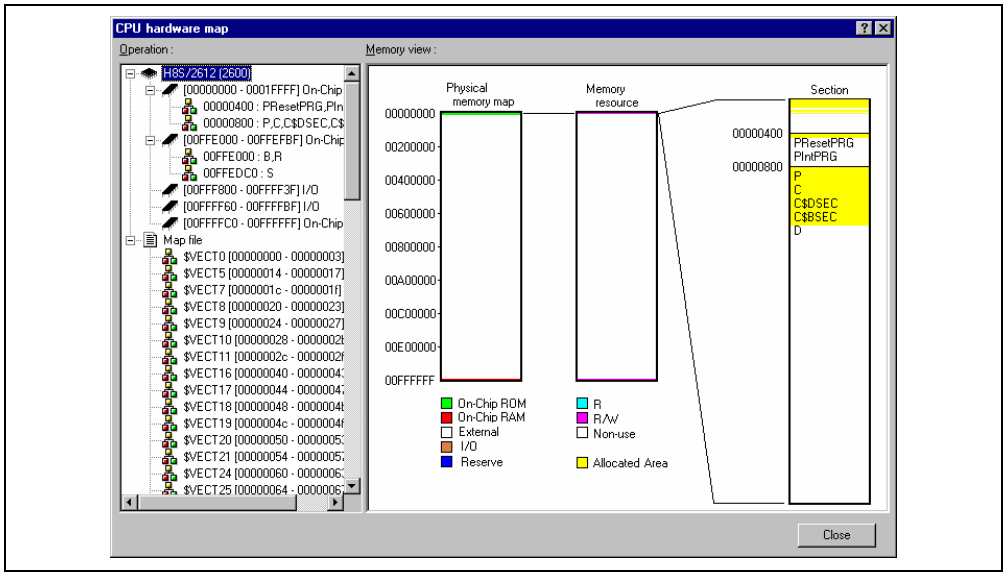


Figure 3.13 CPU hardware map Dialog Box

The following information is displayed:

[Operation] Displays the address range in the upper step.
Displays the linkage map in the lower step.

[Memory View] Displays memory map and memory resource information.
A memory resource already allocated (Allocated Area) is displayed in yellow.

3.3.2 Memory Resource

To use the simulator, the memory resource of the address range being used must be saved.

In this case, the settings can be referred to in the [Simulator] sheet of the [Standard Toolchain] dialog box (figure 3.12). The [Simulator] sheet contains the following buttons:

[View] Displays the memory map information.

[Add] Adds a 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 3.14).

Figure 3.14 Add memory resource Dialog Box

Specify the following items in the [Add memory resource] dialog box:

[Start] Start address

[End] End address

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

The memory resource settings can be changed in the [Memory] tab of the [Simulator System] dialog box. For details, refer to section 4.212, Modifying the Memory Map and Memory Resource Settings.

Modification of the memory resource information in the [Memory] tab of the [Simulator System] 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 [Memory] tab of the [Simulator System] dialog box as well.

3.3.3 Downloading a Program

If there is sufficient memory in the system in which to download your program, you can then proceed to download a program to debug. By default, the program output from the linker of the current project is automatically downloaded.

It is also possible to manually choose the download modules after the project creation. This is achieved via the [Debug Settings] dialog box shown in figure 3.16. This dialog box allows you to control the debug settings throughout your workspace. The tree on the left of the dialog box 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-down list. Multiple sessions or all sessions in this list box can be selected. If you select multiple sessions, you can choose to modify the settings for one or more sessions at once. The [Debug Settings] dialog box 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 box shown in figure 3.16 will be displayed.
2. Select the project and configurations you are interested in adding a download module to in [tree control].
3. Click the [Add] button. The dialog box displayed in figure 3.15 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 [Filename] field allows you to browse to the download module on your disk, or simply type it into [edit control] if you have not yet built the module.
6. Enter an offset address to the [Offset] field (suitable only for some object formats).
7. Specify the memory access size as 1, 2, 4, or 8.
8. To download only the debugging information, check the [Download debug information only] check box.
9. To verify memory during downloading, check the [Perform memory verify during download] check box.
10. When the user clicks the [OK] button, 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 [Up] and [Down] buttons to position the module correctly.

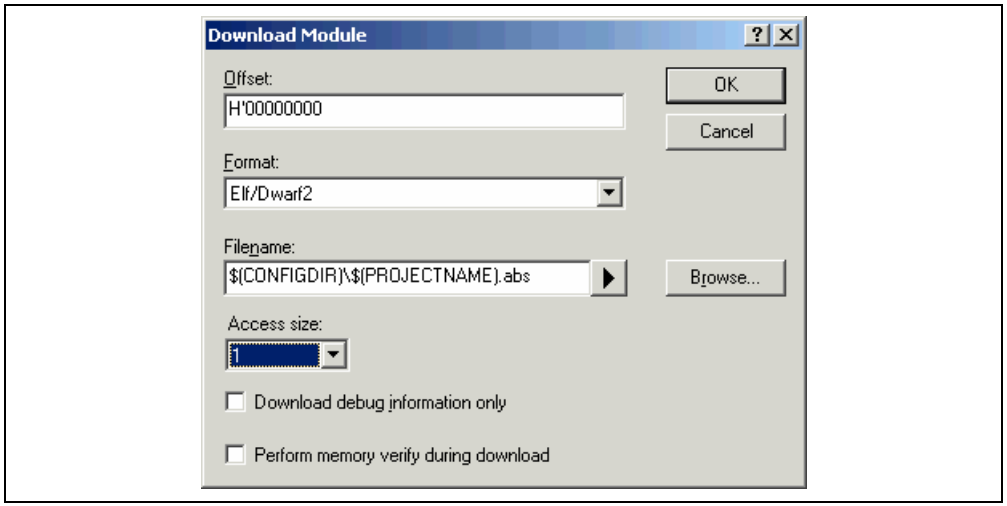


Figure 3.15 Download Module Dialog Box

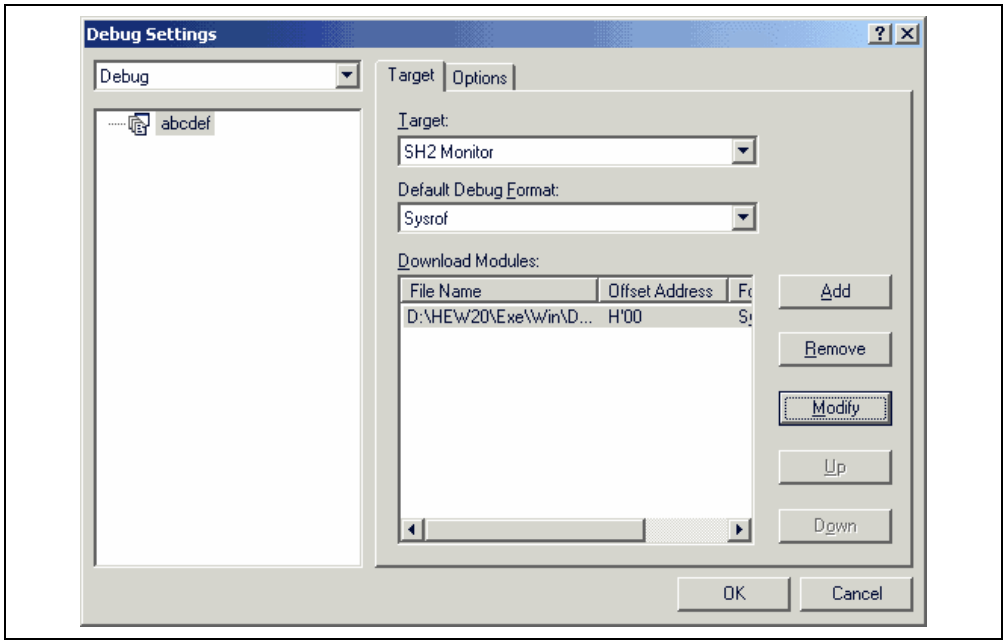


Figure 3.16 Debug Settings Dialog Box ([Target] Sheet)

Any changes made in the [Debug Settings] dialog box do not take effect until you click the [OK] button.

The default debug format is set to the first download module in the list by default. Only one default debug object format can be specified for each session. All currently installed debugger formats are listed here.

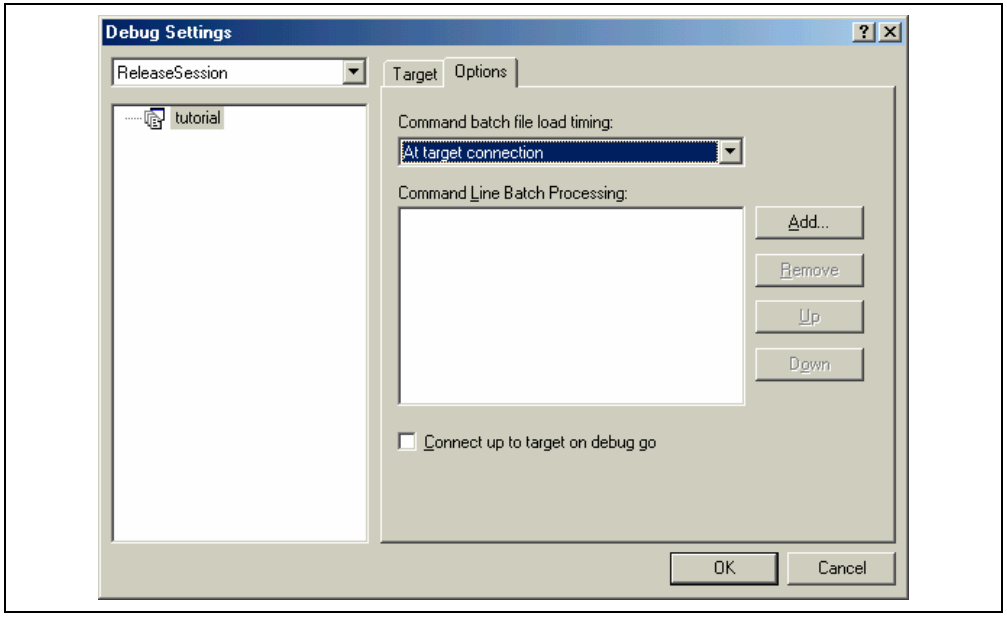


Figure 3.17 Debug Settings Dialog Box ([Options] Sheet)

The [Options] sheet (figure 3.17) of the [Debug Settings] dialog box contains the option that determines when a target is connected. If the option is checked, the target is not connected until you select [Build -> Debug -> Go]. If the option is unchecked, the target is connected whenever a configuration is opened. This happens when a new workspace or project is opened or also when you switch configuration using the toolbar or [Build Configurations] dialog box.

The [Command Line Batch Processing] list in the sheet contains a list of command line batch commands. These batch commands can perform debug operations via the command line. When these commands are executed can be selected in the [Command batch file load timing] drop-down list. Add the batch file to be executed after selecting an item in the [Command batch file load timing] drop-down list. The order in the list is the order in which the command line batch files will be executed when the timing event occurs.

3.3.4 Manual Download of Modules

Once you have decided which modules are to be downloaded to the target, it is possible to manually update the modules of the connected target 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 modules under the [Download Modules] folder in the [Workspace] window.

3.3.5 Automatic Download of Modules

When you select [Debug -> Run…], the HEW calculates whether any of the files or debugger settings that 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 module is downloaded to the target.

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

3.3.6 Unloading of Modules

It is possible to manually unload downloaded modules in the list whenever you want from the [Debug -> Unload Modules] menu item. This can also be done from the [Download Module] pop-up menu in the [Workspace] window.

When a module is unloaded, its symbols are erased from the HEW debugging system, but the memory contents of the target remains unmodified. After a module has been unloaded, it cannot be debugged unless it is reloaded.

3.4 Debugger Sessions

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

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

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

3.4.1 Selecting a Session

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

- From the toolbar
  Select a session from the drop-down list box (figure 3.18) in the toolbar.

Figure 3.18 Toolbar Selection

- From the dialog box
  1. Select [Options -> Debug Sessions…]. This will open the [Debug Sessions] dialog box (figure 3.19).

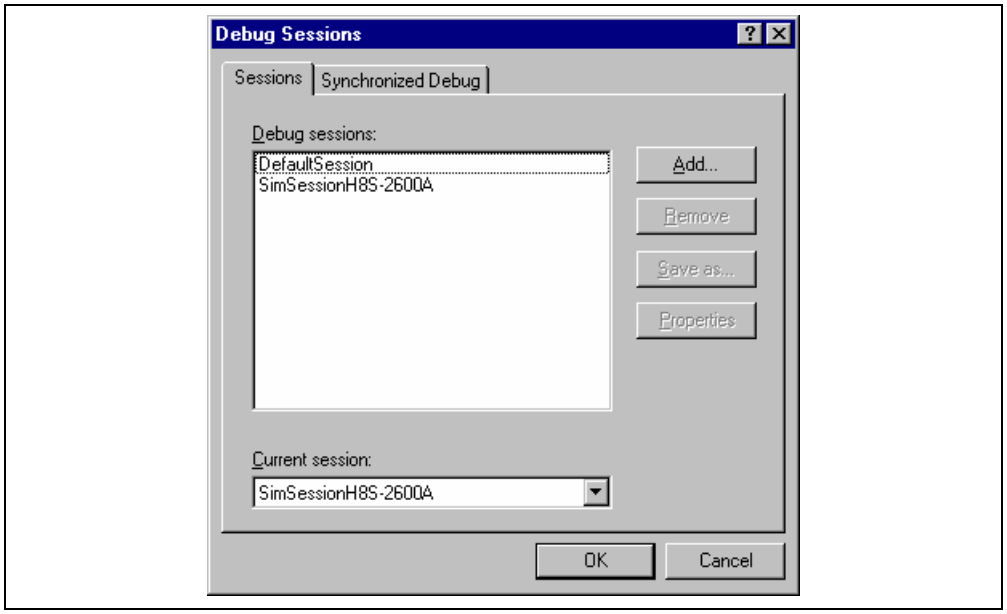


Figure 3.19 Debug Sessions Dialog Box

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

3.4.2 Adding and Deleting Sessions

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

- To add a new empty session
  1. Select [Options -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 3.19).
  2. Click the [Add…] button to display the [Add new session] dialog box (figure 3.20).
  3. Check the [Add new session] radio button.
  4. Enter a name for the session.
  5. Click the [OK] button to close the [Debug Sessions] dialog box.
  6. This creates a file with the same name as the entered session name. If the file name already exists, an error is displayed.

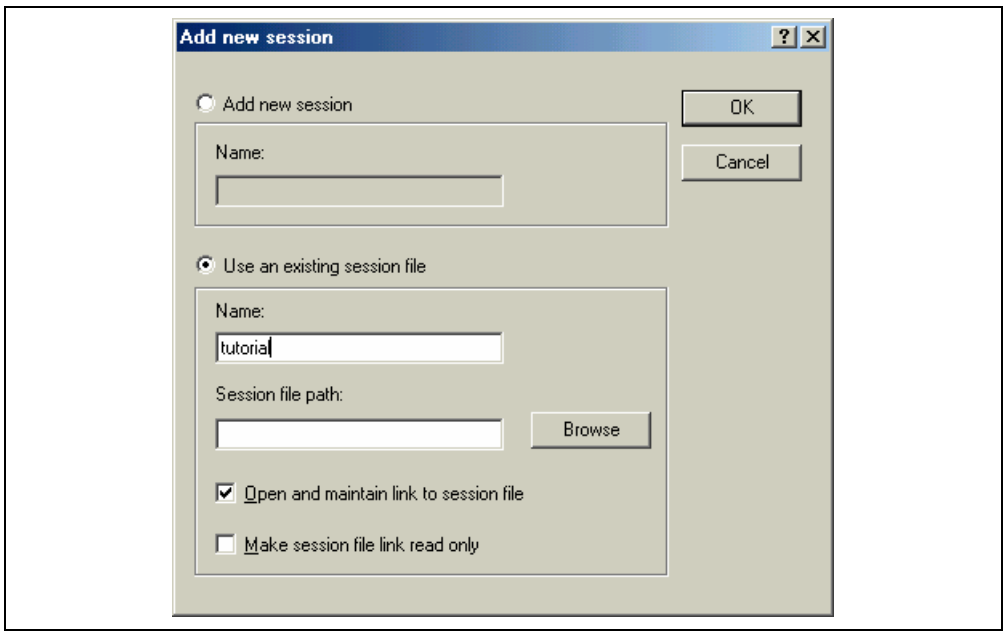


Figure 3.20 Add new session Dialog Box

- To import an existing session into a new session file
  1. Select [Options -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 3.19).
  2. Click the [Add…] button to display the [Add new session] dialog box (figure 3.20).
  3. Check the [Use an existing session file] radio button.
  4. Enter a name for the session.
  5. Browse to the existing session file location that you would like to import into the current project.
     If the [Open and maintain link to session file] check box is not checked, the imported new session file is generated in the project directory.
     If the [Open and maintain link to session file] check box is checked, a new session file is not generated in the project directory but is linked to the current session file.
     If the [Make session file link read only] check box is checked, the linked session file is used as read-only.
  6. Click the [OK] button to close the [Debug Sessions] dialog box.

- To remove a session
  1. Select [Options -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 3.19).
  2. Select the session you would like to remove.
  3. Click the [Remove] button.
     Note that the current session cannot be removed.
  4. Click the [OK] button to close the [Debug Sessions] dialog box.

- To view the session properties
  1. Select [Options -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 3.19).
  2. Select the session you would like to view the properties for.
  3. Click the [Properties] button to display the [Session Properties] dialog box (figure 3.21).

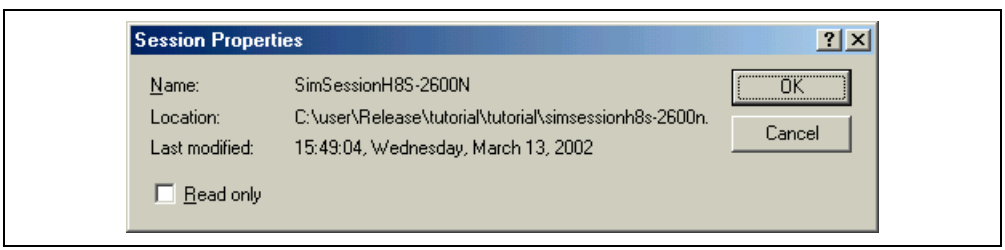


Figure 3.21 Session Properties Dialog Box

- To make a session read-only
  1. Select [Options -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 3.19).
  2. Select the session you would like to make read-only.
  3. Click the [Properties] button to display the [Session Properties] dialog box (figure 3.21).
  4. Check the [Read only] check box to make the link read-only. This is useful if you are sharing debugger-setting files and you do not want data to be modified accidentally.
  5. Click the [OK] button.

- To save a session with a different name
  1. Select [Options -> Debug Sessions…] to display the [Debug Sessions] dialog box (figure 3.19).
  2. Select the session you would like to save.
  3. Click the [Save as] button to display the [Save Session] dialog box (figure 3.22).
  4. Browse to the new file location.
  5. If you want to export the session file to another location, leave the [Maintain link] check box unchecked. If you would like the HEW to use this location instead of the current session location, check the [Maintain link] check box.
  6. Click the [OK] button.

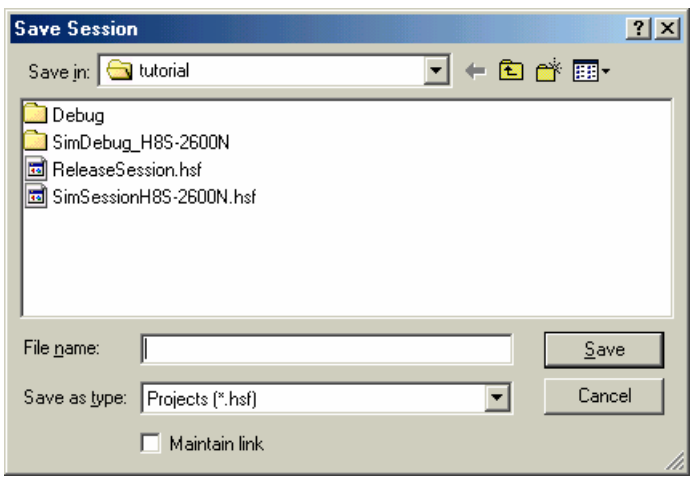


Figure 3.22 Save Session Dialog Box

3.4.3 Saving Session Information

- To save a session
  Select [File -> Save Session]. A standard file save dialog is displayed.

Note: You can display the dialog box that asks you whether or not the session is saved.. Clicking [No] loses the changes you made in the session. This dialog box is displayed by checking [Confirm before saving a session] on the [Workspace] tab in the [Option] dialog box selected from the [Tools->Options] menu.

3.4.4 Reloading Session Information

- To reload a session
  Select [File -> Reload Session]. Clicking this will lose any changes to your session currently and the reload the current session into HEW. A confirmation will be displayed before the reload happens.

3.4.5 Debugging Multiple Targets

For the method to debug multiple targets in synchronization, refer to section 4.23, Synchronizing Multiple Debugging Platforms.

Section 4 Debugging

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

4.1 Viewing a Program

How to look at your program as source code and assembly language mnemonics is explained here. 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 via the user interface.

Note: After a break occurs, the HEW displays the location of the program counter (PC) on the editor. In most cases, for example if an Elf/Dwarf2 based project is moved from its original path, the source file may not be automatically found. In this case, the HEW will open a source file browser dialog box to allow you to manually locate the file. This path will then be used to update any other source files in this debug project.

4.1.1 Viewing the Source Code

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

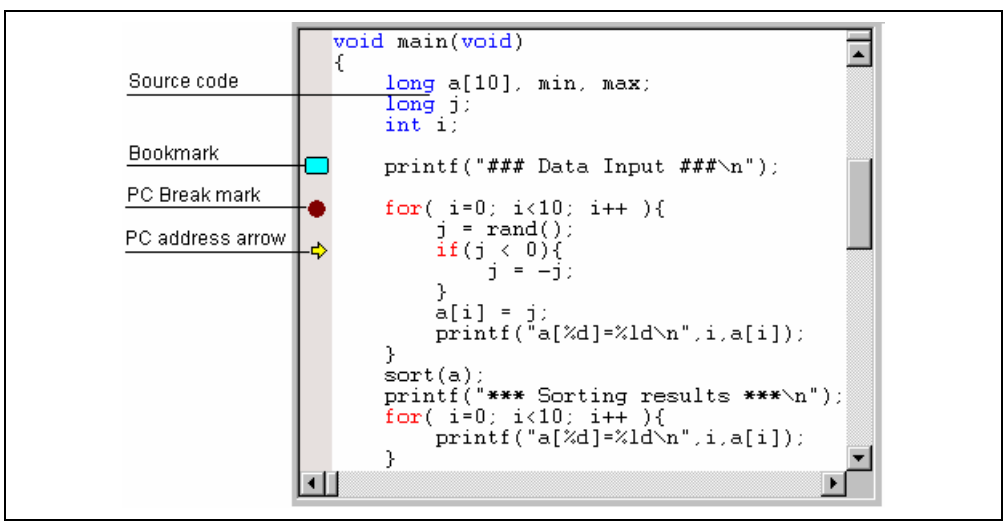


Figure 4.1 Editor Window

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 4.1.

4.1.2 Source Address Column

After your program has been downloaded, the [Editor] window displays the addresses for the current source file (figure 4.2). The addresses are shown on the left-hand side of the [Editor] window. This is useful for deciding where to set the PC or hardware breakpoints.

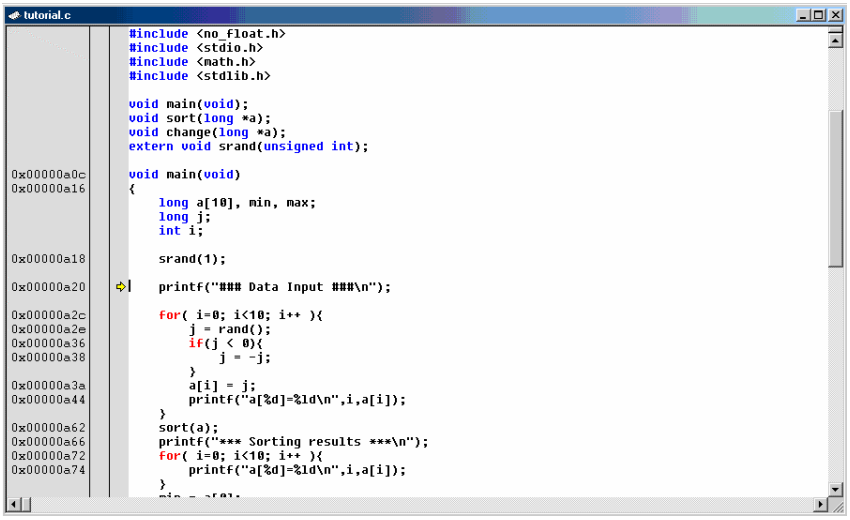


Figure 4.2 Editor Window and Address Column

➲ To switch off a column in all source files

1. Right-click on the [Editor] window or select the [Edit] menu.
2. Click the [Define Column Format…] menu item.
3. The [Global Editor Column States] dialog box is displayed.
4. The check box indicates whether the column is enabled or not. If it is checked, the column is enabled. If the check box is gray, the column is enabled in some files and disabled in others.
5. Click the [OK] button for the new column settings to take effect.

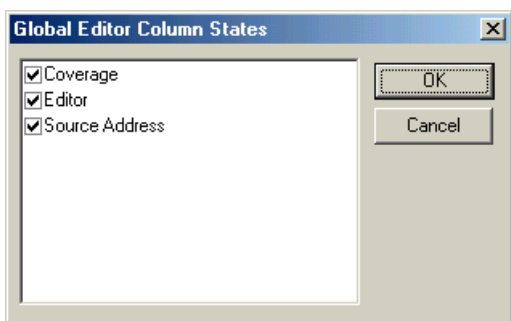


Figure 4.3 Global Editor Column States Dialog Box

➲ To switch off a column in one source file

1. Right-click on the [Editor] window which contains the column you wish to remove to display the pop-up menu.
2. Click the [Columns] menu item to display a cascaded menu item. Each column is displayed in this pop-up menu. If the column is enabled, it has a tick mark next to its name. Clicking the entry will toggle whether the column is displayed or not.

4.1.3 Debugger Columns

The debugging platform can add columns to the [Editor] window, which is called as the debugger column. These added columns are referred to as debugger columns.

The columns that can be added to the [Source] window differ according to the debugging platform. The [Coverage] column that graphically displays code coverage during debugger execution is an example of the columns that can be added.

Right-clicking on the column displays the respective pop-up menu for that column. Double-clicking on the column has a different effect depending on the column. For example, in the [Editor] column, this sets a PC breakpoint.

If you are unsure what purpose a column has or what the information it is displaying, place the cursor over the column, and a tool tip is displayed.

4.1.4 Viewing the Assembly-Language Code

If you have a source file open, right-click to open the pop-up menu and select [Go to Disassembly] to open the [Disassembly] window at the same address as the current [Editor] window.

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

The [Disassembly] window opens at the current PC location and shows [Address] and [Code] (optional) which show the disassembled mnemonics (with labels when available).

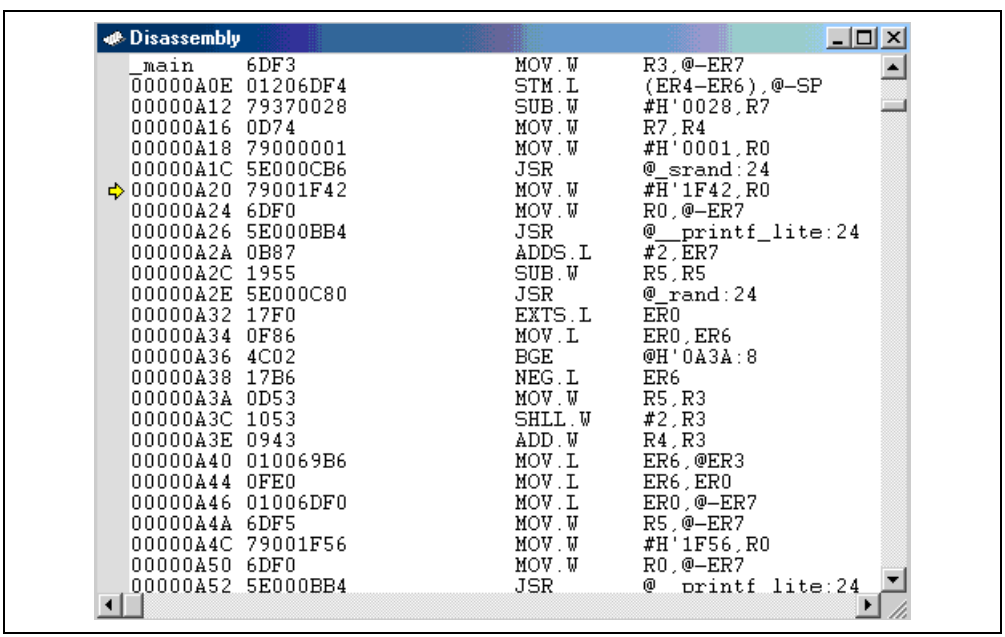


Figure 4.4 Disassembly Window

4.1.5 Modifying the 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.

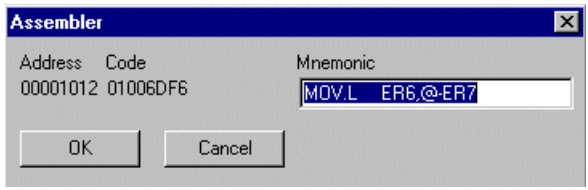


Figure 4.5 Assembler Dialog Box

The address, machine code, and disassembled instruction are displayed. Enter 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 the [OK] button will assemble the instruction into memory and close the dialog box. Clicking the [Cancel] button or pressing the Esc key will close the dialog box.

Note: The assembly-language display is disassembled from the actual machine code in the debugging platform's memory. If the memory contents are changed the dialog box (and [Disassembly] window) will show the new assembly-language code, but the source file displayed in the [Editor] window will be unchanged. This is the same even if the source file contains an assembler.

4.1.6 Viewing a Specific Address

When you are viewing your program in the [Disassembly] window, you may wish to look at another area of your program's code. Rather than scrolling through a lot of code in the program, you can go directly to a specific address. Select [Set Address] from the pop-up menu, and the dialog box shown in figure 4.6 is displayed.

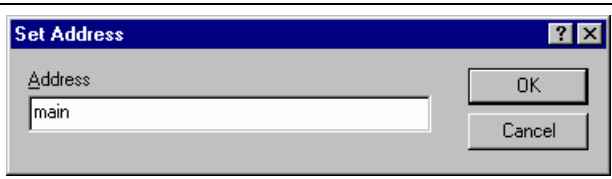


Figure 4.6 Set Address Dialog Box

Enter the address or label name in the edit box and either click on the [OK] button or press the Enter key. The [Disassembly] window will be updated to show the code at the new address. When an overloaded function or a class name is entered, the [Select Function] dialog box opens for you to select a function. For details, refer to section 4.10, Elf/Dwarf2 Support.

4.1.7 Viewing the Current Program Counter Address

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

4.2 Operating Memory

This section describes how to look at memory areas in the CPU's address space. How to look at a memory area in different formats, how to fill and move a memory block, and how to load and verify a memory area with a disk file are described.

4.2.1 Viewing a Memory Area

To look at a memory area, choose [View -> CPU ->Memory…] with the Ctrl + M accelerator or click the [View Memory] toolbar button [] to open the [Memory] window. This will open the [Format] dialog box shown in figure 4.7.

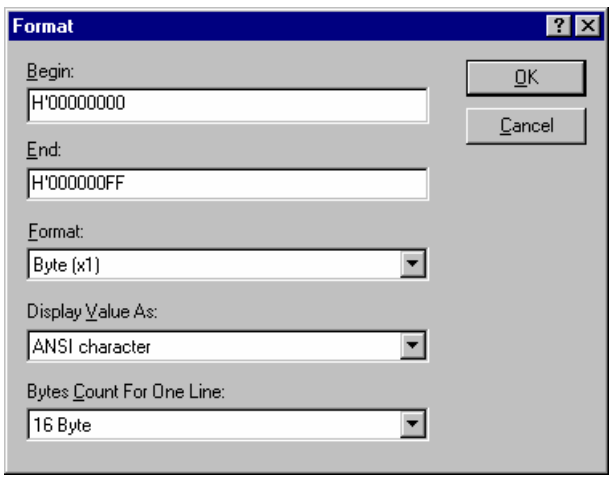


Figure 4.7 Format Dialog Box

Enter the range you wish to display as an address value or an equivalent symbol in the [Begin] and [End] fields. Select the data size and format for the display from the [Display Value As] and [Format] drop-down lists, respectively. Select the number of bytes displayed in one line from the [Bytes Count For One Line] drop-down list. Click the [OK] button or press the Enter key, and the dialog box closes and the [Memory] window opens. The display can be scrolled within the range of the entered display start and end addresses.

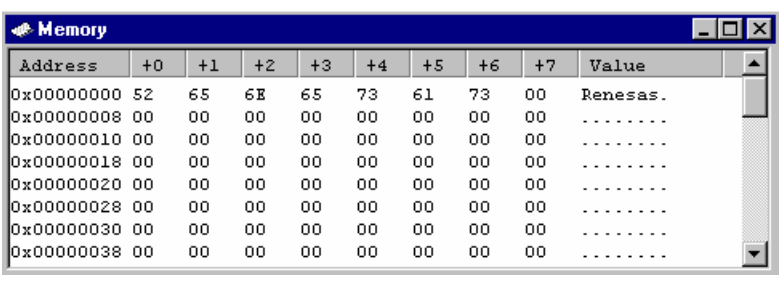


Figure 4.8 Memory Window

There are three display columns:

[Address] First address of the memory data displayed on this row

[+n] Memory data read from [Address] and 'n' means the offset value from the first address of the row. Data is read from the debugging platform's physical memory in the access width, and then converted to the display width.

[Value] Data displayed in an alternative format

4.2.2 Displaying Data in Different Formats

If you want to change the display format of the [Memory] window, select [Format] from the pop-up menu. The dialog box shown in figure 4.9 is displayed.

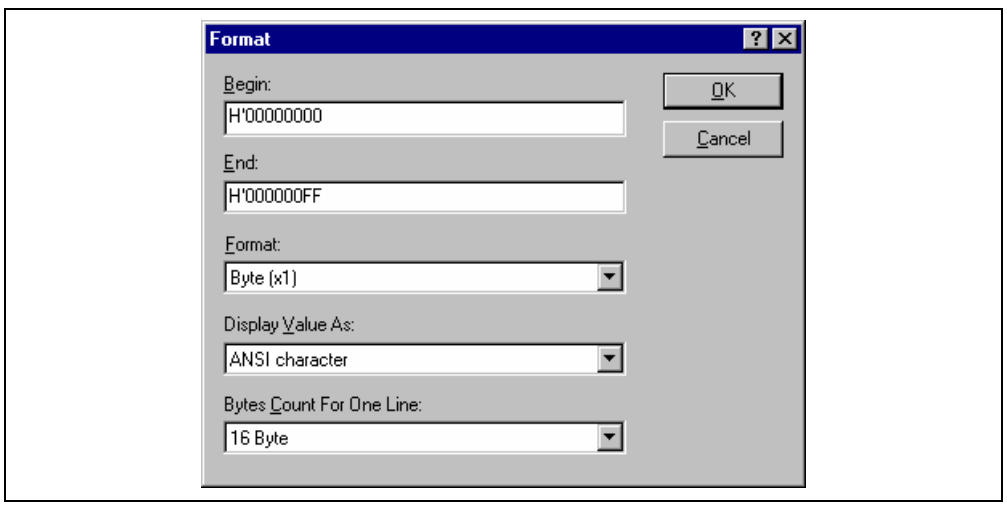


Figure 4.9 Format Memory Display Dialog Box

To display and edit memory in different widths, use the [Format] drop-down list. For example, choose the [Byte] option and the display will be updated to show the memory area as individual bytes.

The data can be converted into different formats, as shown in the third column [Value]. The list of formats depends on the data selection.

To change the number of bytes displayed on one line, use the [Bytes Count For One Line] drop-down list. For example, choose the [8 Byte] option and the display will be updated to show the 8-byte data on one line.

4.2.3 Splitting Up the Window Display

To vertically divide the [Memory] window display into two, select [Split] from the pop-up menu and move the split-up bar. Moving the split-up bar to the top end or bottom end of the window cancels the split-up display.

4.2.4 Viewing a Different Memory Area

To change the memory area displayed in the [Memory] window, use the scroll bars. To quickly look at a new address, use the [Format] dialog box. This can be opened by choosing [Format] from the pop-up menu.

Enter the new address value, and click the [OK] button or press the Enter key. The dialog box closes and the [Memory] window display is updated with the data at the new address. When an overloaded function or a class name is entered, the [Select Function] dialog box opens for you to select a function.

4.2.5 Modifying the Memory Contents

The memory contents can be modified via the [Edit Memory] dialog box. Move the cursor on the memory unit (according to the [Memory] window display choice) that you wish to change. Either double-click on the memory unit or press the Enter key. The dialog box shown in figure 4.10 is displayed.

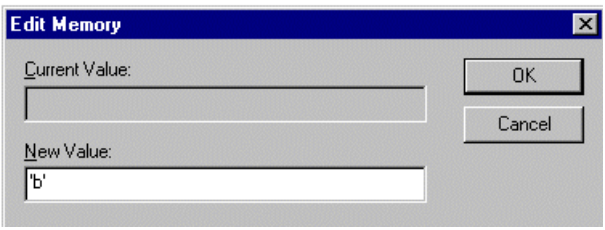


Figure 4.10 Edit Memory Dialog Box

A number or C/C++ expression can be entered in the [New Value] field. After you have entered the new number or expression, click the [OK] button or press the Enter key. Then the dialog box closes and the new value is written into memory.

The memory contents can also be modified by moving the cursor on the memory unit and entering the new value in hexadecimal through the keyboard.

4.2.6 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 (according to the [Memory] window display choice) and dragging the mouse to the last unit. The selected range is highlighted.

If the memory address range is larger than or outside the [Memory] window, you can enter the start address and byte count in the respective fields of the [Format] dialog box.

4.2.7 Finding a Value in Memory

To find a value in memory, open the [Memory] window and select [Search] from the pop-up menu. Otherwise, when the [Memory] window has already been opened, simply press the [F3] key.

The [Search Memory] dialog box shown in figure 4.11 is displayed.

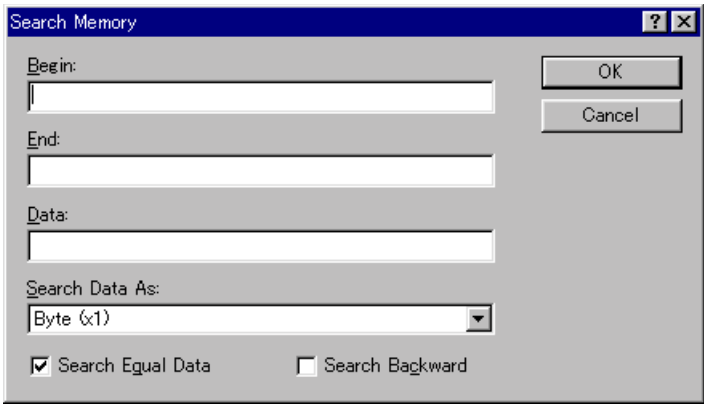


Figure 4.11 Search Memory Dialog Box

Enter the start and end addresses of the range in which to search (the start and end address of the [Memory] will be automatically filled in) and the data value to search for, and select the search format. If pattern search is selected as the search format, a byte string of up to 256 bytes can be searched for. The end address can also be prefixed by a plus (+); the end address will become the (start address) + (entered value).

Search conditions other than pattern search are data match/mismatch and search direction. Note that only data match and forward direction can be selected with pattern search.

Click the [OK] button or press the Enter key. The dialog box closes and the HEW searches the range for the specified data. If the data is found, the address at which the data has been found is displayed in the [Memory] window.

If the data could not be found, the [Memory] window display remains unchanged and a message box informing that the data could not be found is displayed.

If [Search Next] is selected from the pop-up menu in the state where data has been found, the search will continue from the next address.

4.2.8 Filling a Memory Area with a Value

A value can be set as the contents of a memory address range using the memory fill function.

To fill the same value in a memory range, choose [Fill] from the pop-up menu of the [Memory] window or choose [Fill] from the [Memory] drop-down menu. The [Fill Memory] dialog box is shown in figure 4.12.

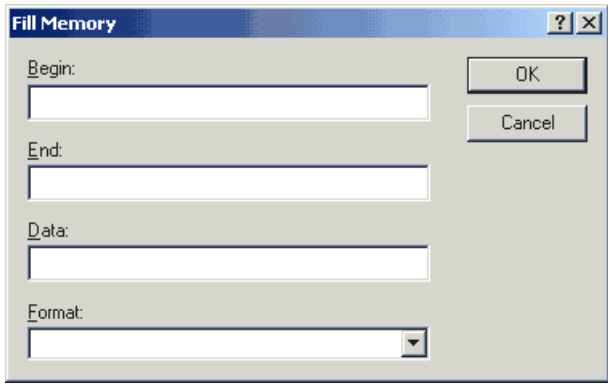


Figure 4.12 Fill Memory Dialog Box

If an address range has been selected in the [Memory] window, the specified start and end addresses will be displayed. Select a format from the [Format] drop-down list and enter the data value in the [Data] field. It is also possible to fill the memory by checking the [Verify] check box and comparing the data value to be filled and the filled memory data. On clicking the [OK] button or pressing the Enter key, the dialog box closes and the new value is written into the memory range.

4.2.9 Copying a Memory Area

You can copy a memory area using the memory copy function. Select a memory range and then choose [Copy…] from the pop-up menu. The [Copy Memory] dialog box opens (figure 4.13).

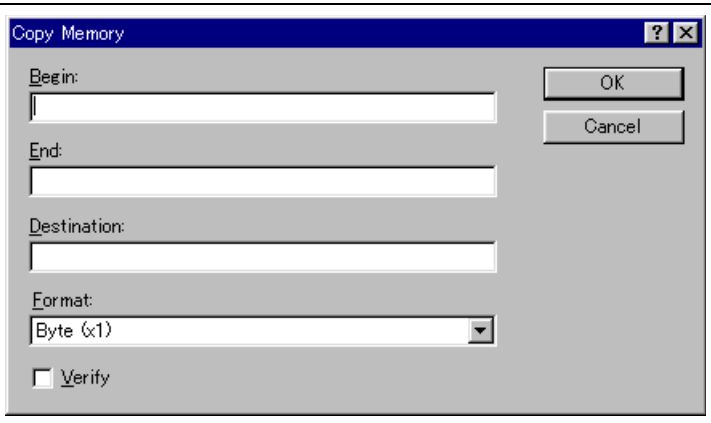


Figure 4.13 Copy Memory Dialog Box

The source start address and end address selected in the [Memory] window will be displayed in the [Begin] and [End] fields. Checking the [Verify] check box enables copying while comparing the copy source and copy destination. The copy unit can be selected in the [Format] list box. Enter the destination start address in the [Destination] field and click the [OK] button or press the Enter key. This will close the dialog box and copy the memory block to the new address.

4.2.10 Saving and Verifying a Memory Area

A memory area in the address space can be saved into a disk file using the memory save function. Open the [Save Memory As] dialog box by choosing [File -> Save memory...].

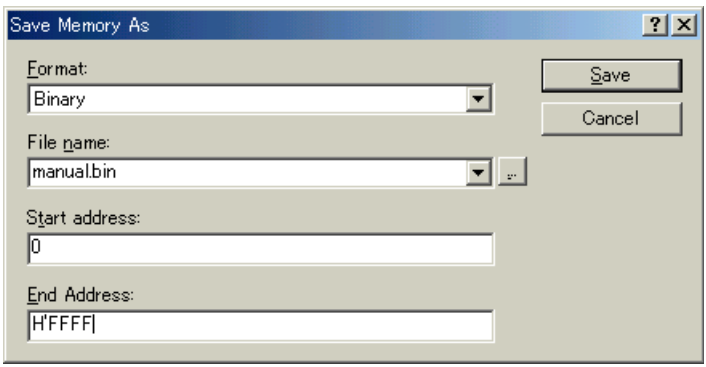


Figure 4.14 Save Memory As Dialog Box

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-down list contains the previous four file names used for saving memory.

Clicking the [Browse…] button can open the standard [File Save As] dialog box. On clicking the [OK] button or pressing the Enter key, the dialog box closes and the memory block will be saved into the disk as a file of the specified format type. When file saving has completed, a confirmation message box may be displayed.

A memory area in the address space can be verified using the memory verify function. Open the [Verify Memory] dialog box by choosing [File -> Verify Memory...].

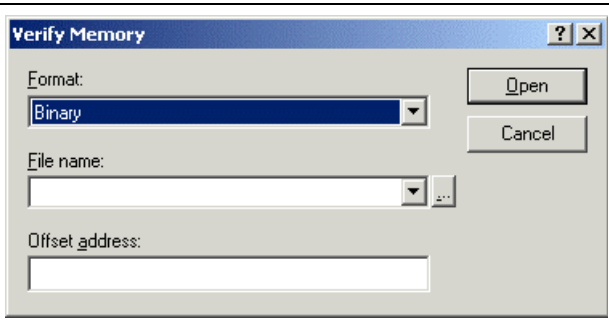


Figure 4.15 Verify Memory Dialog Box

4.2.11 Disabling Update of the Window Contents

Automatic update of the [Memory] window contents, which is performed when user program execution stops and in other cases, can be disabled. This is done by checking [Lock Refresh] in the pop-up menu.

4.2.12 Updating the Window Contents

The [Memory] window contents can be forcibly updated. This is done by checking [Refresh] in the pop-up menu.

4.2.13 Comparing the Memory Contents

The contents of two memory blocks can be compared. Open the [Compare Memory] dialog box by selecting [Memory -> Compare…] from the main menu or by selecting [Compare…] from the pop-up menu of the [Memory] window.

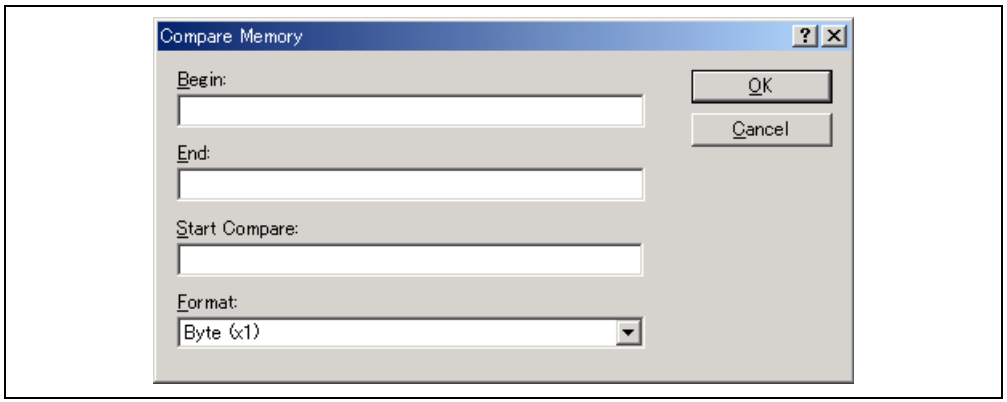


Figure 4.16 Compare Memory Dialog Box

Enter the comparison format ([Format]), the start address ([Begin]) and end address ([End]) of the source memory area, and the start address ([Start Compare]) of the destination memory area. If the memory block is already highlighted in the [Memory] window, the start and end addresses will be automatically filled in when the [Compare Memory] dialog box is opened.

If there is a mismatch, the address where it was found is displayed in a message box.

4.2.14 Loading a Memory Area from a File

A file can be loaded to the debugging platform's memory. Select [Load…] from the pop-up menu of the [Memory] window to open the [Load Program] dialog box.

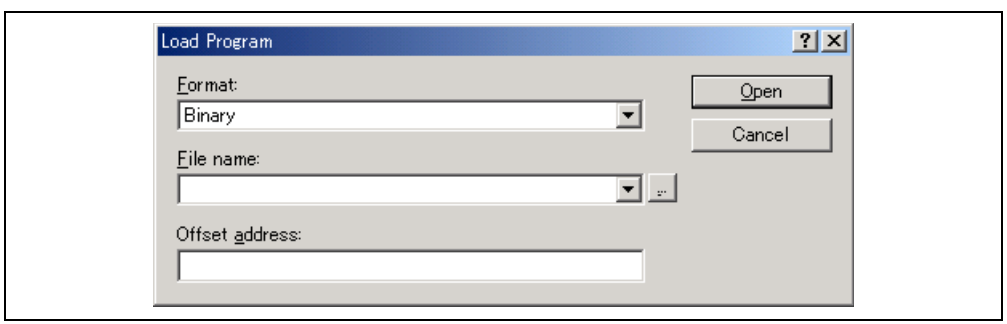


Figure 4.17 Load Program Dialog Box

Enter the file format ([Format]) and the file name ([File name]). If the load address value is to be changed, enter the offset value in the offset field ([Offset address]), otherwise enter zero.

4.3 Displaying Memory Contents as an Image

The memory contents can be displayed as an image in the [Image View] window.

4.3.1 Opening the Image View Window

Choose [View -> Graphic -> Image…] or click the [Image] toolbar button [ ] to open the [Image Properties] dialog box shown in figure 4.18.

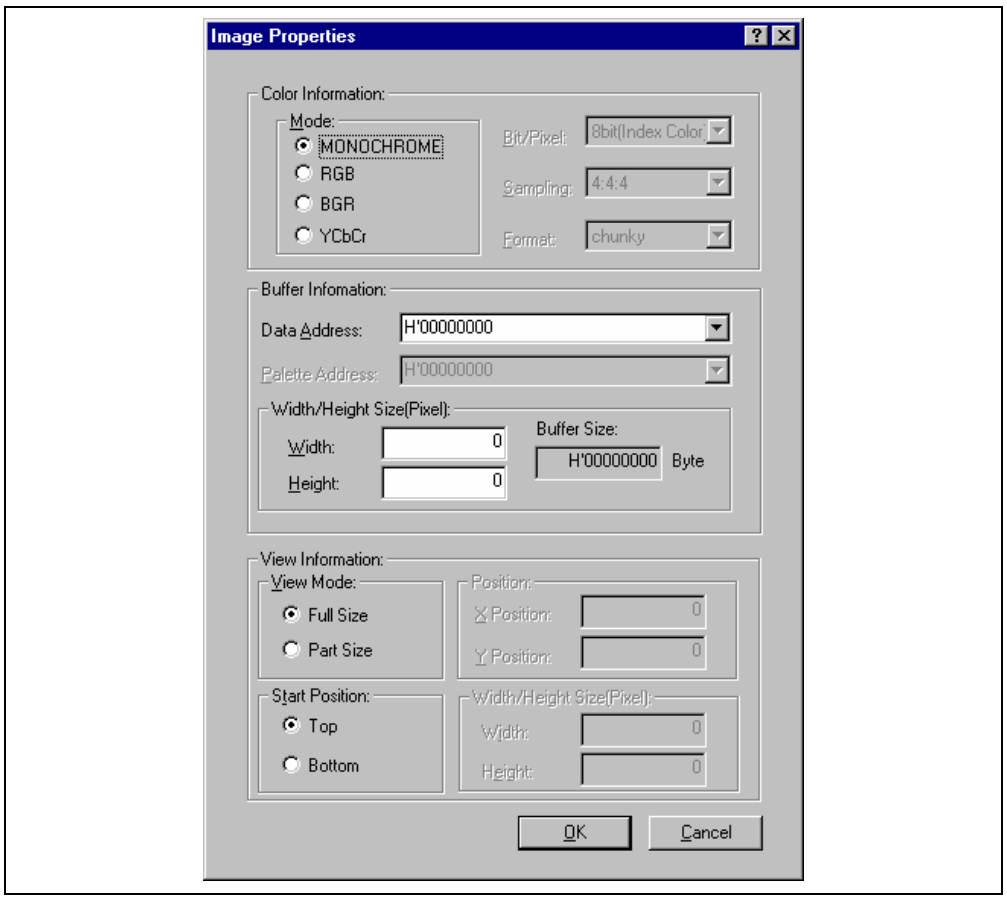


Figure 4.18 Image Properties Dialog Box

The [Image Properties] dialog box is used to specify the display method of the [Image View] window.

The following items are to be specified:

[Color Information] Specifies the color information of the image to be displayed.

[Mode] Specifies the format.

[MONOCHROME]	Displays in black and white.
[RGB]	Displayed in R (red), G (green), and B (blue)

[BGR] Displayed in B (blue), G (green), and R (red)

[YCbCr] Displayed by Y (brightness), Cb (color difference in blue), and Cr (color difference in red)

[Bit/Pixel] Specifies Bit/Pixel according to the selected [Mode]. (Valid when RGB or BGR is selected)

[Sampling] Specifies the format of sampling. (Valid when YCbCr is selected)

[Format] Specifies Chunky/planar. (Valid when YCbCr is selected)

[Buffer Information] Specifies the area to store data, size, and the address of the palette.

[Data Address] Specifies the start address of the memory where image data is to be displayed. (Displayed in hexadecimal)

[Palette Address] Specifies the start address of the memory of color palette data. (Displayed in hexadecimal) (Valid when 8Bit is selected for RGB or BGR)

[Width/Height Size] Specifies the width and height of the image.

[Width (Pixel)] Specifies the width of the image. (When a prefix is omitted, the values are input and displayed in decimal.)

[Height (Pixel)] Specifies the height of the image. (When a prefix is omitted, the values are input and displayed in decimal.)

[Buffer Size] Displays the buffer size of the image from the width and height (Displayed in hexadecimal)

[View Information] Specifies the location, size, and data start location of the part to be displayed among the entire image.

[View Mode] Specifies the entire/part to be displayed in the image.

[Full Size] Displays the entire image.

[Part Size] Displays part of the image.

[Start Position]

[Top] Displays data from the upper left.

[Bottom] Displays data from the lower left.

[Position] Specifies the start position of the image where part of the image is to be displayed. (Valid when [Part Size] is selected)

[X Position] Specifies the X axis of the start location. (When a prefix is omitted, the values are input and displayed in decimal.)

[Y Position] Specifies the Y axis of the start location. (When a prefix is omitted, the values are input and displayed in decimal.)

[Width/Height Size] Specifies the height and width of the image to be displayed partly.

[Width (Pixel)] Displays the width of display. (When a prefix is omitted, the values are input and displayed in decimal.)

[Height (Pixel)]	Displays the height of display. (When a prefix is omitted, the values are input and displayed in decimal.)

After the settings have been made in the [Image Properties] dialog box, clicking the [OK] button opens the [Image View] window.

Even after the [Image View] window is displayed, the display contents can be modified by opening this dialog box by choosing [Properties…] from the pop-up menu.

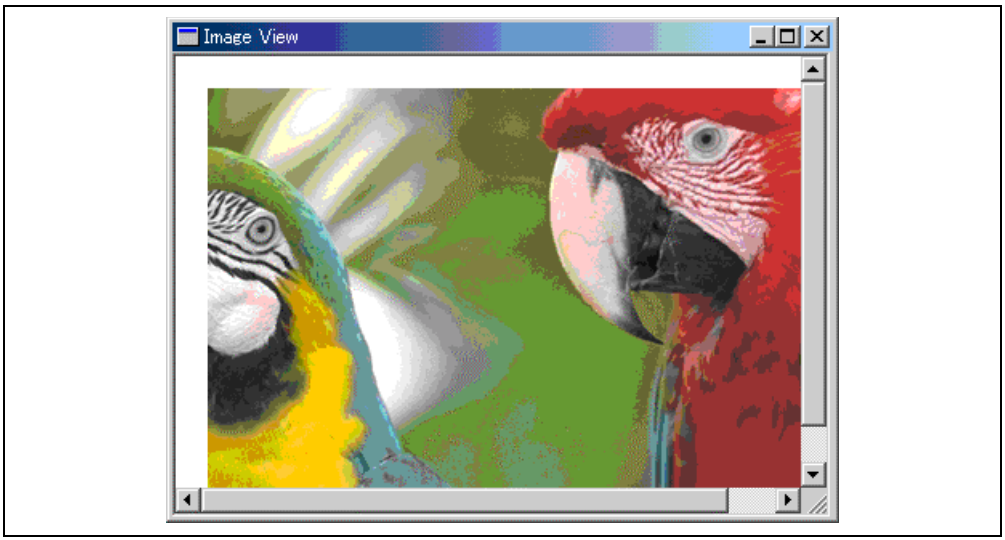


Figure 4.19 Image View Window

Displays the memory contents as an image.

4.3.2 Automatically Updating the Window Contents

Checking [Auto Refresh -> Not Refresh] in the pop-up menu will not refresh the window.

Checking [Auto Refresh -> Stop] in the pop-up menu will allow the window contents to be automatically updated when user program execution stops.

Checking [Auto Refresh -> Realtime] in the pop-up menu will allow the window contents to be automatically updated in an instruction interval specified with the [Response] command.

4.3.3 Updating the Window Contents

Selecting [Refresh Now] from the pop-up menu immediately updates the window contents.

4.3.4 Displaying the Pixel Information

Double-clicking within the window displays information on the pixel on which the mouse pointer is located in the [Pixel Information] dialog box.

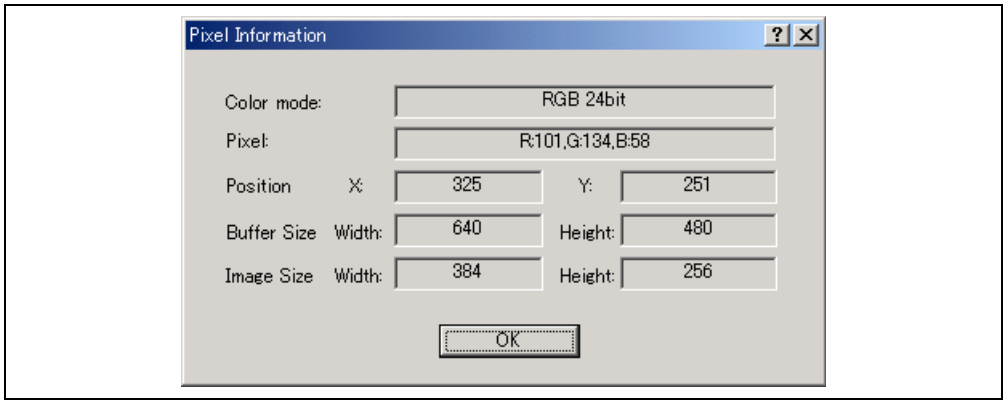


Figure 4.20 Pixel Information Dialog Box

This dialog box displays pixel information on the cursor location.

[Color Mode] Displays the format of the image.

[Pixel] Displays color information of the cursor location. (Displayed in decimal)

[Position] Displays the cursor location in X and Y axis. (Displayed in decimal)
[X] Displays the X axis of the cursor location.
[Y] Displays the Y axis of the cursor location.

[Buffer Size] Displays the buffer size. (Displayed in decimal)
[Width] Displays the buffer width.
[Height] Displays the buffer height.

[Image Size] Displays the width and height of the display. (Displayed in decimal)
[Width] Displays the width.
[Height] Displays the height.

4.4 Displaying Memory Contents as Waveforms

Memory contents can be displayed as waveforms in the [Waveform View] window.

4.4.1 Opening the Waveform View Window

Choose [View -> Graphic -> Waveform…] or click the [Waveform] toolbar button [ ] to open the [Waveform Properties] dialog box shown in figure 4.21.

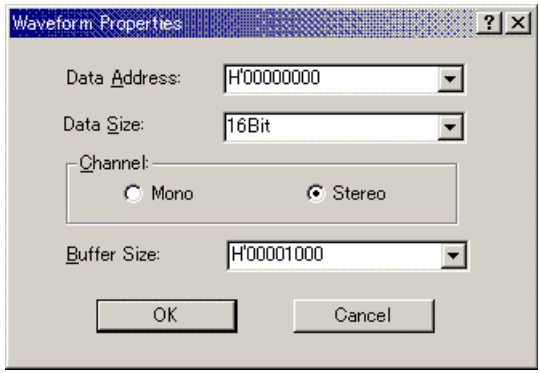


Figure 4.21 Waveform Properties Dialog Box

Specifies the waveform format. The following items can be specified.

[Data Address] Specifies the start address of data in memory. (Displayed in hexadecimal)

[Data Size] Selects 8Bit or 16Bit.

[Channel] Specifies Mono or Stereo.

[Buffer Size] Specifies the buffer size of data. (Displayed in hexadecimal)

After the settings have been made in the [Waveform Properties] dialog box, clicking the [OK] button opens the [Waveform View] window.

Even after the [Waveform View] window is displayed, the display contents can be modified by opening this dialog box by choosing [Properties…] from the pop-up menu.

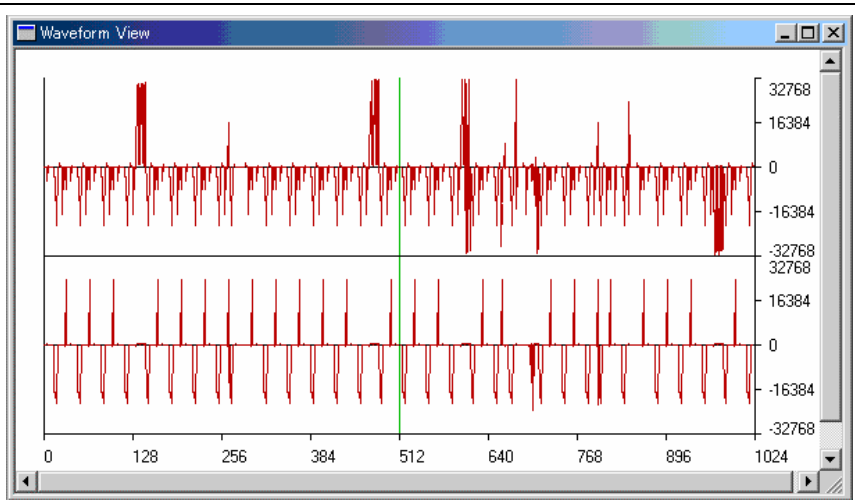


Figure 4.22 Waveform View Window

Displays the memory contents as waveforms. The X axis shows the number of sampling data and the Y axis shows the sampling value.

4.4.2 Automatically Updating the Window Contents

Checking [Auto Refresh -> Not Refresh] in the pop-up menu will not refresh the window.

Checking [Auto Refresh -> Stop] in the pop-up menu will allow the window contents to be automatically updated when user program execution stops.

Checking [Auto Refresh -> Realtime] in the pop-up menu will allow the window contents to be automatically updated in an instruction interval specified with the [Response] command.

4.4.3 Updating the Window Contents

Selecting [Refresh Now] from the pop-up menu immediately updates the window contents.

4.4.4 Zoom-In Display

Selecting [Zoom In] from the pop-up menu displays the waveforms with the horizontal axis enlarged.

4.4.5 Zoom-Out Display

Selecting [Zoom Out] from the pop-up menu displays the waveforms with the horizontal axis reduced.

4.4.6 Resetting the Zoom Display

Selecting [Reset Zoom] from the pop-up menu displays the waveforms in its original size.

4.4.7 Setting the Zoom Magnification

In the [Zoom Magnification] submenu of the pop-up menu, the zoom magnification can be selected from 2, 4, or 8.

4.4.8 Setting the Horizontal Scale

In the [XScale] submenu of the pop-up menu, the size of the X axis can be selected from 128, 256, or 512 pixels.

4.4.9 Non-Display of Cursor

Selecting [Clear Cursor] from the pop-up menu hides the cursor display.

4.4.10 Displaying the Sampling Information

Selecting [Sample Information…] from the pop-up menu displays the [Sample Information] dialog box.

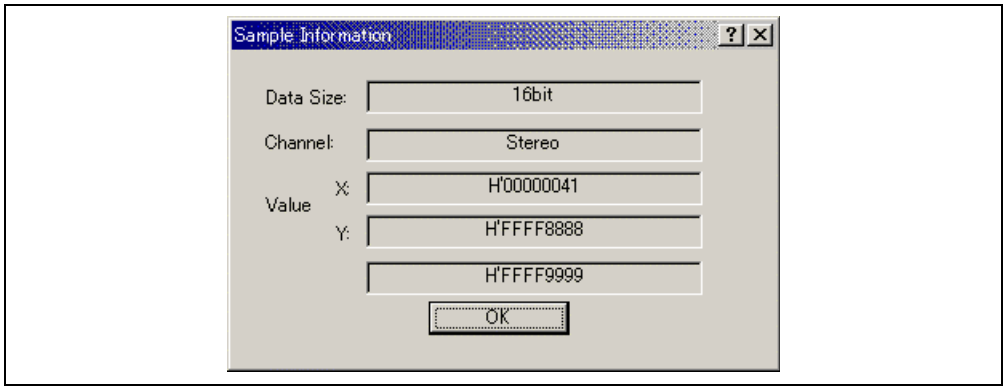


Figure 4.23 Sample Information Dialog Box

Displays the sampling information of the cursor location in the [Waveform View] window. The following information is displayed.

[Data Size] Displays 8bit or 16bit.

[Channel] Displays the data channel.

[Value] [X] Displays the X axis of cursor location.
[Y] Displays the Y axis of cursor location (displays Y axes for both the upper and lower plots when Stereo is selected).

4.5 Viewing the I/O Register

As well as a CPU and ROM/RAM, a microcomputer also contains on-chip peripheral modules. The exact number and type of peripheral modules differ between devices but the typical modules are a DMA controller, serial communications interface, A/D converter, integrated timer unit, bus state controller, and watchdog timer. Registers that are mapped to the microcomputer's address space controls the on-chip peripheral modules.

The [Memory] window enables you to look at data in continuous memory addresses as byte, word, longword, single-precision floating-point, double-precision floating-point, or ASCII values. However, registers of different sizes are allocated to non-continuous memory addresses in the I/O register. To handle this register, the HEW has the [IO] window to facilitate checking and setting up of these kinds of registers.

4.5.1 Opening the IO Window

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

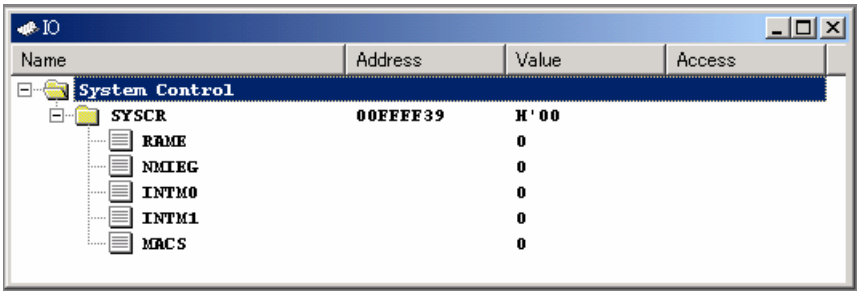


Figure 4.24 IO Window

4.5.2 Expanding the I/O Register Display

To display the names, addresses, and values of the I/O registers, double-click on the module name or select the module name by clicking on it or using the cursor keys and press the Enter key. The module display will expand to show the individual registers of that peripheral module and their names, addresses, and values. Double-clicking (or pressing the Enter key) again on the module name will close the I/O register display.

For a display in the bit level, expand the I/O register in a similar way to the [Registers] window.

4.5.3 Manually loading an IO file

To manually load an IO file right click on the IO window and select the "Load IO file…" menu item on the IO window pop-up. A standard file browser is invoked. Simply select the file you require to load and click OK. The IO file will be loaded into the window.

4.5.4 Modifying the I/O Register Contents

To edit the value in an I/O register, type hexadecimal values directly into the window. To enter more complex expressions, double-click or press the Enter key on the register to open a dialog box to modify the register contents. When you have entered the new number or expression, click the [OK] button or press the Enter key; the dialog box closes and the new value is written into the register.

4.6 Looking at Labels

Symbol information is included in the debugging information, which is used when the HEW links the user program source code to the actual code in the memory. Symbol information is also included in the debug object file. This information is a list of names that indicate addresses in the program. These names are called labels in the HEW. The [Disassembly] window shows the first eight characters of each label instead of the corresponding address or as a part of an instruction operand*.

Note: When a label value matches an operand, the corresponding instruction operand is replaced by the label. If two or more labels have the same value, the one that comes first in alphabetical order is displayed.

Hint: When [edit control] accepts an address or a value, a label can be input instead.

4.6.1 Listing Labels

Choose [View -> Symbol -> Labels] or click the [View Labels] toolbar button to list all labels defined in the current debugger session.

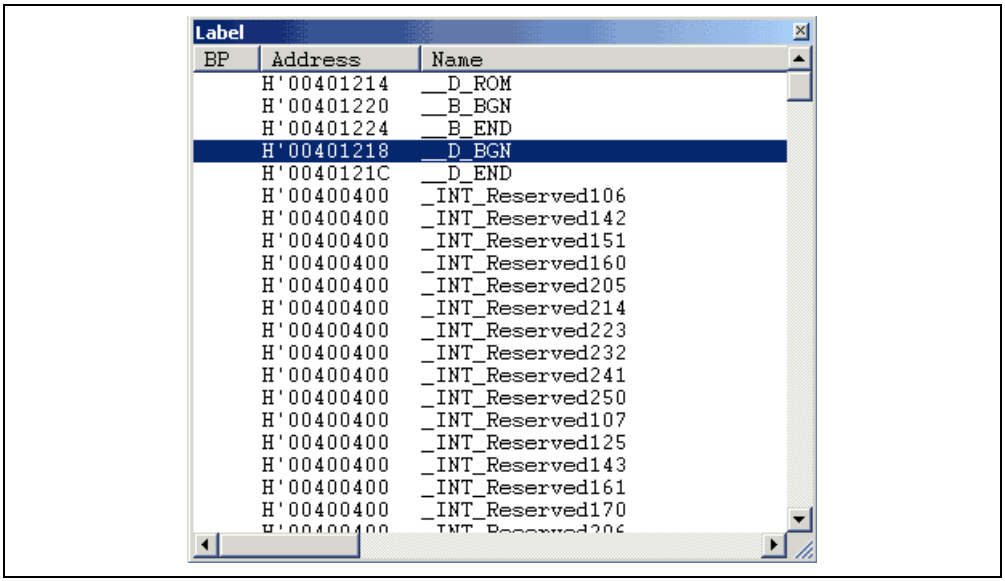


Figure 4.25 Label Window

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

Double-clicking in the [BP] column can set or clear a software breakpoints at the start of the function.

4.6.2 Adding a Label

Choose [Add..] from the pop-up menu and open the [Add Label] dialog box to add a label:

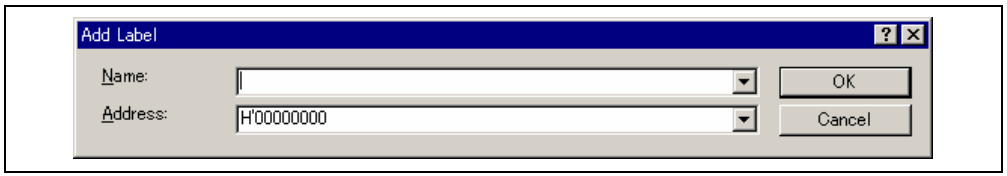


Figure 4.26 Add Label Dialog Box

Enter the new label name into the [Name] field and the corresponding value into the [Address] field and press [OK]. The [Add Label] dialog box closes and the label list is updated to show the new label. When an overloaded function or a class name is entered in the [Address] field, the [Select Function] dialog box opens for you to select a function. For details, refer to section 4.10.3, Supporting Duplicate Labels.

4.6.3 Editing a Label

Choose [Edit...] from the pop-up menu and open the [Edit Label] dialog box to edit a label:

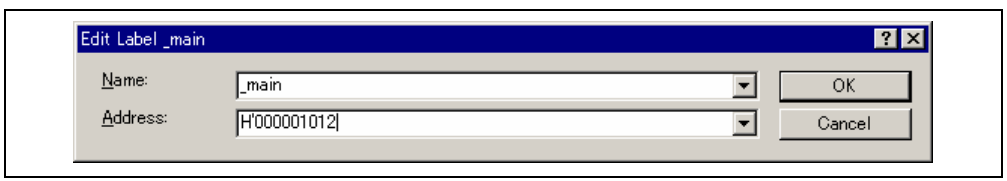


Figure 4.27 Edit Label Dialog Box

Edit the label name and value as required and then press [OK] to save the modified version in the label list. The list display is updated to show the new label details. When an overloaded function or a class name is entered in the [Address] field, the [Select Function] dialog box opens for you to select a function. For details, refer to section 4.10.3, Supporting Duplicate Labels.

4.6.4 Deleting a Label

To delete a label, select the label and choose [Delete] from the pop-up menu. A confirmation message box appears:

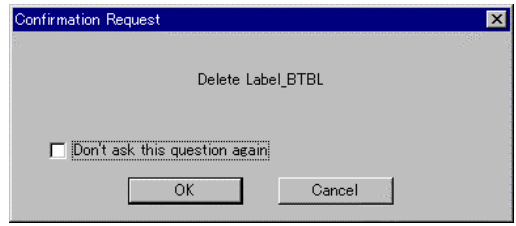


Figure 4.28 Message Box for Confirming Label Deletion

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

4.6.5 Deleting All Labels

To delete all the labels from the list, choose [Delete All] from the pop-up menu. A confirmation message box appears:

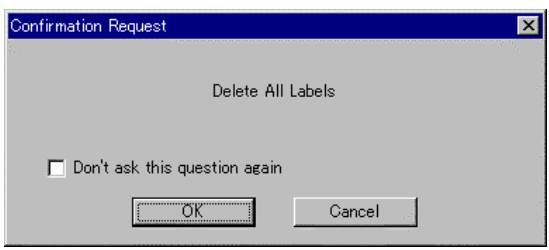


Figure 4.29 Message Box for Confirming All Label Deletion

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

4.6.6 Loading Labels from a File

A symbol file can be loaded and merged into the HEW's current symbol table. Choose [Load...] from the pop-up menu to open the [Load Symbols] dialog box:

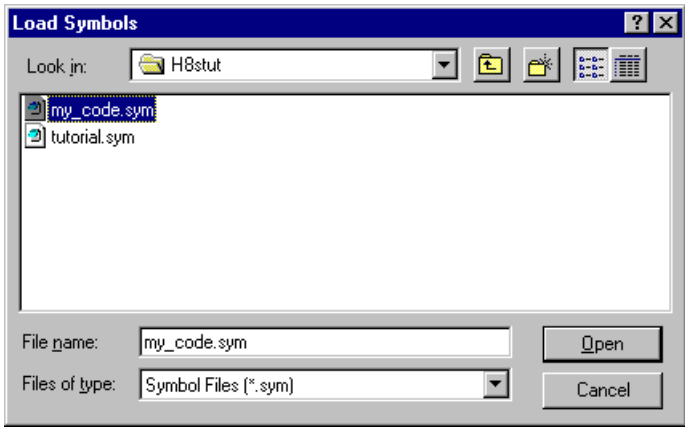


Figure 4.30 Load Symbols Dialog Box

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

4.6.7 Saving Labels into a File

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

See appendix B, Symbol File Format, for symbol file format.

Once a file is specified by the [Save As...] menu, the current symbol table can be saved in the same symbol file just by choosing [Save] from the pop-up menu.

4.6.8 Searching for a Label

Choose [Find...] from the pop-up menu to open the [Find Label] dialog box:

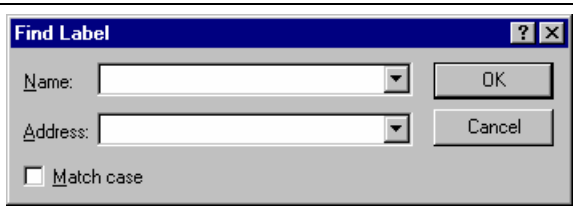


Figure 4.31 Find Label Dialog Box

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

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

4.6.9 Searching for the Next Label

Choose [Find Next] from the pop-up menu to find the next occurrence of the label containing the text that you entered.

4.6.10 Viewing the Source Corresponding to a Label

Select a label and choose [View Source] from the pop-up menu to open the [Source] or [Disassembly] window containing the address corresponding to the label.

4.7 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 [Register] window.

4.7.1 Opening the Register Window

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

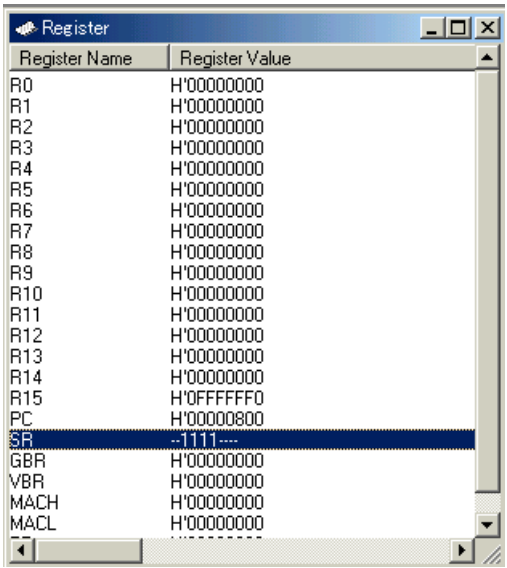


Figure 4.32 Register Window

4.7.2 Expanding a Bit Register

If a register is used as a set of flags at the bit level for the control of state, its one-character symbol rather than its state indicate each bit. Double-click on the register's name to display the [Edit Register] dialog box and switch each bit on or off. Checking the check box for any bit specifies it as holding a 1, while removing the check specifies it as a 0.

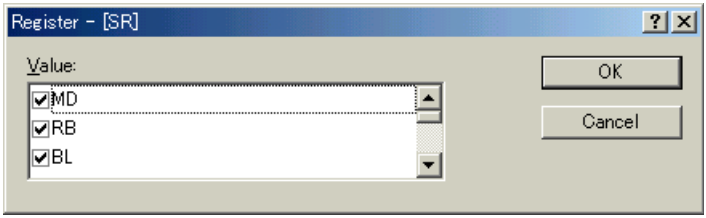


Figure 4.33 Expanding a Bit Register

4.7.3 Choosing a Register to be Displayed

To choose a register to be displayed in the [Register] window, choose [Settings…] from the pop-up menu. This dialog box is shown in figure 4.34.

Figure 4.34 Set Dialog Box

4.7.4 Splitting Up the Window Display

To vertically divide the [Register] window display into two, select [Split] from the pop-up menu and move the split-up bar. Moving the split-up bar to the top end or bottom end of the window cancels the split-up display.

4.7.5 Modifying Register Contents

To change a register's contents, open the [Edit Register] dialog box in one of the following methods:

- Enter a value directly in the window.
- Double-click the register you want to change.
- Select the register you want to change, and choose [Edit…] from the pop-up menu.

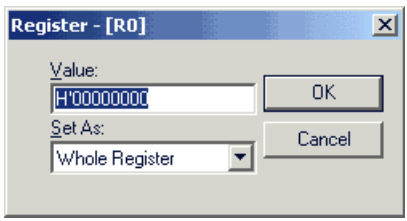


Figure 4.35 Edit Register Dialog Box

You can enter a number or C/C++ expression in the [Value] field. You can choose whether to modify the whole register contents, a masked area, floating or flag bits by selecting an option from the drop-down list (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.

4.7.6 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.

4.8 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.

4.8.1 Running from Reset

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

The program will run until it hits a breakpoint or a break condition is met. You can stop the program manually by choosing [Debug->Halt], or by clicking the [Halt] toolbar button .

Note: The program will start running from whatever address is stored in the Reset Vector location. Therefore it is important to make sure that this location contains the address of your startup code.

4.8.2 Continuing Run

When your program is stopped, 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 [Go] toolbar button, or choose [Debug->Go].

To continue running from a specified address which is not the stop address, change the PC value in one of the following ways, and click the [Go] toolbar button or choose [Debug->Go].

- Change the PC value in the [Register] window. Refer to section 4.7.5, Modifying Register Contents.
- Place the text cursor (not the mouse cursor) to a target line in the [Source] or [Disassembly] window, and choose [Set PC Here] from the pop-up menu.

4.8.3 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 pop-up 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 [Halt] toolbar button.

2. The Go To Cursor feature requires a PC breakpoint - if you have already used all those available, then the feature will not work.

4.8.4 Running from a Specified Address

The [Run Program] dialog box allows the user to run the program from any address. Choose [Debug -> Run...] to open the [Run Program] dialog box.

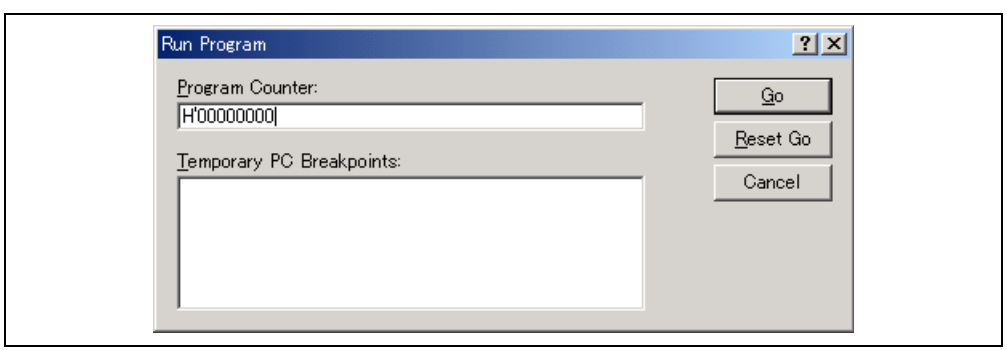


Figure 4.36 Run Program Dialog Box

The following execution conditions can be specified in this dialog box:

[Program Counter]	Instruction address to start execution. The initial value is the current PC value.
[Temporary PC Breakpoints]	A temporary PC breakpoint. When execution started by this dialog box stops, this breakpoint is cleared.

Note: The [Temporary PC Breakpoints] feature requires a PC breakpoint - if you have already used all those available then the feature will not work.

Clicking the [Go] button starts execution according to the settings. Clicking the [Reset Go] button starts execution from the reset vector. Clicking the [Cancel] button closes this dialog box without executing instructions.

4.8.5 Single Step

To debug your code it is very useful to be able to step a single line or instruction at a time and examine the effect of that instruction on the system. In the [Editor] window, 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.

4.8.6 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].

4.8.7 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].

4.8.8 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].

4.8.9 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:

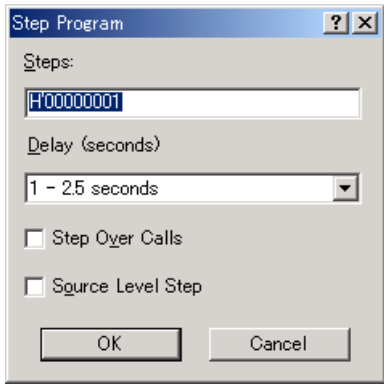


Figure 4.37 Step Program Dialog Box

[Steps]	Number of steps to be executed.
[Delay (seconds)]	Delay between steps when the program is automatically stepped. Value 0 to 6 can be specified where value 0 indicates the longest delay.
[Step Over Calls]	Selecting this box steps over function calls.
[Source Level Step]	Selecting this box steps the program at the source level.

Clicking the [OK] button or pressing the Enter key starts step execution.

4.8.10 Executing Multiple Targets

Multiple targets can be debugged in synchronous execution. For details, refer to section 4.23, Synchronizing Multiple Debugging Platforms.

4.9 Stopping Your Program

This section describes how you can halt execution of your application's code. This section describes how to do this directly by using the [Halt] button and by setting breakpoints at specific locations in your code.

4.9.1 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). Click on the [Halt] toolbar button, or choose [Debug->Halt Program].

When the program has been stopped by [Halt], "Stop" is displayed in the [Debug] sheet of the [Output] window.

4.9.2 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 required, use the [Event] window,

which can be opened by clicking the button or choosing [View -> Code -> Event]. For details, refer to section 4.15, Using the Simulator/Debugger Breakpoints.

- To set a PC breakpoint in the [Editor] window
  1. Make sure that the [Disassemble] or an [Editor] window is open at the place you want to set a PC breakpoint.
  2. Choose [Toggle Breakpoint] from the pop-up menu, or press F9, at the line showing the address at which you want the program to stop.
  3. You will see a red circle appear in the gutter to indicate that a PC breakpoint has been set.
  4. The current breakpoint set can be enabled or disabled by using [Enable/Disable Breakpoint] in the pop-up menu.

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 Breakpoint" displayed in the [Debug] sheet of the [Output] window, and the [Source] or [Disassembly] window is updated with the PC breakpoint line marked with an arrow in the gutter.

Note: When a break occurs, the program stops just before it is about to execute the line or instruction at which you set a program PC breakpoint. If you choose Go or Step after stopping at the PC breakpoint, then the line marked with an arrow will be the next instruction to be executed. When multiple targets are debugged, it is possible to specify either or both of them is stopped. For details, refer to section 4.23, Synchronizing Multiple Debugging Platforms.

- To set a PC breakpoint by using the [Breakpoints] dialog box

  The [Breakpoints] dialog box is shown in figure 4.38. It allows the user to view the current breakpoints set. Clicking the [Edit Code] button displays the source where each breakpoint is set. The [Remove] or [Remove All] button deletes one or all breakpoints, respectively. The check box of each breakpoint enables or disables the breakpoint.

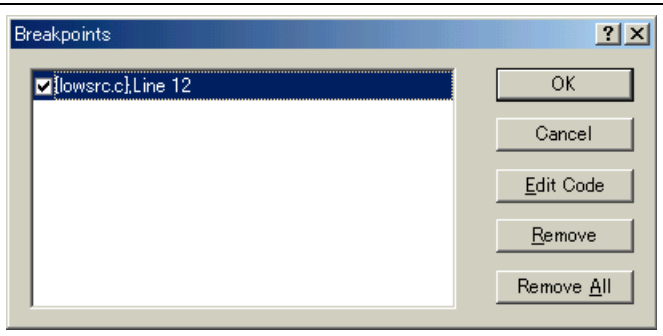


Figure 4.38 Breakpoints Dialog Box

- Toggling PC breakpoints

  It is possible to toggle the [PC Breakpoints] setting by either double-clicking in the [BP] column of the line where the PC breakpoint is set or placing the cursor on the line and pressing the F9 key. The setting to be toggled depends on the debugging platform.

4.10 Elf/Dwarf2 Support

The HEW supports the Elf/Dwarf2 object file format for debugging applications written in C/C++ and assembly language for Renesas 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

4.10.1 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)
```

4.10.2 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++)
```

4.10.3 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.

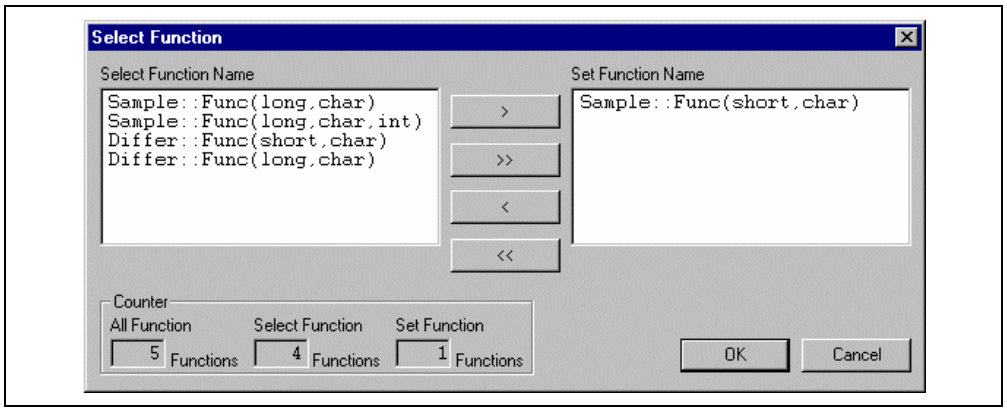


Figure 4.39 Select Function Dialog Box

Select overloaded functions or member functions in the [Select Function] dialog box. Generally, one function can be selected at one time; only for setting breakpoints, multiple functions can be selected. This dialog box has three areas.

[Select Function Name]	Displays the same-name functions or member functions and their detailed information.	
[Set Function Name]	Displays the function to be set and their detailed information.	
[Counter]	[All Function]	Displays the number of same-name functions or member functions.
	[Select Function]	Displays the number of functions displayed in the [Select Function Name] list box.
	[Set Function]	Displays the number of functions displayed in the [Set Function Name] list box.

Selecting a Function

Click the function you wish to select in the [Select Function Name] list box, and click the [>] button. You will see the selected function in the [Set Function Name] list box. To select all functions in the [Select Function Name] list box, click the [>>] button.

Deselecting a Function

Click the function you wish to deselect from the [Set Function Name] list box, and click the [<] button. To deselect all functions, click the [<<] button. The deselected function will be moved from [Set Function Name] list box back to the [Select Function Name] list box.

Setting a Function

Click the [OK] button to set the functions displayed in the [Set Function Name] list box. The functions are set and the [Select Function] dialog box closes.

Clicking the [Cancel] button closes the dialog box without setting the functions.

4.10.4 Debugging an Overlay Program

This section explains the settings for using the overlay functions.

Displaying Section Group

When the overlay mode is used, that is, when several section groups are assigned to the same address range, the address ranges and section groups are displayed in the [Overlay] dialog box.

Open the [Overlay] dialog box by choosing [Memory->Configure Overlay].

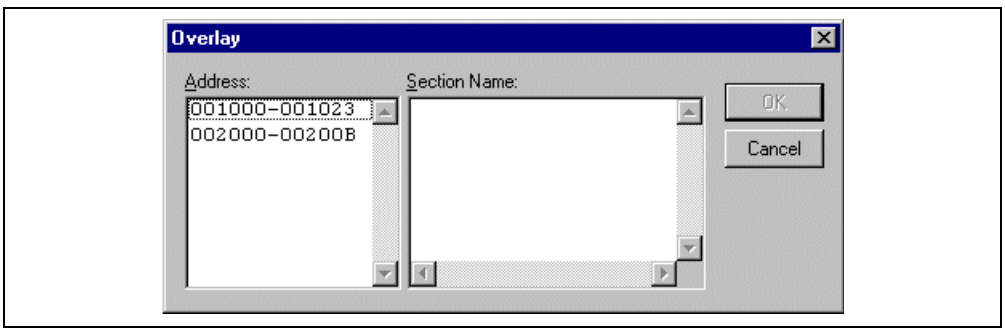


Figure 4.40 Overlay Dialog Box (at Opening)

This dialog box has two areas: the [Address] list box and the [Section Name] list box.

The [Address] list box displays the address ranges used in the overlay mode. Click to select one of the address ranges in the [Address] list box.

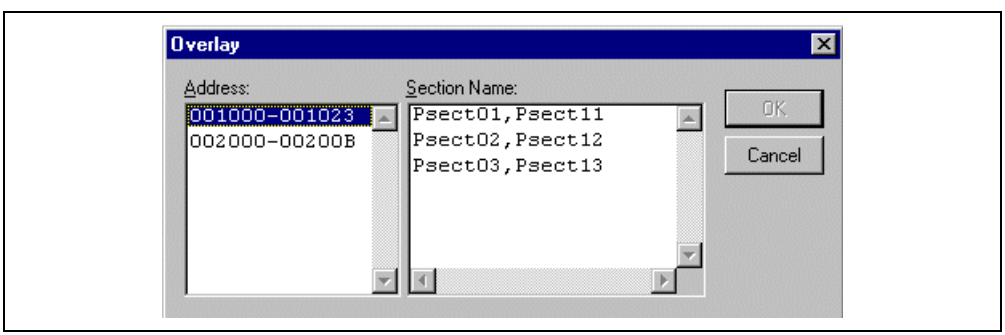


Figure 4.41 Overlay Dialog Box (Address Range Selected)

The [Section Name] list box displays the section groups assigned to the selected address range.

➲ Setting Section Group

When using the overlay function, the highest-priority section group must be selected in the [Overlay] dialog box; otherwise the HEW will operate incorrectly.

First click one of the address ranges displayed in the [Address] list box. The section groups assigned to the selected address range will then be displayed in the [Section Name] list box.

Click to select the section group with the highest-priority among the displayed section groups.

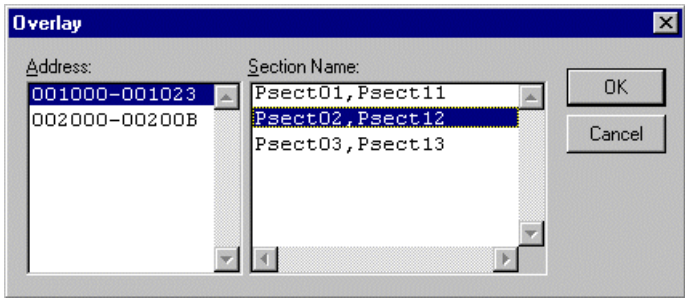


Figure 4.42 Overlay Dialog Box (Highest-Priority Section Group Selected)

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.

4.11 Looking at Variables

This section describes how you can look at variables in the source program.

4.11.1 Tooltip Watch

The quickest way to look at a variable in your program is to use the Tooltip Watch feature.

➲ To use Tooltip Watch:

Display the source file containing the variable that you want to examine on the [Editor] window.
Rest the mouse cursor over the variable name that you want to examine - a tooltip will appear near the variable showing a value of that variable.

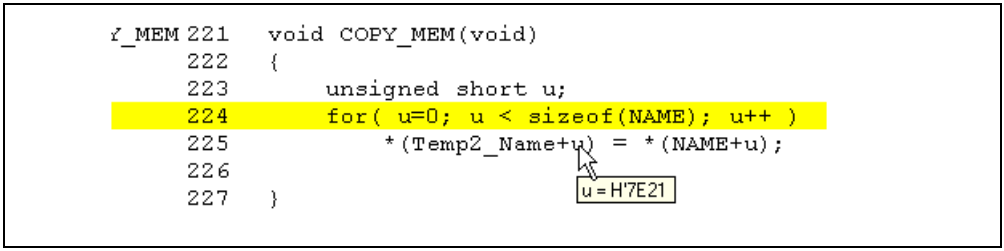


Figure 4.43 Tooltip Watch

4.11.2 Instant Watch

Display the source file containing the variable that you want to examine on the [Editor] window.
Rest the mouse cursor over the variable name that you want to examine and choose [Instant Watch...] from the pop-up menu; the [Instant Watch] dialog box will appear and display the variable at the cursor location.

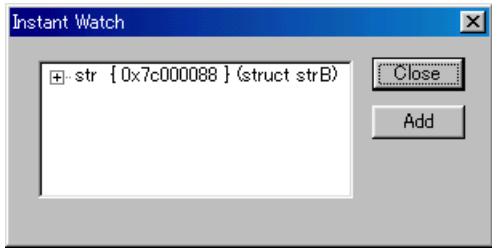


Figure 4.44 Instant Watch Dialog Box

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

4.11.3 Watch Window

You can view any value in the [Watch] window.

Opening a [Watch] Window

To open a [Watch] window, choose [View->Symbol->Watch] or click on the [Watch] toolbar button if it is visible. A [Watch] window opens. Initially the contents of the window will be blank.

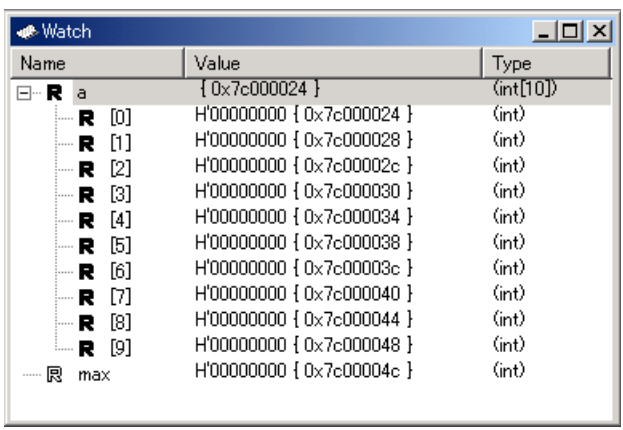


Figure 4.45 Watch Window

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

The following items are displayed.

[Name] Name of the variable

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

[Type] Type of the variable

The [R] mark shows that the value of the variable can be updated during user program execution. When the color of the [R] mark is black, a value is updated in an instruction interval specified with the [Response] command

Adding a Watch Item

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

➲ To use Add Watch from a [Watch] window:

Open the [Watch] window.
Choose [Add Watch] from the pop-up menu.
The [Add Watch] dialog box opens:

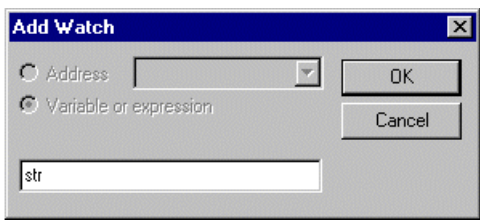


Figure 4.46 Add Watch Dialog Box

Enter the name of the variable that you wish to watch and click [OK]. The variable is added to the [Watch] window. A variable can be dragged from the [Editor] window and dropped into the [Watch] window.

Note: If the variable that you have added is a local variable that is not currently in scope, the HEW will add it to the [Watch] window but its value will be 'Not available now'.

Expanding a Watch Item

If a watch item is a pointer, array, or structure, then you will see a plus sign (+) expansion indicator to left of its name, this means that you can expand the watch item. To expand a watch item, click on it. The item expands to show the elements (in the case of structures and arrays) or data value (in the case of pointers) indented by one tab stop, and the plus sign changes to a minus sign (-). If the elements of the watch item also contain pointers, structures, or arrays then they will also have expansion indicators next to them.

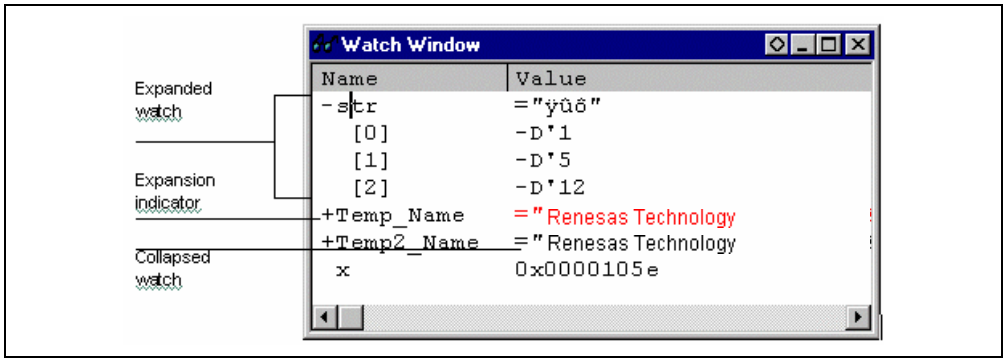


Figure 4.47 Expanding a Watch Item

To collapse an expanded watch item, click on the item again. The item's elements will collapse back to the single item and the minus sign changes back to a plus sign.

Editing a Watch Item's Value

You may wish to change the value of a watch variable, e.g. for testing purposes or if the value is incorrect due to a bug in your program. To change a watch item's value use the Edit Value function.

➲ Editing a watch item's value:

Enter a value directly in the window.

In another way, select the item to edit by clicking on it, you will see a flashing cursor on the item.
Choose [Edit Value] from the pop-up menu.
The [Edit Value] dialog box opens:

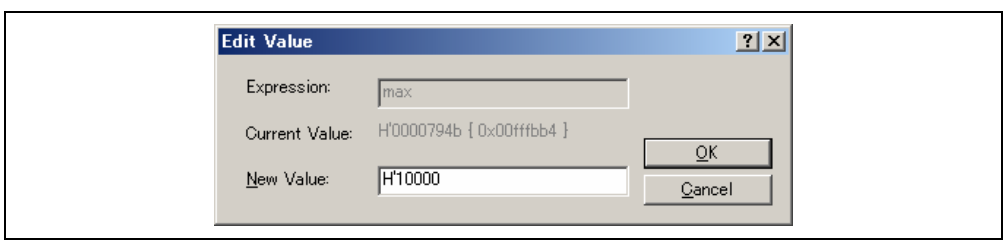


Figure 4.48 Edit Value Dialog Box

Enter the new value or expression in the [New Value] field and click [OK]. The [Watch] window is updated to show the new value.

Deleting a Watch Item

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

To delete all watch items, choose [Delete All] from the pop-up menu. All items are deleted and the [Watch] window is updated.

Specifying Realtime Update

The R mark shown to the left of each variable indicates whether the variable is updated in real time. When an R mark is black, the value of the corresponding variable will be automatically updated in an instruction interval specified with the [Response] command during user program execution.

A pop-up menu containing the following options is available in the [Watch] window:

- Auto Update
  Marks the selected variable with a bold R and updates the variable in real time.
- Auto Update All
  Marks all variables with bold Rs and updates all variables in real time.
- Delete Auto Update
  Marks the selected variable with an outlined R and cancels realtime update.
- Delete Auto Update All
  Marks all variables with outlined Rs and cancels realtime update.

Modifying the Radix

The radix for the selected variable display can be modified by choosing [Radix] from the pop-up menu.

Saving the Watch Window Contents in a File

To save the contents of the [Watch] window, choose [Save As...] from the pop-up menu; the Save As dialog box opens. It allows the user to specify the name of a file and to save the contents of the [Watch] window in the file. If the [Append] check box is selected, the window contents are appended to the existing file, and if it is not selected, the existing file is overwritten.

Opening a Memory Window

The contents of the memory area to which the selected variable is assigned can be displayed in the [Memory] window. Choose [Go To Memory...] from the pop-up menu; the [Set Address] dialog box opens, showing the information (start address, end address, and size) of the selected variable as default. Clicking [OK] opens the [Memory] window.

4.11.4 Locals Window

The local variables and their values can be displayed in the [Locals] window.

Opening the Locals Window

To open the [Locals] window, choose [View->Symbol->Locals] or click the [Locals] toolbar button  .

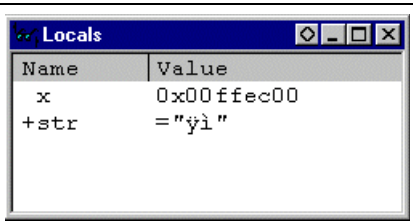


Figure 4.49 Locals Window

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.

4.12 Viewing the Profile Information

The profile function enables function-by-function measurement of the performance of the application program in execution. This makes it possible to identify parts of an application program that degrade its performance and the reasons for such degradation.

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

4.12.1 Stack Information Files

The profile function allows the HEW to read the stack information files (extension: ".SNI") which are output by the Optimizing Linker (ver. 7.0 or later). Each of these files contains information related to the calling of static functions in the corresponding source file. Reading the stack information file makes it possible for the HEW to display this information to do with the calling of functions without executing the user application (i.e. before measuring the profile data). However, this feature is not available when [Setting->Only Executed Functions] is checked in the pop-up menu of the [Profile] window.

When the HEW does not read any stack information files, the data about the functions executed during measurement will be displayed by the profile function.

To make the linker create a stack information file, choose [Options -> H8,H8/300 siries Standard Toolchain…], and select [Other] from the [Category] list box and check the [Stack information output] box in the [Link/Library] sheet of the [Standard Toolchain] dialog box.

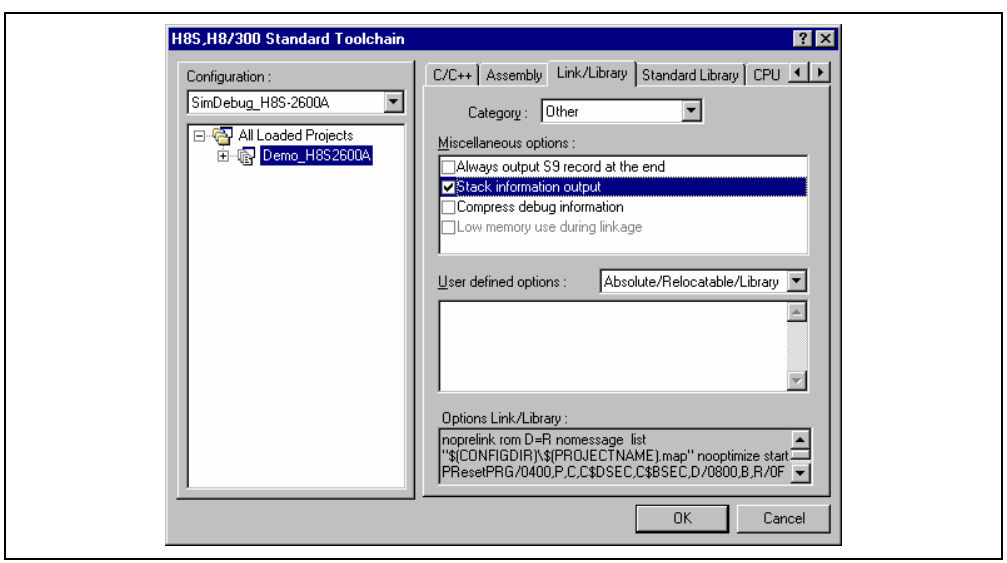


Figure 4.50 Standard Toolchain Dialog Box (1)

4.12.2 Profile Information Files

To create a profile information file, choose the [Output Profile Information Files…] menu option from the pop-up menu of the [Profile] window and specify the file name, after measuring a profile data of the application program.

This file contains information on the number of times functions are called and global variables are accessed. The Optimizing Linker (ver. 7.0 or later) is capable of reading the profile information file and optimizing the allocation of functions and variables in correspondence with the status of the actual operation of the program.

To input the profiler information file to the linker, choose [Optimize] from the [Category] list box and check the [Include Profile] box in the [Link/Library] sheet of the [Standard Toolchain] dialog box, and specify the name of the profile information file.

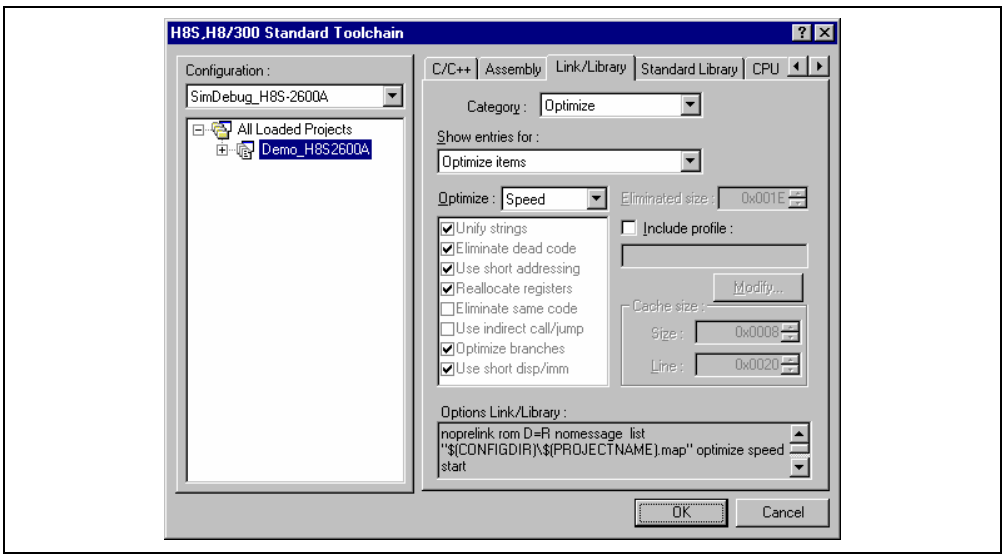


Figure 4.51 Standard Toolchain Dialog Box (2)

To enable the settings in the [Include Profile] box, specify the [Optimize] list box as some setting other than [None].

4.12.3 Loading Stack Information Files

You can select whether or not to read the stack information file in a message box for confirmation that is displayed when a load module is loaded. Clicking the [OK] button of the message box loads the stack information file. The message box for confirmation will be displayed when:

- There are stack information files (extension: "*.SNI").
- The [Load Stack Information Files (SNI files)] check box is checked in the [Confirmation] sheet of the [Options] dialog box (figure 4.52) that can be opened by choosing [Tools->Options] from the main menu.

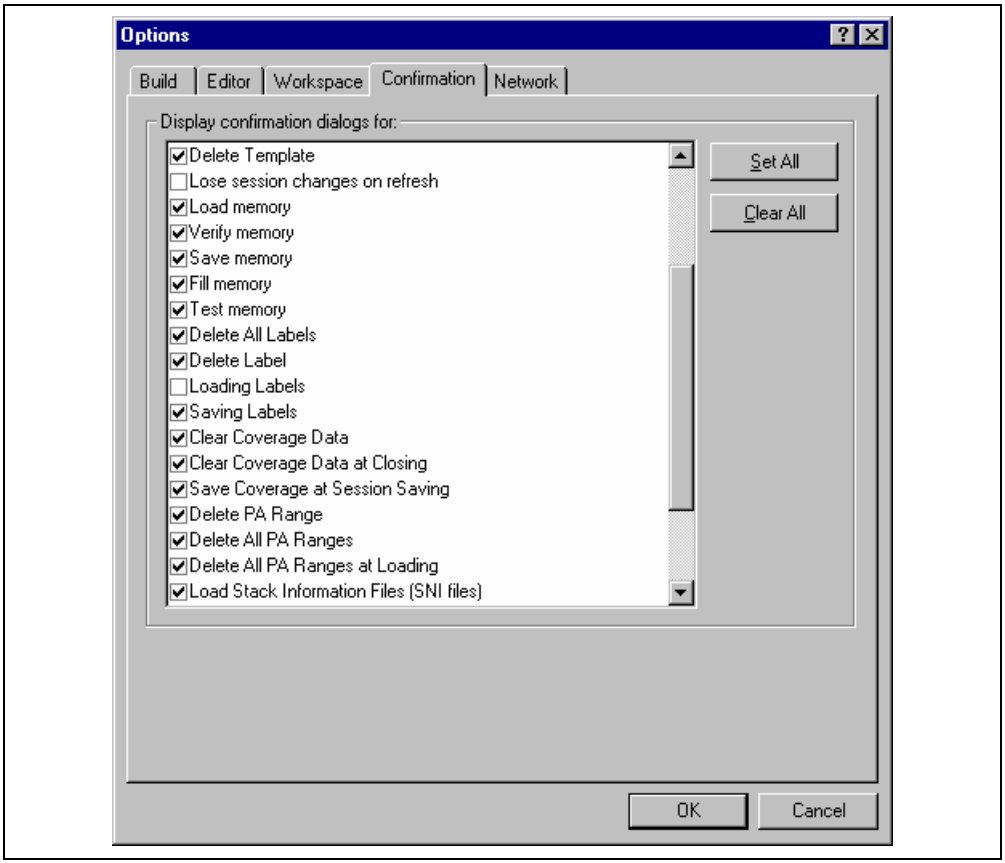


Figure 4.52 Options Dialog Box

4.12.4 Enabling the Profile

Choose [View->Performance->Profile] to open the [Profile] window.

Choose [Enable Profiler] from the pop-up menu of the [Profile] window. The item on the menu will be checked.

4.12.5 Specifying Measuring Mode

You can specify whether to trace functions calls while profile data is acquired. When function calls are traced, the relations of function calls during user program execution are displayed as a tree diagram. When not traced, the relations of function calls cannot be displayed, but the time for acquiring profile data can be reduced.

To stop tracing function calls, choose [Disable Tree (Not traces function call)] from the pop-up menu in the [Profile] window (a check mark is shown to the left of the menu item).

When acquiring profile data of the program in which functions are called in a special way, such as task switching in the OS, stop tracing function calls.

4.12.6 Executing the Program and Checking the Results

After the user program has been executed and execution has been halted, the results of measurement are displayed in the [Profile] window.

The [Profile] window has two sheets; a [List] sheet and a [Tree] sheet.

4.12.7 List Sheet

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

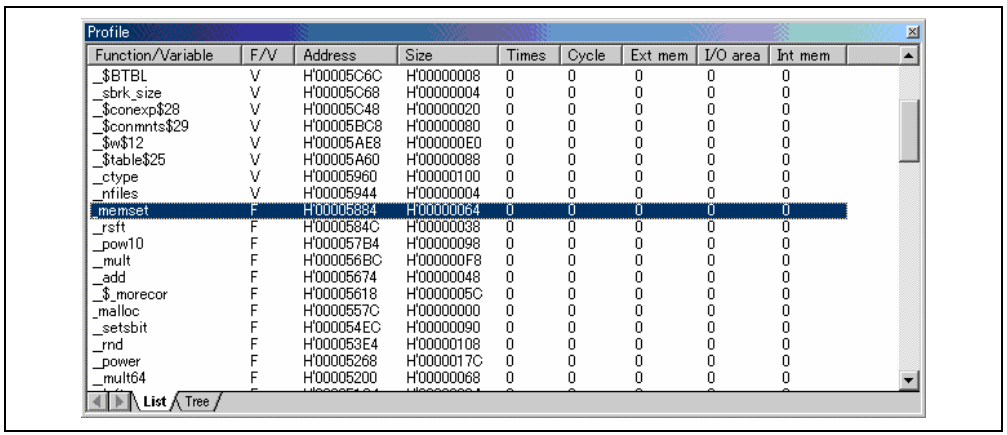


Figure 4.53 List Sheet

Clicking the column header sorts the items in an alphabetical or ascending order. Double-clicking the [Function/Variable] or [Address] column displays the source program corresponding to the address in the line.

Right-clicking on the mouse within the window displays a pop-up menu. For details on this pop-up menu, refer to section 4.12.8, Tree Sheet.

4.12.8 Tree Sheet

This sheet displays the relation of function calls along with the profile data that are values when the function is called. This sheet is available when [Disable Tree (Not traces function call)] is not selected from the pop-up menu in the [Profile] window.

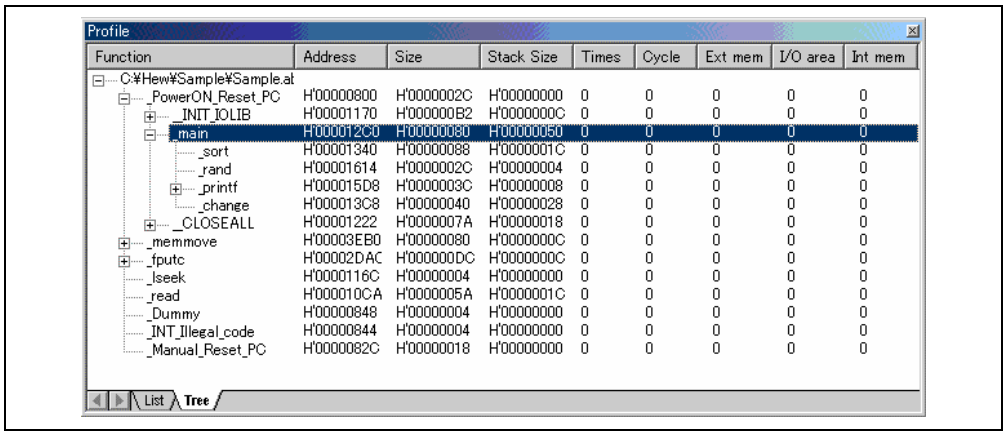


Figure 4.54 Tree Sheet

Double-clicking a function in the [Function] column expands or reduces the tree structure display. The expansion or reduction is also provided by the "+" or "-" key. Double-clicking the [Address] column displays the source program corresponding to the specific address.

Right-clicking on the mouse within the window displays a pop-up menu. Supported menu options are described in the following:

- View Source

  Displays the source program or disassembled memory contents for the address in the selected line.
- View Profile-Chart

  Displays the [Profile-Chart] window focused on the function in the specified line.
- Enable Profiler

  Toggles acquisition profile data. When profile data acquisition is active, a check mark is shown to the left of the menu text.
- Not trace the function call

  Stops tracing function calls while profile data is acquired. This menu is used when acquiring profile data of the program in which functions are called in a special way, such as task switching in the OS.

  To display the relation of function calls in the [Tree]sheet of the [Profile] window, acquire profile data without selecting this menu. In addition, do not select this menu when optimizing the program by the optimizing linkage editor using the acquired profile information file.
- Find…

  Displays the [Find Text] dialog box to find a character string in the [Function] column. Search is started by inputting a character string to be found in the edit box and clicking [Find Next] or pressing the Enter key.
- Find Data…

  Displays the [Find Data] dialog box.

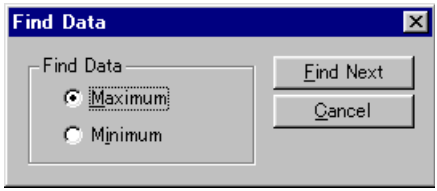


Figure 4.55 Find Data Dialog Box

By selecting the column to be searched in the [Column] combo box and the search type in the [Find Data] group and entering [Find Next] button or Enter key, search is started. If the [Find Next] button or the Enter key is input repeatedly, the second larger data (the second smaller data when the Minimum is specified) is searched for.

- Clear Data
  Clears the number of times functions are called and profile data. Data in the [List] sheet of the [Profile] window and the data in the [Profile-Chart] window are also cleared.
- Output Profile Information Files…
  Displays the [Save Profile Information Files] dialog box. Profiling results are saved in a profile information file (.pro extension). The optimizing linkage editor optimizes user programs according to the profile information in this file. For details of the optimization using the profile information, refer to the manual of the optimizing linkage editor.

Note: If profile information has been acquired by choosing the [Not trace the function call] menu, the program cannot be optimized by the optimizing linkage editor.

- Output Text File…
  Displays the [Save Text of Profile Data] dialog box. Displayed contents are saved in a text file.
- Setting
  This menu has the following submenus (the menus available only in the [List] sheet are also included).
  1. Show Functions/Variables
     Displays both functions and global variables in the [Function/Variable] column.
  2. Show Functions
     Displays only functions in the [Function/Variable] column.
  3. Show Variables
     Displays only global variables in the [Function/Variable] column.
  4. Only Executed Functions
     Only displays the executed functions. If a stack information file (.sni extension) output from the optimizing linkage editor does not exist in the directory where the load module is located, only the executed functions are displayed even if this check box is not checked.
  5. Include Data of Child Functions
     Sets whether or not to display information for a child function called in the function as profile data.
- Properties...
  This menu cannot be used in this simulator/debugger.

4.12.9 Profile-Chart Window

The [Profile-Chart] window displays the relation of calls for a specific function. This window displays the specified function in the middle, with the callers of the function on the left and the callees of the function on the

right. The numbers of times the function calls the called functions or is called by the calling functions are also displayed in this window.

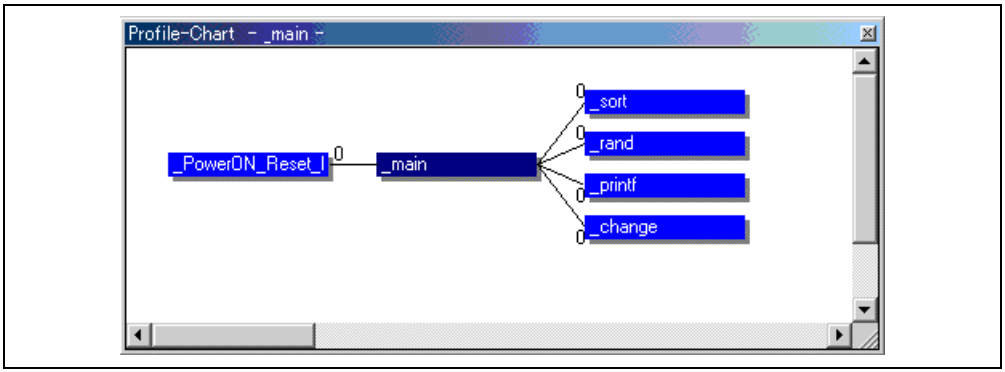


Figure 4.56 Profile-Chart Window

4.12.10 Types and Purposes of Displayed Data

The profile function is able to acquire the following information:

Address	You can see the locations in memory to which the functions are allocated. Sorting the list of functions and global variables in order of their addresses allows the user to view the way the items are allocated in the memory space.
Size	Sorting in order of size makes it easy to find small functions that are frequently called. Setting such functions as inline may reduce the overhead of function calls.
Stack Size	When there is deep nesting of function calls, pursue the route of the function calls and obtain the total stack size for all of the functions on that route to estimate the amount of stack being used.
Times	Sorting by the number of calls or accesses makes it easy to identify the frequently called functions and frequently accessed global variables.

Profile Data

- [Times] (the number of function calls or global variable accesses)
- [Cycle] (the number of execution cycles)
- [Ext_mem] (the number of external memory accesses)
- [I/O_area] (the number of internal I/O area accesses)
- [Int_mem] (the number of internal memory accesses)

The number of execution cycles is calculated by subtracting the total execution cycles at a specific function call instruction execution from the total execution cycles at a return instruction execution of a specific function.

4.12.11 Creating Profile Information Files

To create a profile information file, choose the [Output Profile Information Files…] menu option from the pop-up menu. The [Save Profile Information Files] dialog box is displayed. Pressing the [Save] button after selecting

a file name will write the profile information to the selected file. Pressing the [Save All] button will write the profile information to all of the profile information files.

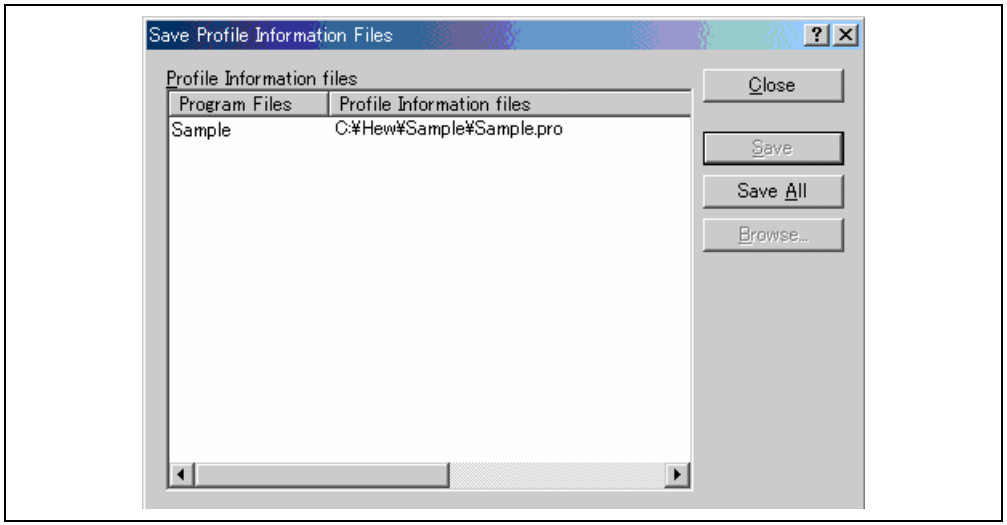


Figure 4.57 Save Profile Information Files Dialog Box

4.12.12 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.

4.13 Viewing the Function Call History

The [Stack Trace] window shows the function call history.

4.13.1 Opening the Stack Trace Window

To open the [Stack Trace] window, choose [View -> Code -> Stack Trace] or click the [Stack Trace] toolbar button  .

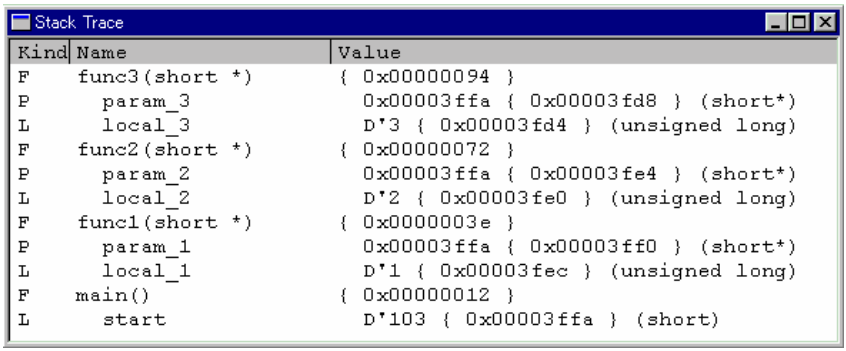


Figure 4.58 Stack Trace Window

The following items are displayed.

[Kind] Indicates the type of the symbol.
F: Function
P: Function parameter
L: Local variable

[Name] Indicates the symbol name.

[Value] Indicates the value, address, and type of the symbol.

4.13.2 Viewing the Source Program

Select a function and choose [Go to Source] from the pop-up menu to display, then the source program corresponding to the function, which has been selected by opening the [Editor] window, is displayed.

4.13.3 Specifying the View

Choose [View Setting...] from the pop-up menu to open the [Stack Trace Setting] dialog box, which allows the user to specify the [Stack Trace] window settings.

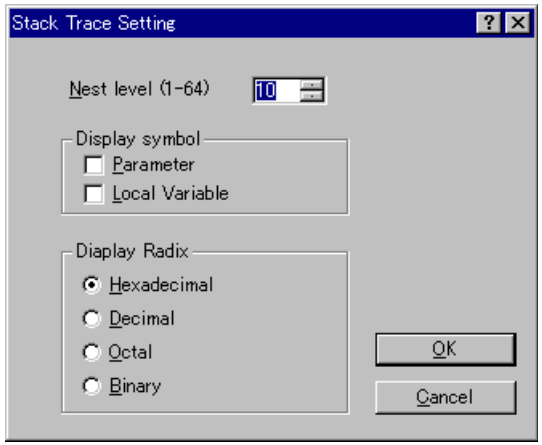


Figure 4.59 Stack Trace Setting Dialog Box

[Nest level] Specifies the level of function call nesting to be displayed in the [Stack Trace] window.

[Display symbol] Specifies the symbol types to be displayed in addition to functions.

[Display Radix] Specifies the radix for displays in the [Stack Trace] window.

4.14 Viewing the Trace Information

The simulator/debugger acquires the results of each instruction execution as trace information and displays it in the [Trace] window. The conditions for the trace information acquisition can be specified in the [Trace Acquisition] dialog box.

4.14.1 Opening the Trace Window

To open the [Trace] window, choose [View -> Code -> Trace] or click the [Trace] toolbar button .

4.14.2 Specifying Trace Acquisition Conditions

After the [Trace] window opens, the trace acquisition conditions must be specified in the [Trace Acquisition] dialog box. To open this dialog box, choose [Acquisition...] from the pop-up menu.

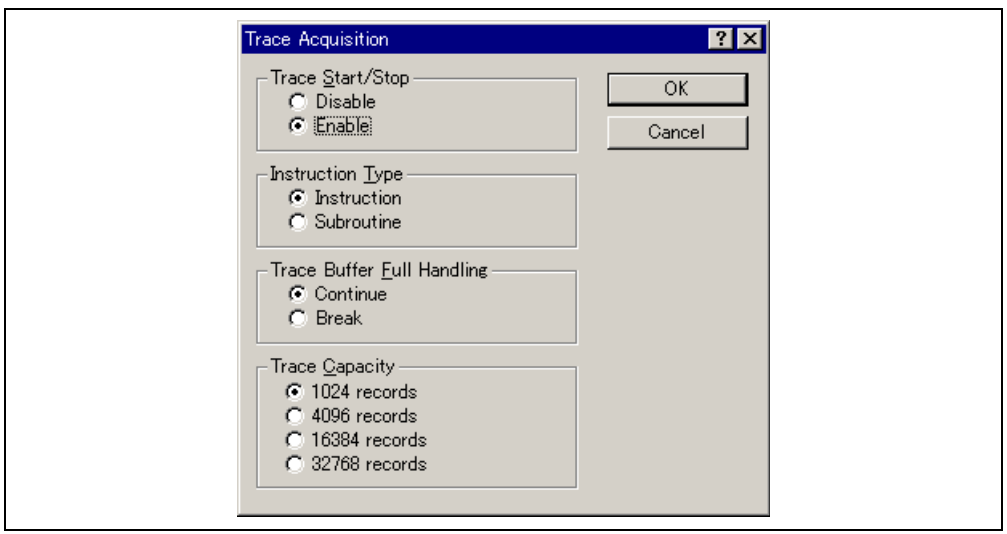


Figure 4.60 Trace Acquisition Dialog Box

This dialog box specifies the conditions for trace information acquisition.

[Trace Start/Stop]

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

[Instruction Type]

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

[Trace Buffer Full Handling]

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

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

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

4.14.3 Acquiring Trace Information

After trace acquisition is enabled, trace information is acquired during instruction execution. The acquired information will be displayed in the [Trace] window.

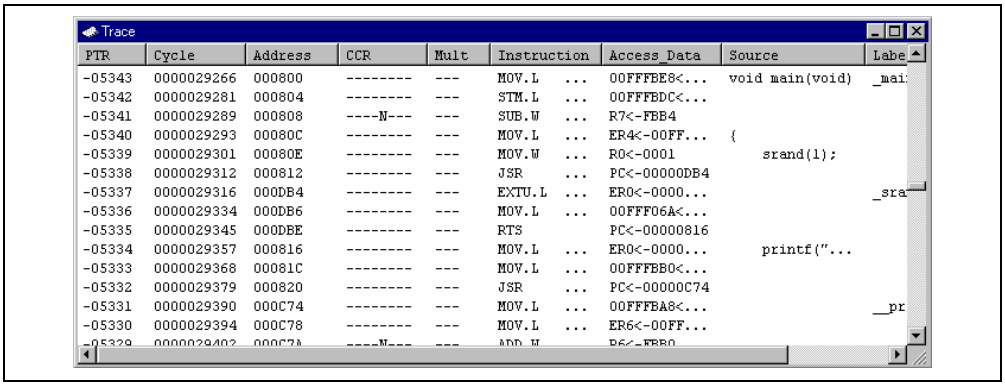


Figure 4.61 Trace Window

This window displays the following trace information items:

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

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

[Address] Instruction address

[CCR] Displays the contents of the condition code register (CCR) as a mnemonic.

[Mult] Displays the flag in the multiplier as a mnemonic (H8S/2600 series only).

[Instruction] Instruction mnemonic

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

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

4.14.4 Searching for a Trace Record

Use the [Trace Search] dialog box to search for a trace record. To open this dialog box, choose [Find...] from the pop-up menu.

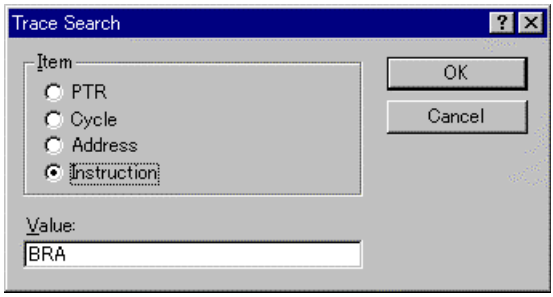


Figure 4.62 Trace Search Dialog Box

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

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

[Cycle] Total number of instruction execution cycles

[Address] Instruction address

[Instruction] Instruction mnemonic

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

When a trace record that matches the search conditions is found, the line for the trace record will be highlighted. When no matching trace record is found, a message dialog box will appear.

If a find operation is successful, choosing [Find Next] from the pop-up menu will move to the next found item.

4.14.5 Clearing the Trace Information

Choose [Clear] form the pop-up menu to empty the trace buffer that stores the trace information. If more than one [Trace] window is open, all [Trace] windows will be cleared as they all access the same buffer.

4.14.6 Saving the Trace Information in a File

Choose [Save...] from the pop-up menu to open the [Save As] file dialog box, which allows the user to save the contents of the trace buffer as a text file. A range can be specified based on the [PTR] number (saving the complete buffer may take several minutes). Note that this file cannot be reloaded into the trace buffer.

4.14.7 Viewing the Source File

The [Editor] window corresponding to the selected trace record can be displayed in the following two ways:

- Select a trace record and choose [View Source] from the pop-up menu.
- Double-click a trace record

The [Editor] or [Disassembly] window opens and the selected line is marked with a cursor.

4.14.8 Trimming the Source

Choose [Trim Source] from he pop-up menu to remove the white space from the left side of the source.

When the white space is removed, a check mark is shown to the left of the [Trim Source] menu. To restore the white space, choose [Trim Source] while the check mark is shown.

4.14.9 Analyzing Statistical Information

Choose [Statistic] from the pop-up menu to open the [Trace Statistic] dialog box and analyze statistical information under the specified conditions.

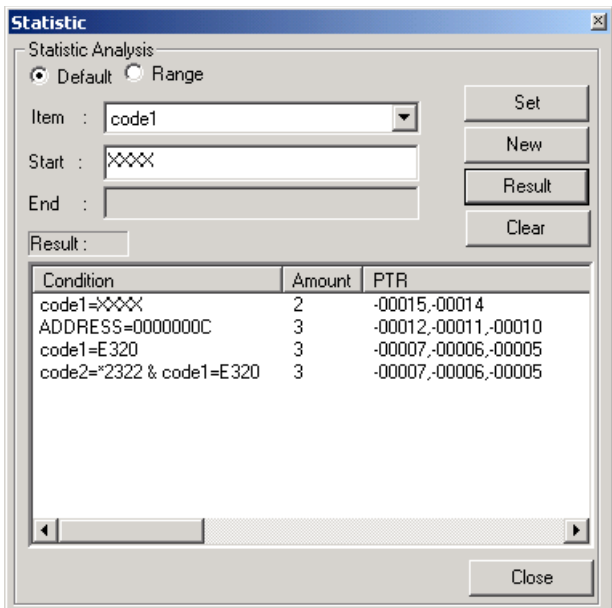


Figure 4.63 Trace Statistic Dialog Box

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

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

[Set] Adds a new condition to the current one

[New] Creates a new condition

[Result] Obtains the result of statistical information analysis

[Clear] Clears all condition and results of statistical information analysis

Clicking the [Close] button closes this dialog box.

4.15 Using the Simulator/Debugger Breakpoints

Sophisticated breakpoint functions are available in the simulator/debugger in addition to the HEW standard PC breakpoints. The user can specify break conditions and actions after a break condition is satisfied, and can display the breakpoints set.

4.15.1 Listing the Breakpoints

Choose [View -> Code -> Eventpoints] or click the [Eventpoints] toolbar button  to open the [Event] window, which lists the breakpoints set.

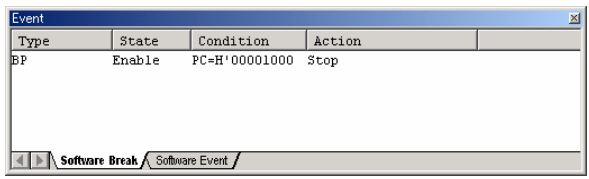


Figure 4.64 Event Window

The following items are displayed:

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

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

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

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

For [Action], the condition specifying [Stop] is displayed on the [Software Break] tab and the condition specifying others is on the [Software Event] tab.

4.15.2 Setting a Breakpoint

Choose [Add...] from the pop-up menu in the [Event] window to open the [Set Break] dialog box, which allows the user to set a break condition.

The [Set Break] dialog box has two pages: [Condition] page for specifying break conditions and [Action] page for specifying the action when a break condition is satisfied. In the [Action] page, the pop-up menu is selected on the [Software Break] and [Software Event] tabs for specifying [Stop] and others, respectively.

Specifying Break Conditions:

Specify break conditions in the [Condition] page in the [Set Break] dialog box (figure 4.65).

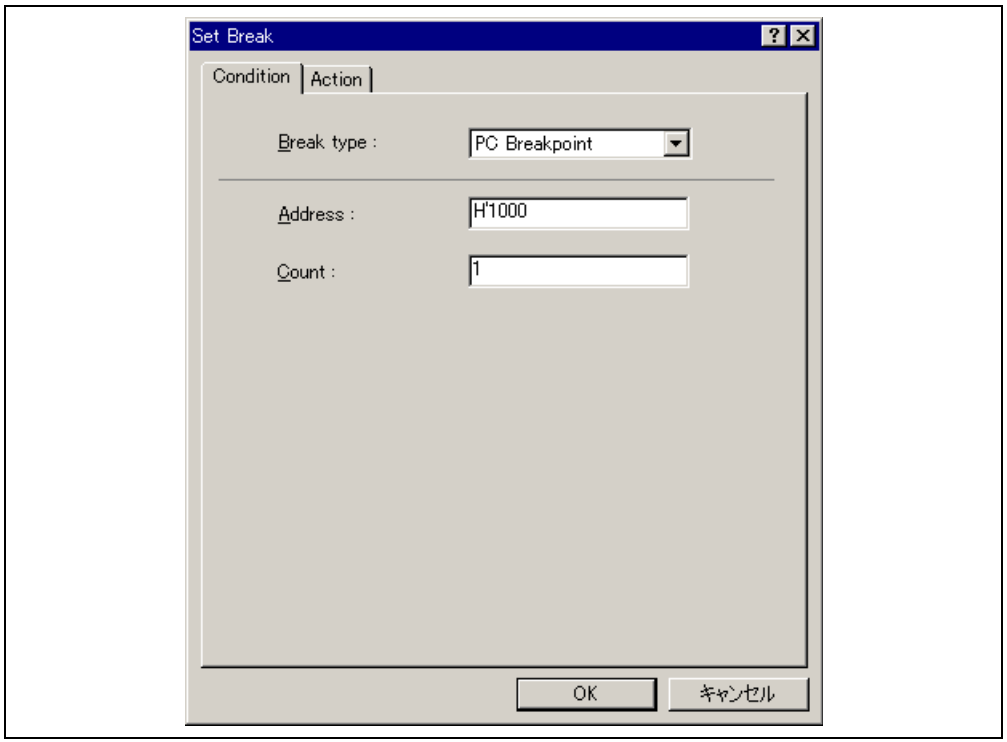


Figure 4.65 Set Break Dialog Box (Condition Page)

This dialog box specifies break conditions.

Set a break type in the [Break type] box. The following shows the conditions that can be specified in each type.

[PC Breakpoint] Up to 1024 breakpoints can be specified
- [Address] Address where a break occurs
- [Count] Number of times that a specified instruction is fetched (when the prefix is omitted, values must be input and are displayed in decimal) (1 to 16383, default: 1)

[Break Access] Up to 1024 addresses can be specified
- [Start address] Start address of memory where a break occurs if the memory is accessed

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

[Break Data] Up to 1024 values of data can be specified
[Start address] Address of memory where a break occurs
[Data] Data value that causes a break
[Size] Data size
[Option] Data match/mismatch

[Break Register] Up to 1024 register names can be specified
[Register] Register name where break conditions are specified
[Size] Data size
[Data] Data value that causes a break (If no data is input, a break occurs whenever data is written to the register)
[Option] Data match/mismatch

[Break Sequence] Only one address can be specified
[Address1] to [Address8] Pass addresses that are the conditions to generate a break. (All eight breakpoints do not have to be set.)

[Break Cycle] Up to 1024 cycles can be specified
[Cycle] Number of cycles to determine a break (H'1 to H'FFFFFFFF).
A condition will be satisfied by a number of cycles of [Cycle] x n. However, the specified number of cycles and the number of cycles that satisfied the condition may be different.
[Count] Number of times breaks will occur
[ALL] A break will occur whenever a condition is satisfied.
[Times] (when the prefix is omitted, values must be input and are displayed in hexadecimal) (1 to 65535)
A break will occur only when the number of times the conditions is satisfied is equal to or below the value specified for [Times].

When [PC Breakpoint] and [Break Sequence] is selected, if an overloaded function or class name including a member function is specified in address, the [Select Function] dialog box opens. In the dialog box, select a function. For details, refer to section 4.10.3, Supporting Duplicate Labels.

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

Specifying the Action at Break:

Specify the action when a break condition is satisfied in the [Action] page in the [Set Break] dialog box (figure 4.66).

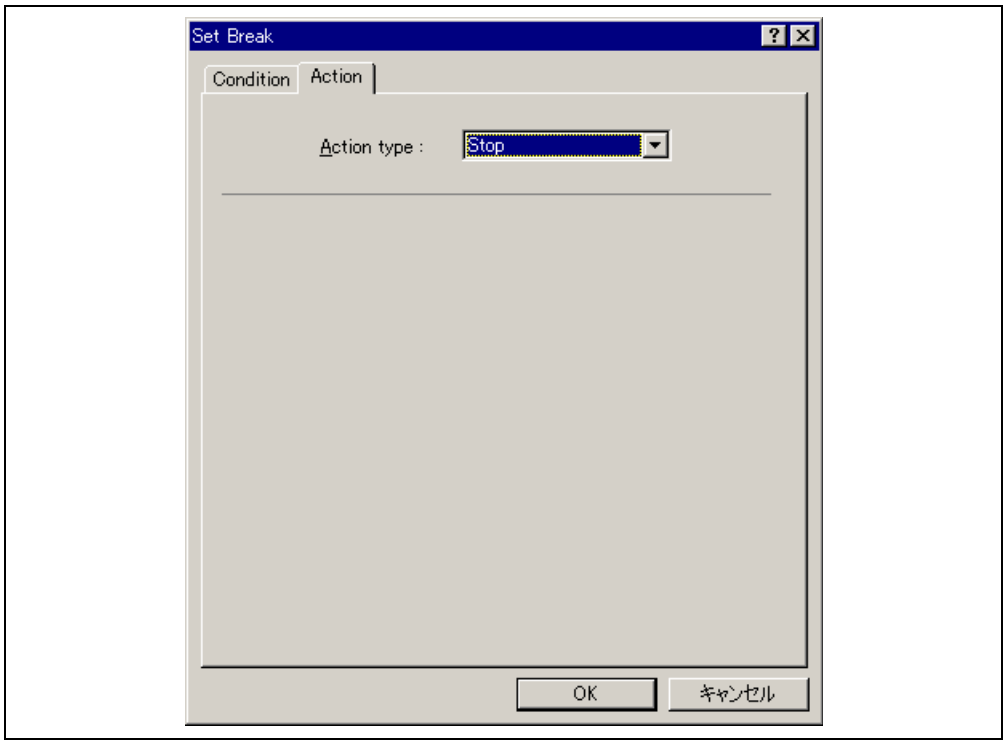


Figure 4.66 Set Break Dialog Box (Action Page)

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

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

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

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

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

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

[Interrupt] Interrupts the program when a condition has been satisfied. For details, refer to section 2.16, Pseudo-Interrupts.

[Interrupt type1]	Specifies the interrupt vector number (when the prefix is omitted, values must be input and are displayed in hexadecimal) (0 to H'FF)
[Priority]	Interrupt priority (when the prefix is omitted, values must be input and are displayed in hexadecimal) (H'0 to H'11) Whether or not the interrupt is accepted is determined by the CPU's specification of the selected debugging platform. However, when H'8 or larger is specified for the priority, interrupts are always accepted.

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

4.15.3 Modifying Breakpoints

Select a breakpoint to be modified, and choose [Edit...] from the pop-up menu to open the [Set Break] dialog box, which allows the user to modify the break conditions. The [Edit...] menu is only available when one breakpoint is selected.

4.15.4 Enabling a Breakpoint

Select a breakpoint and choose [Enable] from the pop-up menu to enable the selected breakpoint.

4.15.5 Disabling a Breakpoint

Select a breakpoint and choose [Disable] from the pop-up menu to disable the selected breakpoint. When a breakpoint is disabled, the breakpoint will remain in the list, but a break will not occur when the specified conditions have been satisfied.

4.15.6 Deleting a Breakpoint

Select a breakpoint and choose [Delete] from the pop-up menu to remove the selected breakpoint. To retain the breakpoint but not have it cause a break when its conditions are met, use the [Disable] option (see section 4.15.5, Disabling a Breakpoint).

4.15.7 Deleting All Breakpoints

Choose [Delete All] from the pop-up menu to remove all breakpoints.

4.15.8 Viewing the Source Line for a Breakpoint

Select a breakpoint and choose [Go to Source] from the pop-up menu to open the [Source] or [Disassembly] window at address of breakpoint. The [Go to Source] menu is only available when one breakpoint is selected.

4.15.9 Closing Input or Output File

Select a breakpoint and choose [Close File] from the pop-up menu to close the selected [File Input] or [File Output] data file and to reset the address to read the file.

4.15.10 Closing All Input and Output Files

Choose [Close All Files] from the pop-up menu to close all [File Input] and [File Output] data files and to reset the address to read the file.

4.16 Analyzing Performance

Use the [Performance Analysis] window to select a function name and analyze the performance.

4.16.1 Opening the Performance Analysis Window

Choose [View -> Performance -> Performance Analysis] or click the [PA] toolbar button  to open the [Performance analysis] window.

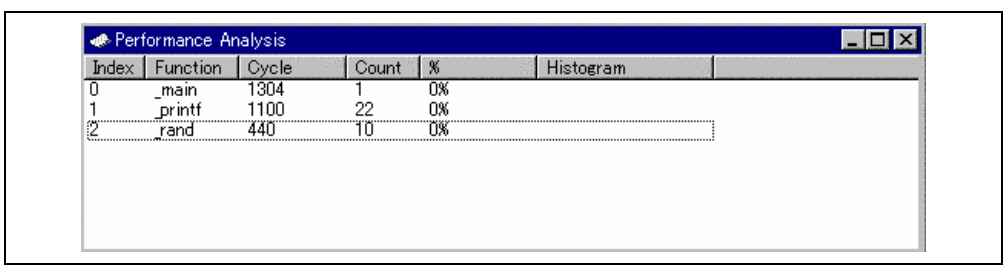


Figure 4.67 Performance Analysis Window

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

The number of execution cycles are calculated as follows:

Execution cycles = total number of execution cycles when execution returns from the function
– total number of execution cycles when the target function is called

The following items are displayed:

[Index] Index number of the set condition

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

[Cycle] Total number of instruction execution cycles

[Count] Total number of calls for the function

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

[Histogram] Histogram display of the above ratio

4.16.2 Specifying a Target Function

After the [Performance Analysis] window is open, choose [Add Range...] from the pop-up menu or press the Insert key to open the [Performance Option] dialog box, which allows the user to specify a function to be analyzed.

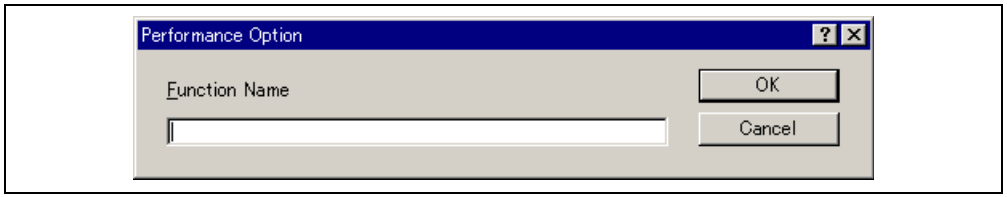


Figure 4.68 Performance Option Dialog Box

This dialog box specifies a function (including a label) to be evaluated. Up to 255 functions can be specified in total. When an overloaded function or a class name including a member function is specified, the [Select Function] dialog box opens. In the dialog box, select a function. For details, refer to section 4.10.3, Supporting Duplicate Labels.

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

Select a function that has been set and choose [Edit Range] from the pop-up menu or press the Enter key to open the [Performance Option] dialog box and to change the function to be evaluated.

4.16.3 Starting Performance Data Acquisition

Choose [Enable Analysis] from the pop-up menu (a check mark is shown to the left of [Enable analysis]) to start acquiring performance analysis data.

4.16.4 Resetting Data

Choose [Reset Counts/Times] from the pop-up menu to clear the current performance analysis data.

4.16.5 Deleting a Target Function

Select a function and choose [Delete Range] from the pop-up menu to delete the selected target function and to recalculate the data for the other ranges. The selected function can also be deleted by the Delete key.

4.16.6 Deleting All Target Functions

Choose [Delete All Ranges] from the pop-up menu to delete all the current target functions and to clear the performance analysis data.

4.17 Acquiring Code Coverage

The [Coverage] window acquires code coverage information (C0 coverage and C1 coverage) in the range specified by the user, and displays the information.

4.17.1 Opening the Coverage Window

Choose [View -> Code -> Coverage...] or click the [Coverage] toolbar button to open the [Open Coverage] dialog box.

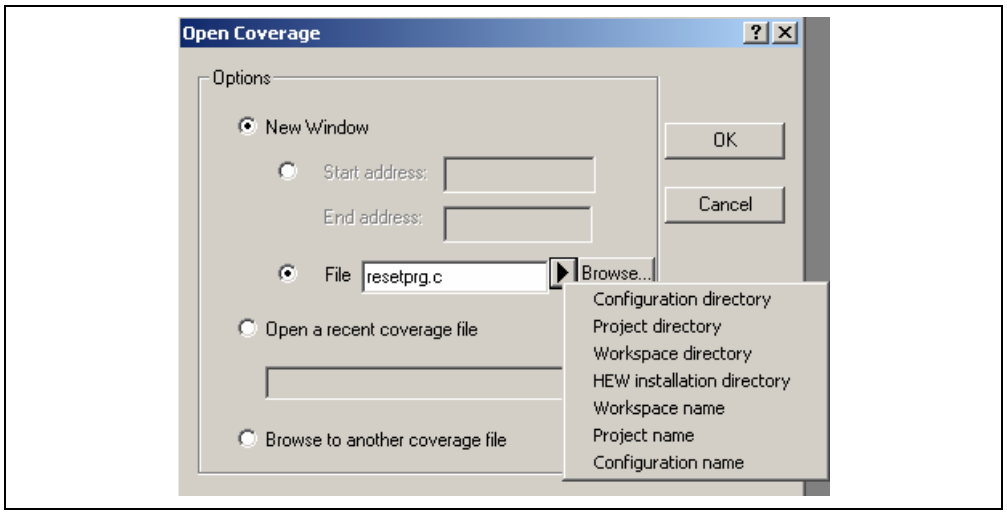


Figure 4.69 Open Coverage Dialog Box

This dialog box specifies the coverage acquiring range. To set coverage for a new range, the following two ways are available:

- Specifying the start and end addresses on the new window

[Start Address] Start address of coverage information display
(When a prefix is omitted, the values is input in hexadecimal.)

[End Address] End address of coverage information display
(When a prefix is omitted, the value is input in hexadecimal.)

- Specifying the file on the new window

[File] Source file whose type number is .C or .CPP in the current project.
Functions in the specified file can be set as the coverage range.
If the type number of the file is omitted, .C is complemented. The file that has other type numbers than .C or .CPP cannot be specified.
A placeholder or the [Browse…] button is available.

To use the settings saved in a coverage information file, choose the file from [Open a recent coverage file], or open a file open dialog box by [Browse to another coverage file] and select the file. When [Open a recent coverage file] is selected, up to four recent files that have been saved are displayed.

Clicking [OK] opens the [Coverage] window.

- Coverage window (specifying address)

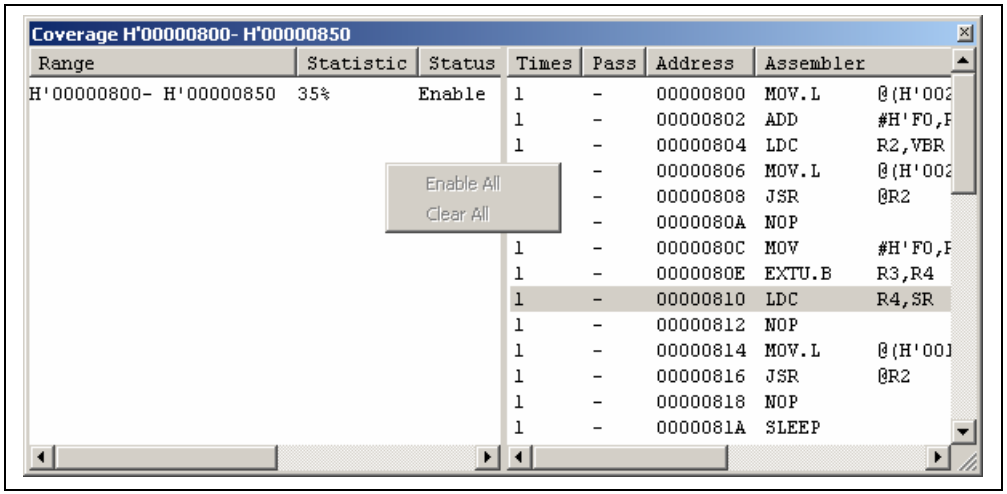


Figure 4.70 Coverage Window (Specifying Address)

This window is divided into two by a splitter.

- Left side

This window displays the coverage range and statistical information. The following item is displayed:

[Range] Address range

[Statistic] Percentage of the instructions executed within the range

[Status] Enable or Disable status of the coverage range

- Right side

This window displays the coverage information of the function in C/C++ and assembly-language level. You can select a function by clicking the function name in left side of the window.

The following items are displayed:

[Times] Number of times instruction was executed

[Pass] Execution status of conditional branch instructions
T: A branch occurred because the condition was satisfied
F: No branch has occurred because the condition was not satisfied

[Address] Instruction Address

[Assembler] Disassembled display

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

When the [Coverage] window is closed, the acquired coverage information and the conditions to acquire information will be cleared.

- Coverage window (specifying source file)

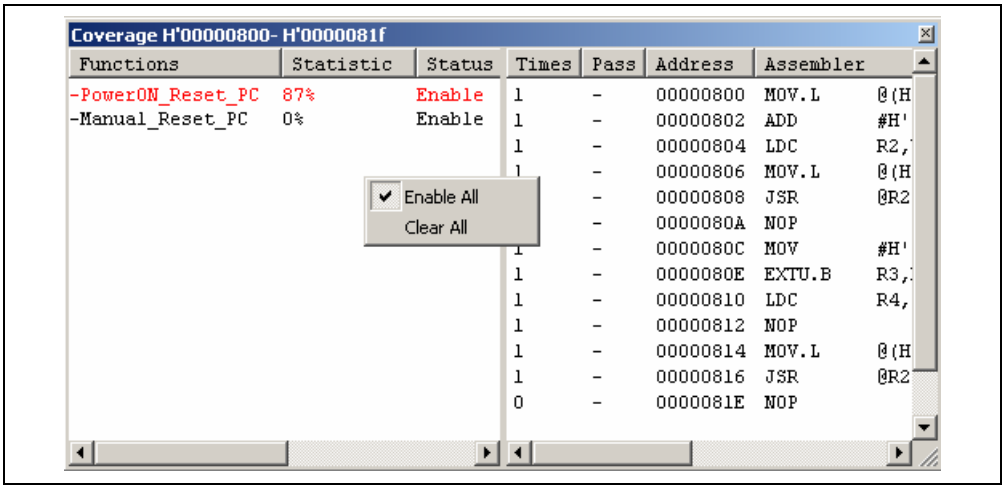


Figure 4.71 Coverage Window (Specifying Source File)

This window is divided into two by a splitter.

- Left side

This window displays the coverage range and statistical information. The following item is displayed:

[Function] List of functions

[Statistic] Percentage of the instruction executed within the function

[Status] Enable or Disable status of the respective function

Note: 1. The functions can be sorted by their names, percentage, status, either in ascending or descending order, by clicking the Column tab ("Functions" or "Statistic" or "Status").
2. By selecting the popup menu item ("Enable All"), all the functions can be set "enable" or "disable".
3. By selecting the popup menu item ("Clear All"), the coverage data of all the functions can be cleared.

- Right side

This window displays the coverage information of the function in C/C++ and assembly-language level. You can select a function by clicking the function name in left side of the window.

The following items are displayed:

[Times] Number of times instruction was executed

[Pass] Execution status of conditional branch instructions
T: A branch occurred because the condition was satisfied
F: No branch has occurred because the condition was not satisfied

[Address] Instruction Address

[Assembler] Disassembled display

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

When the [Coverage] window is closed, the acquired coverage information and the conditions to acquire information will be cleared.

4.17.2 Acquiring Coverage Information

Select [Enable Coverage] from the pop-up menu and execute the user program to acquire coverage information.

4.17.3 Viewing the Source Window

Choose [View Source] from the pop-up menu to open the [Editor] window and to display the [Editor] window corresponding to the cursor location in the [Coverage] window.

4.17.4 Changing the Display Address

Choose [Go to Address...] to open the [Go To Address] dialog box.

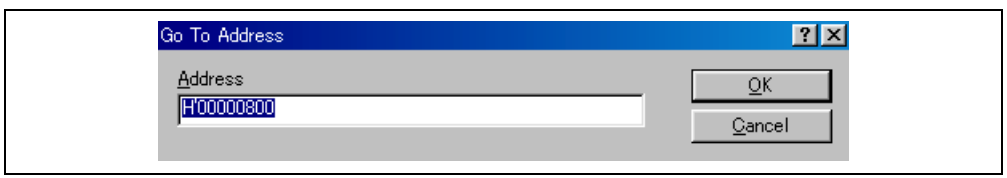


Figure 4.72 Go To Address Dialog Box

This dialog box changes the address displayed in the [Coverage] window.

4.17.5 Changing the Coverage Range

- Specifying the coverage range with an address

Choose [Set Range...] from the pop-up menu to open the [Coverage Range] dialog box.

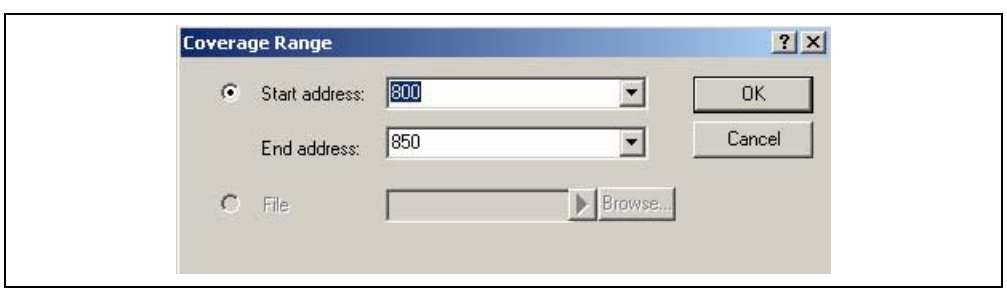


Figure 4.73 Coverage Range Dialog Box (Specifying Address)

This dialog box specifies the condition to acquire instruction execution information. The following items can be specified.

[Start Address] Start address (When a prefix is omitted, the value is input in hexadecimal.)

[End Address] End address (When a prefix is omitted, the value is input in hexadecimal.)

Clicking [OK] changes the coverage range.

- Specifying the coverage range with a source file

Choose [Set Range...] from the pop-up menu to open the [Coverage Range] dialog box.

Figure 4.74 Coverage Range Dialog Box (Specifying Source File)

This dialog box specifies the condition to acquire instruction execution information. The following items can be specified.

[File] Source file whose type number is .C or .CPP in the current project.
Functions in the specified file can be set as the coverage range.
If the type number of the file is omitted, .C is complemented. The file that has other type numbers than .C or .CPP cannot be specified.
A placeholder or the [Browse…] button is available.

Clicking [OK] changes the coverage range.

4.17.6 Clearing Coverage Information

Choose [Clear Data] from the pop-up menu to clear the acquired coverage information.

4.17.7 Saving Coverage Information in a File

Choose [Save Data...] from the pop-up menu to open the [Save Data] dialog box, which allows the user to save the coverage information in a file.

Figure 4.75 Save Data Dialog Box

This dialog box specifies the location and name of a coverage information file to be saved. The placeholder or the [Browse...] button can be used.

If a file name extension is omitted, .COV is automatically added. If a file name extension other than .COV or .TXT is specified, an error message will be displayed.

4.17.8 Loading Coverage Information from a File

Choose [Load Data...] from the pop-up menu to open the [Load Data] dialog box, which allows the user to load the coverage information from a file.

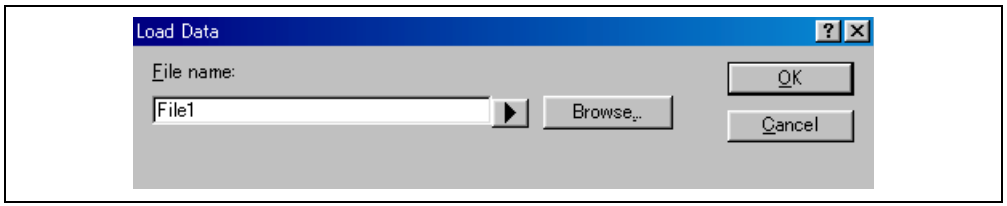


Figure 4.76 Load Data Dialog Box

This dialog box specifies the location and name of a coverage information file to be loaded. The placeholder or the [Browse...] button can be used.

Only .COV files can be loaded. If a file name extension other than .COV is specified, an error message will be displayed.

Note: If the coverage range is specified by a source file, the saved .COV file cannot be loaded.

4.17.9 Updating the Information

Choose [Refresh] from the pop-up menu to update the [Coverage] window to the latest information.

4.17.10 Stopping Update

Choose [Lock Refresh] from the pop-up menu to only update [Times] and [Pass] when program execution stops. Memory accesses to update instruction codes in the [Coverage] window are stopped.

4.17.11 Confirmation Request Dialog Box

A confirmation request dialog box will appear when [Clear Data] or [Set Range...] is clicked or an attempt is made to close the [Coverage] window.

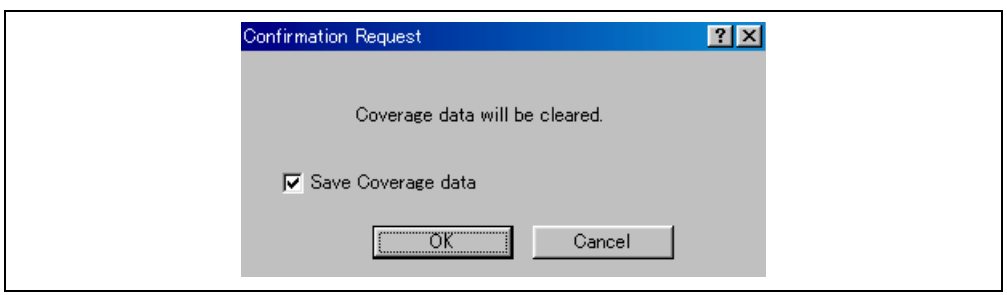


Figure 4.77 Confirmation Request Dialog Box

Clicking [OK] clears the coverage data. Choosing [Save Coverage data] opens the [Save Coverage Data] dialog box (figure 4.78) to save the coverage data in a file before it is cleared.

4.17.12 Save Coverage Data Dialog Box

When [File -> Save Session] menu option is clicked, the [Save Coverage Data] dialog box will appear, which allows the user to save the [Coverage] window data in separate files or a single file.

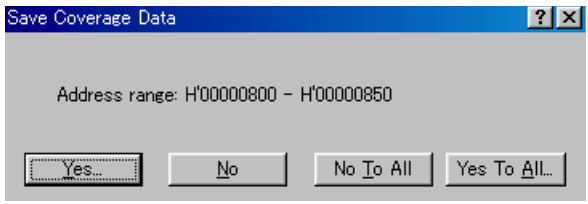


Figure 4.78 Save Coverage Data Dialog Box

When multiple [Coverage] windows are open, a [Save Coverage Data] dialog box will appear for each open coverage window.

Clicking the [No To All] button closes the dialog box without saving any coverage data.

Clicking the [Yes To All] button saves the data of all [Coverage] windows in a single file.

Note: If the file is specified for the coverage range, not all [Coverage] windows can be saved in a single file.

4.17.13 Displaying the Coverage Information in the Editor Window

The coverage information is reflected to the [Editor] window by highlighting the debugger columns corresponding to the source lines of executed instructions. When the coverage settings are modified in the [Coverage] window, the debugger column display will be updated.

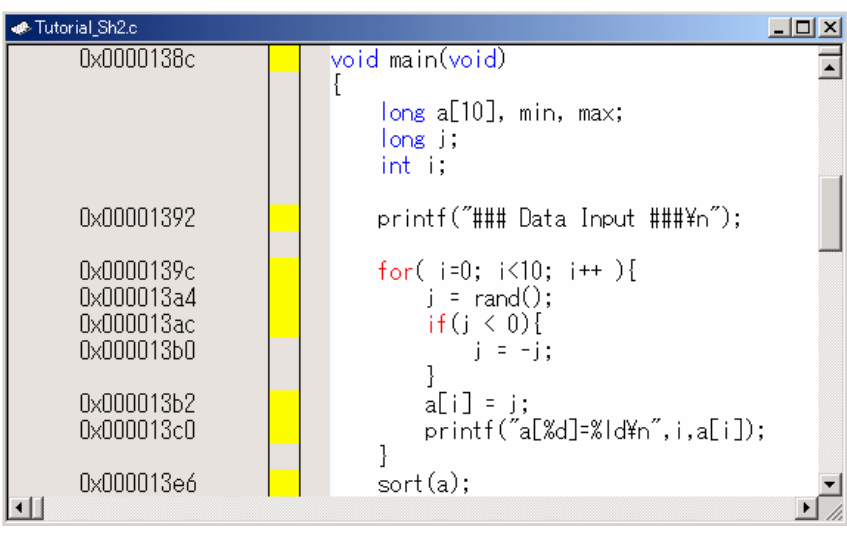


Figure 4.79 Debugger Column (Coverage)

4.18 Viewing the Current Status

Choose [View -> CPU -> Status] or click the [View Status] toolbar button to open the [Status] window and see the current status of the debugging platform.

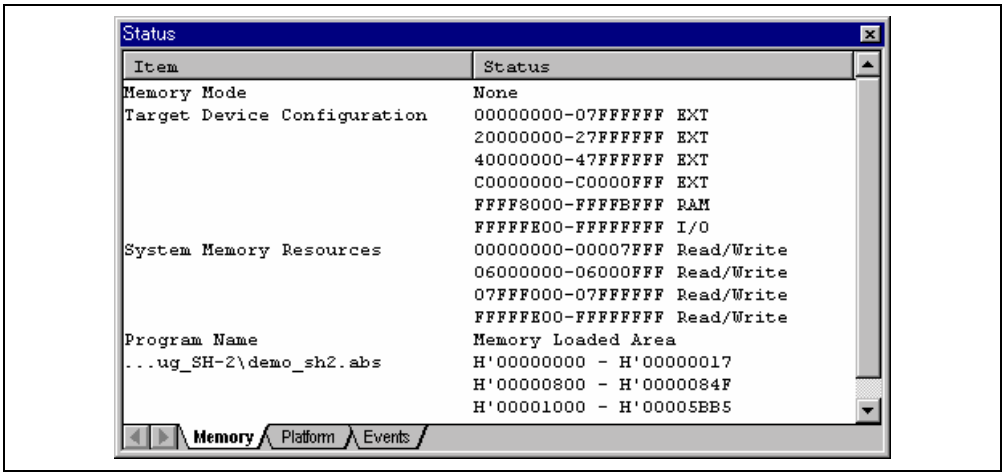


Figure 4.80 Status Window

The [Status] window is split into three sheets:

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

4.19 Debugging with the Command Line Interface

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

4.19.1 Opening the Command Line Window

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

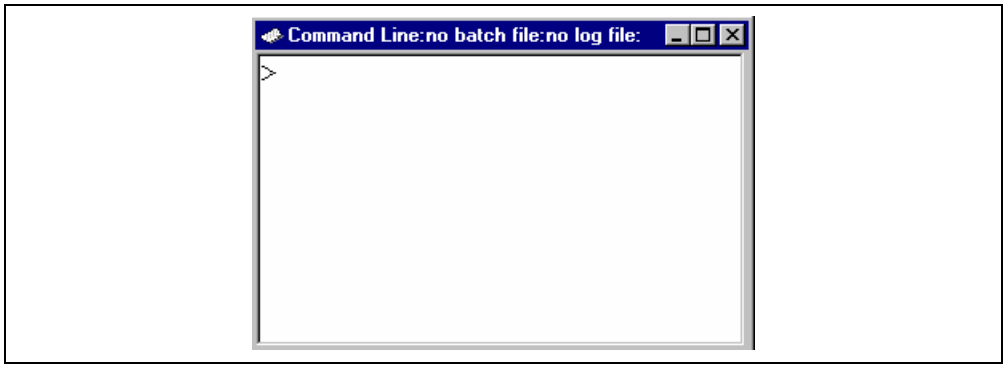


Figure 4.81 Command Line Window

This window allows the user to control the debugging platform by sending text-based commands. A series of predefined command lines can be called from a file and the output can be recorded in a file. The command can be executed by pressing the Enter key after the command is input at the prompt (>) on the last line. For information about the available commands, refer to section 5, Command Lines, and the on-line help.

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

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

The HEW command and TCL (ver. 8.4.1) commands can be input in this window.

4.19.2 Specifying a Command File

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

Choose [Set Batch File...] from the pop-up menu to open the [Set Batch File] dialog box, in which the name of a command file (*.hdc) can be specified. Clicking the [OK] button displays the specified command file name as the window title. Clicking the [Cancel] button closes the dialog box without modifying the setting.

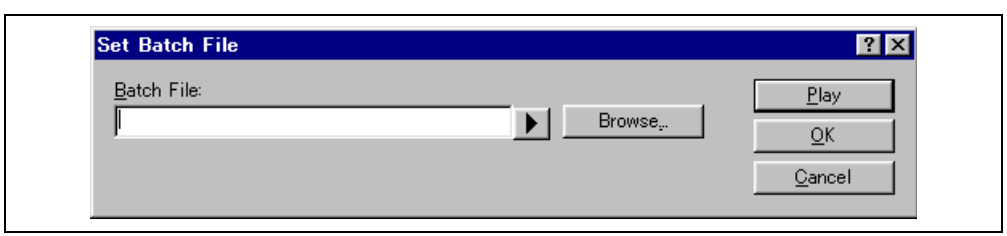


Figure 4.82 Set Batch File Dialog Box

4.19.3 Executing a Command File

Click the [Play] button in the [Set Batch File] dialog box or choose [Play] from the pop-up menu to execute the command file. The [Play] menu is displayed in gray while the file is running and can be used when the command file execution stops and control returns to the user.

4.19.4 Stopping Command Execution

Choose [Stop] from the pop-up menu to stop command execution. The [Stop] menu becomes valid during command execution.

4.19.5 Specifying a Log File

Choose [Set Log File...] from the pop-up menu to open the [Open Log File] dialog box, in which a log file to store the command execution results can be specified.

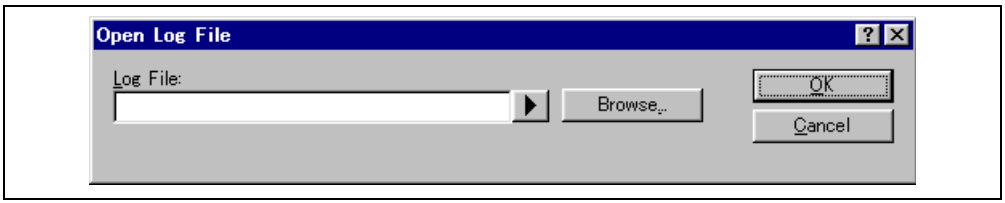


Figure 4.83 Open Log File Dialog Box

Enter the name of a log file (*.log). The logging option is automatically set and the name of the file is shown on the window title bar.

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

4.19.6 Starting or Stopping Logging

Choose [Logging] from the pop-up menu to toggle logging to file on and off. When logging is active, the button becomes effective. Note that the contents of the log file cannot be viewed until logging is completed, or temporarily disabled by clearing the check box. Re-enabling logging will append to the log file.

4.19.7 Entering a Full Path to the File

It is recommended that the full path to a file is specified as a file name in the [Command line] window because the current directory can be moved. However, care must be taken to enter the correct full path to a file when it is entered from the keyboard. To save this trouble, a full path can be easily specified by browsing through files.

Choose [Browse...] from the pop-up menu to open the [Browse] dialog box. Select a file and click [Open] to paste the full path to the selected file to the cursor location. This option can only be used when the cursor is located on the last line.

4.19.8 Pasting a Placeholder

Select a placeholder from the [Placeholder] submenu in the pop-up menu to paste the selected placeholder to the cursor location. This function is only available when the cursor is located on the last line.

4.20 Generating a Pseudo-Interrupt Manually

Choose [View -> CPU -> Trigger] or click the [Trigger] toolbar button to open the [Trigger] window, which allows the user to generate a pseudo-interrupt manually by pressing a button on the window.

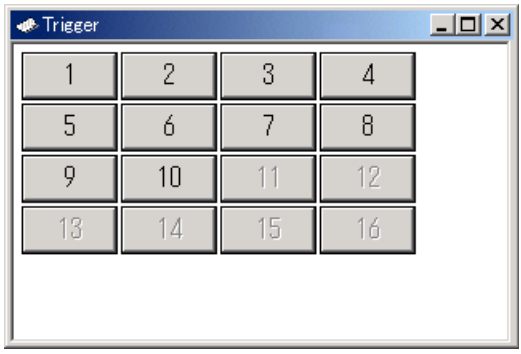


Figure 4.84 Trigger Window

This window displays trigger buttons that generate pseudo-interrupts manually. The details of the interrupt to be generated by pressing each trigger button can be specified in the [Trigger Setting] dialog box.

Up to 256 trigger buttons can be used.

For details on the interrupt processing in the simulator/debugger, refer to section 2.16, Pseudo-Interrupts.

4.20.1 Setting a Trigger Button

Choose [Setting...] from the pop-up menu to open the [Trigger Setting] dialog box and to specify the details of the pseudo-interrupt to be generated by pressing each trigger button.

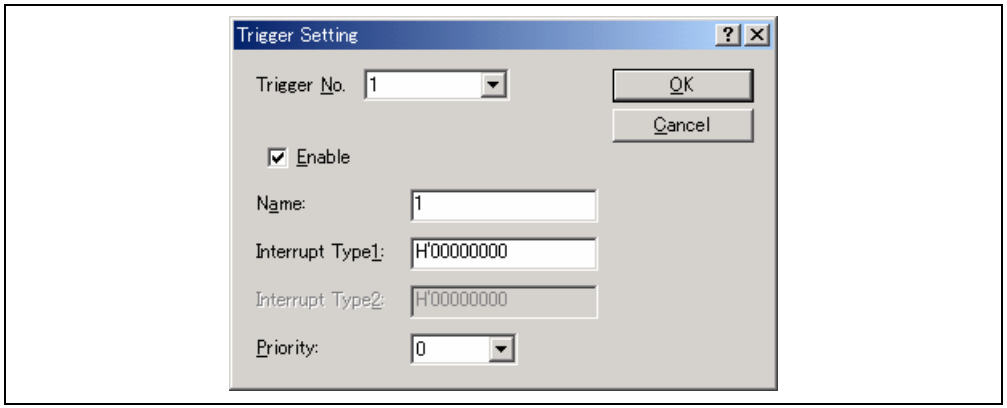


Figure 4.85 Trigger Setting Dialog Box

This dialog box allows the user to specify the details of the pseudo-interrupt to be generated by pressing each trigger button.

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

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

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

[Interrupt Type1] Specifies the Interrupt vector number. (0 to H'FF)

[Priority] Interrupt priority. (0 to H'11)
Whether or not the interrupt is accepted is determined by the CPU's specification of the selected debugging platform. However, when H'8 or larger is specified for the priority, interrupts are always accepted.

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

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

4.20.2 Changing the Number of Trigger Buttons

Specify the number of trigger buttons displayed in the [Trigger] window in the [Number of Buttons] submenu in the pop-up menu. [4], [16], [64], or [256] can be selected.

4.20.3 Changing the Size of Trigger Buttons

Specify the size of trigger buttons displayed in the [Trigger] window in the [Size] submenu in the pop-up menu. [Large], [Normal], or [Small] can be selected.

4.21 Modifying the Simulator/Debugger Settings

This section describes how to modify the simulator system, memory map, and memory resource settings after the simulator/debugger is started.

4.21.1 Modifying the Simulator System

The [System] tab in the [Simulator System] dialog box modifies the system call start location, execution mode, and floating-point rounding mode.

Choose [Options -> Simulator -> System...] or click the [Simulator System] toolbar button to open the [System] tab in this dialog box.

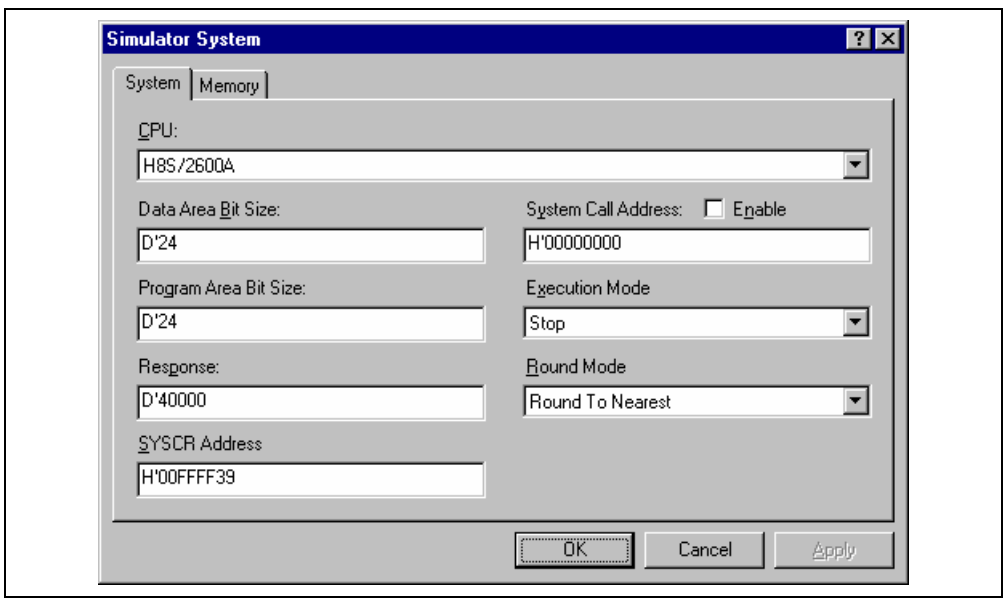


Figure 4.86 Simulator System Dialog Box (System Tab)

The following items can be specified in this dialog box:

[CPU] Displays the current CPU. (The CPU must be specified in the [Debug Settings] dialog box.

[Data Area Bit size] Specifies the number of bits in the data space. The following values can be set in each CPU:
H8/300, H8/300L, H8/300HN, and H8S/2600N: 16
H8S/2000N:
H8/300HA: 17 to 24
H8S/2600A and H8S/2000A: 17 to 32

[Program Area Bit size] Specifies the number of bits in the program area. The following values can be set in each CPU:
H8/300, H8/300L, H8/300HN, and H8S/2600N: 16
H8S/2000N:
H8/300HA: Same as the number of address space bits
H8S/2600A and H8S/2000A: 17 to 32

[SYSCR Address] Specifies the SYSCR address.

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

[Response] Specifies the window refresh timing, that is, how many instructions should be executed between refresh actions (1 to D'65535).

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

[Round Mode] Specifies the rounding mode for floating-point decimal-to-binary conversion.
[Round to nearest] Rounds to the nearest value.
The denormalized number is not converted to 0.
[Round to zero] Rounds toward zero.
The denormalized number is converted to 0.

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

4.21.2 Modifying the Memory Map and Memory Resource Settings

The [Memory] tab in the [Simulator System] dialog box sets and modifies the memory map and memory resource.

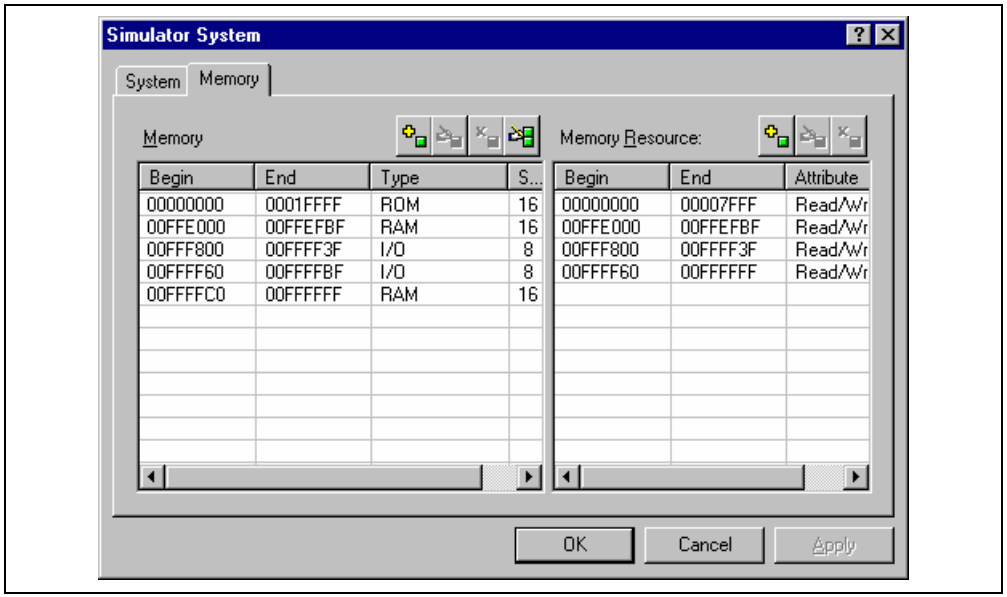


Figure 4.87 Simulator System Dialog Box (Memory Tab)

The following items can be specified in this dialog box:

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

[Memory Resource] Displays the access type and start and end addresses of the current memory resource.

[Memory Resource] can be added, modified, or deleted using the following buttons:

Adds [Memory Resource] items. Clicking this button opens the [Set Memory Resource] dialog box, and memory map items can be specified.

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

Deletes [Memory Resource] items. Select an item to be deleted in the list box and click this button.

[Memory Resource] is the same setting information as that of [Memory Resource] of the [Simulator] sheet in the [H8S, H8/300 Standard Toolchain] dialog box. Modifications are reflected on both items. For the [H8S, H8/300 Standard Toolchain] dialog box, refer to section 3.3.1, Memory Map.

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

Adds [Memory Map] items. Clicking this button opens the [Set Memory Map] dialog box (figure 4.88), and memory map items can be specified.

Modifies [Memory Map] items. Select an item to be modified in the list box and click this button. The [Set Memory Map] dialog box (figure 4.88) opens and memory map items can be modified.

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

[Memory Map] and [Memory Resource] can be reset to the default value by the button. Clicking the [OK] or [Apply] button stores the modified settings. Clicking the [Cancel] button closes this dialog box without modifying the settings.

4.21.3 Set Memory Map Dialog Box

The [Set Memory Map] dialog box specifies the memory map of the target CPU.

The contents displayed in this dialog box depend on the target CPU. The simulator/debugger uses the specified data to calculate the number of cycles for memory accesses.

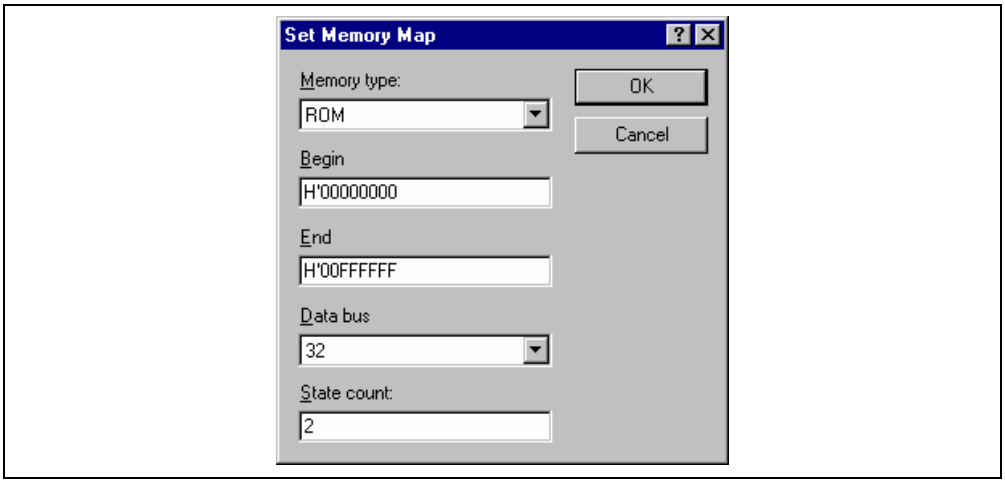


Figure 4.88 Set Memory Map Dialog Box

The following items are specified:

[Memory type] Memory type
[ROM] Internal ROM
[RAM] Internal RAM
[EXT] External memory
[IO] Internal I/O
[EEPROM] EEPROM

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

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

[Data bus] Memory data bus width

[State count] Number of memory access cycles

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

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

4.21.4 Set Memory Resource Dialog Box

The [Set Memory Resource] dialog box sets and modifies memory resources.

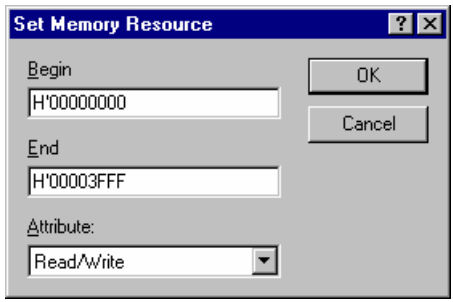


Figure 4.89 Set Memory Resource Dialog Box

The following items are specified:

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

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

[Attribute] Access type

[Read]	Read only
[Write]	Write only
[Read/Write]	Read and write

Click the [OK] button after specifying the [Begin], [End], and [Attribute]. Clicking the [Cancel] button closes this dialog box without modifying the setting.

4.22 Standard I/O and File I/O Processing

Use the [Simulated I/O] window to enable the standard I/O and file I/O system calls from the user program.

4.22.1 Opening the Simulated I/O Window

Choose [View -> CPU -> Simulated I/O] or click the [Simulated I/O] toolbar button  to open the [Simulated I/O] window.

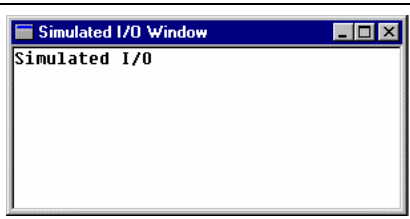


Figure 4.90 Simulated I/O Window

The standard output from the user program is displayed in this window. The key input from this window is handled as the standard input to the user program.

4.22.2 I/O Functions

Table 4.1 lists the supported I/O functions.

Table 4.1 I/O Functions

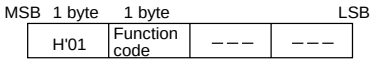
No.	Function Code	Function Name	Description
1	H'01 (16-bit address) H'11 (24-bit address) H'21 (32-bit address)	GETC	Inputs one byte from the standard input device
2	H'02 (16-bit address) H'12 (24-bit address) H'22 (32-bit address)	PUTC	Outputs one byte to the standard output device
3	H'03 (16-bit address) H'13 (24-bit address) H'23 (32-bit address)	GETS	Inputs one line from the standard input device
4	H'04 (16-bit address) H'14 (24-bit address) H'24 (32-bit address)	PUTS	Outputs one line to the standard output device
5	H'05 (16-bit address) H'15 (24-bit address) H'25 (32-bit address)	FOPEN	Opens a file
6	H'06	FCLOSE	Closes a file
7	H'07(16-bit address) H'17 (24-bit address) H'27 (32-bit address)	FGETC	Inputs one byte from a file
8	H'08 (16-bit address) H'18 (24-bit address) H'28 (32-bit address)	FPUTC	Outputs one byte to a file
9	H'09 (16-bit address) H'19 (24-bit address) H'29 (32-bit address)	FGETS	Inputs one line from a file
10	H'0A (16-bit address) H'1A (24-bit address) H'2A (32-bit address)	FPUTS	Outputs one line to a file
11	H'0B	FEOF	Checks for end of the file
12	H'0C	FSEEK	Moves the file pointer
13	H'0D	FTELL	Returns the current position of the file pointer

To perform I/O processing, use the [System Call Address] in the [Simulator System] dialog box (section 4.21.1) in the following procedure.

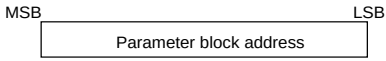
1. Set the address specialized for I/O processing in the [System Call Address], select [Enable] and execute the program.
2. When detecting a subroutine call instruction (BSR, JSR, or BSRF), that is, a system call to the specialized address during user program execution, the simulator/debugger performs I/O processing by using the R0 and R1 values (H8/300, H8/300L series) or the ER1 value (H8/300H, H8S series) as the parameters.

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

- Set the function code (table 4.1) to the R0 register



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



- Reserve the parameter block and input/output buffer areas

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

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

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

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

Parameter Block

(5)

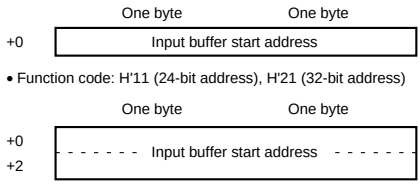
Parameters

(6)

(1) Number corresponding to table 4.1
(2) Function name
(3) Function code
(4) I/O overview
(5) I/O parameter block
(6) I/O parameters

1	GETC H'01, H'11, H'21	Inputs one byte from the standard input device

Parameter Block



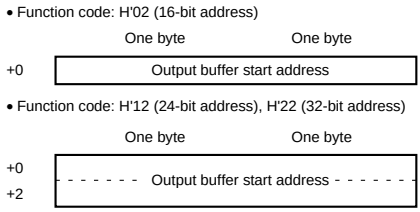
Parameters

- Input buffer start address (input)

Start address of the buffer to which the input data is written to.

2	PUTC H'02, H'12, H'22	Outputs one byte to the standard output device

Parameter Block

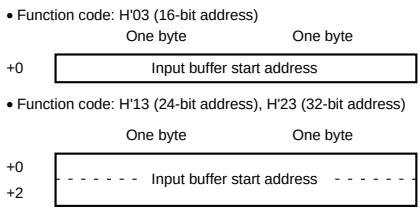


Parameters

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

3	GETS H'03, H'13, H'23	Inputs one line from the standard input device

Parameter Block

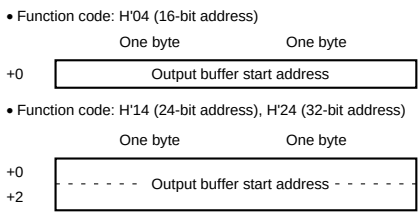


Parameters

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

4	PUTS H'04, H'14, H'24	Outputs one line to the standard output device

Parameter Block



Parameters

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

5	FOPEN	Opens a file
	H'05, H'15, H'25	

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

Parameter Block

• Function code: H'05 (16-bit address)

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

• Function code: H'15 (24-bit address), H'25 (32-bit address)

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

Parameters

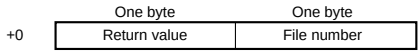
- Return value (output)
  0: Normal completion
  –1: Error
- File number (output)
  The number to be used in all file accesses after opening.
- Open mode (input)
  H'00: "r"
  H'01: "w"
  H'02: "a"
  H'03: "r+"
  H'04: "w+"
  H'05: "a+"
  H'10: "rb"
  H'11: "wb"
  H'12: "ab"
  H'13: "r+b"
  H'14: "w+b"
  H'15: "a+b"
  These modes are interpreted as follows.
  "r": Open for reading.
  "w": Open an empty file for writing.
  "a": Open for appending (write starting at the end of the file).
  "r+": Open for reading and writing.
  "w+": Open an empty file for reading and writing.
  "a+" : Open for reading and appending.

"b" : Open in binary mode.

- Start address of file name (input)
  The start address of the area for storing the file name.

6	FCLOSE	Closes a file
	H'06	

Parameter Block



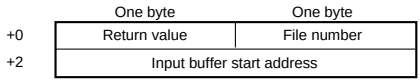
Parameters

- Return value (output)
  0: Normal completion
  –1: Error
- File number (input)
  The number returned when the file was opened.

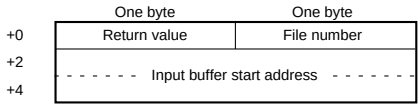
7	FGETC	Inputs one byte from a file
	H'07, H'17, H'27	

Parameter Block

• Function code: H'07 (16-bit address)



• Function code: H'17 (24-bit address), H'27 (32-bit address)



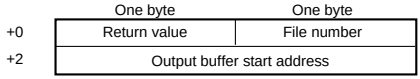
Parameters

- Return value (output)
  0: Normal completion
  –1: EOF detected
- File number (input)
  The number returned when the file was opened.
- Start address of input buffer (input)
  The start address of the buffer for storing input data.

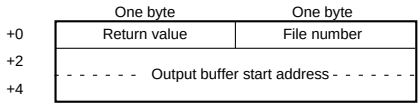
8	FPUTC	Outputs one byte to a file
	H'08, H'18, H'28	

Parameter Block

• Function code: H'08 (16-bit address)



• Function code: H'18 (24-bit address), H'28 (32-bit address)



Parameters

- Return value (output)
  0: Normal completion
  –1: Error
- File number (input)
  The number returned when the file was opened.
- Start address of output buffer (input)
  The start address of the buffer used for storing the output data.

9	FGETS	Reads character string data from a file
	H'09, H'19, H'29	

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

Parameter Block

• Function code: H'09 (16-bit address)

	One byte	One byte
+0	Return value	File number
+2	Buffer size	
+4	Input buffer start address	

• Function code: H'19 (24-bit address), H'29 (32-bit address)

	One byte	One byte
+0	Return value	File number
+2	Buffer size	
+4	Input buffer start address	
+6		

Parameters

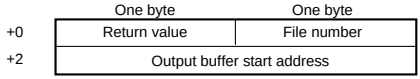
- Return value (output)
  0: Normal completion
  –1: EOF detected
- File number (input)
  The number returned when the file was opened.
- Buffer size (input)
  The size of the area for storing the read data. A maximum of 256 bytes can be stored.
- Start address of input buffer (input)
  The start address of the buffer for storing input data.

10	FPUTS	Writes character string data to a file
	H'0A, H'1A, H'2A	

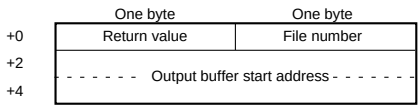
Writes character string data to a file. The NULL code that terminates the character string is not written to the file.

Parameter Block

• Function code: H'0A (16-bit address)



• Function code: H'1A (24-bit address), H'2A (32-bit address)



Parameters

- Return value (output)
  0: Normal completion
  –1: Error
- File number (input)
  The number returned when the file was opened.
- Start address of output buffer (input)
  The start address of the buffer used for storing the output data.

11	FEOF	Checks for end of file
	H'0B	

Parameter Block

	One byte	One byte
+0	Return value	File number

Parameters

- Return value (output)
  0: File pointer is not at EOF
  –1: EOF detected
- File number (input)
  The number returned when the file was opened.

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

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Derection	Unused
+4	Offset upper word	
+6	Offset lower word	

Parameters

- Return value (output)
  0: Normal completion
  –1: Error
- File number (input)
  The number returned when the file was opened.
- Direction (input)
  0: The offset specifies the position as a byte count from the start of the file.
  1: The offset specifies the position as a byte count from the current file pointer.
  2: The offset specifies the position as a byte count from the end of the file.
- Offset (input)
  The byte count from the location specified by the direction parameter.

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

Parameter Block

	One byte	One byte
+0	Return value	File number
+2	Offset upper word	
+4	Offset lower word	

Parameters

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

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

```
            MOV.W     #H'0101,R0
            MOV.W     #PARM,R1
            JSR       @SYS_CALL
STOP        NOP
SYS_CALL    NOP
PARM        .DATA.W   INBUF
INBUF       .RES.B    2
            .END
```

4.23 Synchronizing Multiple Debugging Platforms

Multiple debugging platforms can be operated at the same time in the HEW. There are two methods available to achieve this. These are outlined below; the external method was available in HEW 2.x. The Internal synchronization of debugger targets is the new preferred method for multiple target debugging in HEW 3.0.

4.23.1 External HEW synchronization

Initiating a HEW from another HEW synchronizes multiple debugging platforms. The HEW that initiates another HEW is called the master, and the initiated HEW is called the slave. Choose [Tools -> Launch Slave HEW...] or click the [Launch Slave HEW] toolbar button  to initiate a slave HEW.

The slave HEW has the same functionality as the master HEW.

The slave HEW is notified of the following actions done in the master HEW to ensure synchronization of the slave HEW and the master HEW.

- Reset go
- Go
- Stop debugging

Note: The master HEW can initiate multiple slave HEW applications, but slave HEW applications cannot be nested (no slave HEW can initiate another slave HEW).

4.23.2 Internal HEW synchronization

The HEW also supports internal multiple target debugging. This will allow you to connect to multiple target components in the same HEW application. These targets can then be debugged simultaneously. The system allows the user to setup a number of sessions with different targets. Then when debugging the sessions can be synched so that certain events in one session can trigger the same events in the others. This is very similar to that seen in the External HEW synchronization section above. This facility though has the added advantage that it is easy to swap sessions and see what is happening in the same application.

➲ To setup internal HEW synchronization:

1. Select the **[Options->Debug Sessions…]** menu item.
2. Select the "Synchronized Debug" tab of the dialog. The dialog displayed in figure 4.91 is displayed.
3. Select the sessions you wish to synchronize. All currently available sessions in the workspace are displayed.
4. Click the synchronize session button. The icons for these sessions should changed to checked rather than unchecked.
5. Click the Enable synchronized debugging check box to switch this facility on.
6. Click OK to verify the changes.

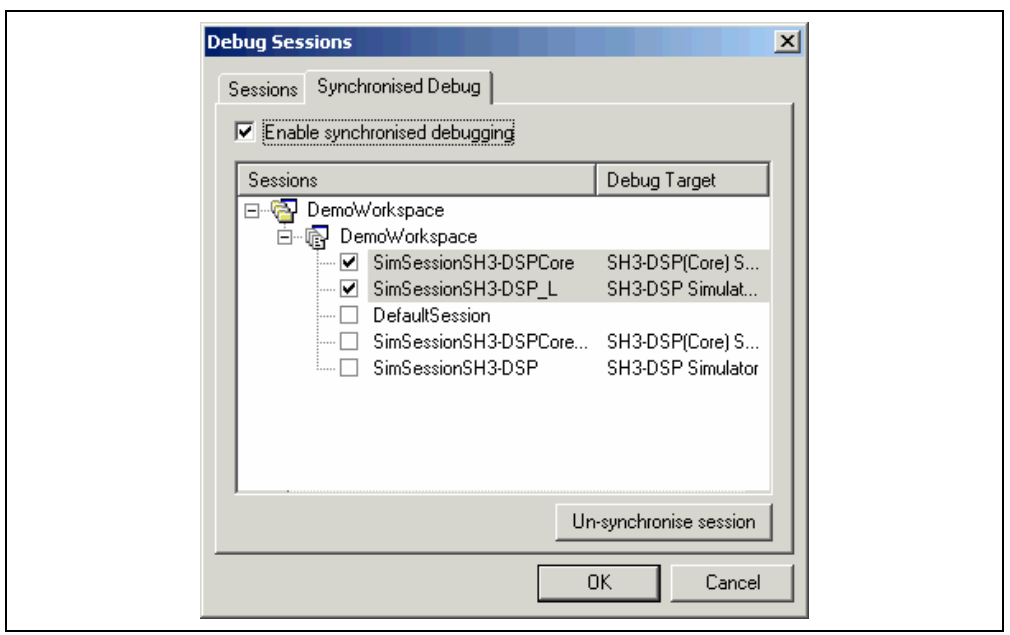


Figure 4.91 Simulator Memory Resource Dialog Box

➲ To use internal HEW synchronization:

1. Follow the options as specified in "To setup internal HEW synchronization".
2. Click on the session combo-box located on the standard toolbar. Select the Synch. Debug selection. This is shown in figure 4.92. This option is only available when synchronized debugging has been added to the system.

3. Once selected the HEW debugger enables the Synchronized debugging facilities. This means the addition of another toolbar this is named Sync. session. This can also be viewed in Figure 4.92.
4. The enable disable toolbar button on the Synch. session toolbar allows you to temporarily switch off the synchronization without losing your settings for this feature.
5. When enabled changing the session in the Sync. session combo box changes the session you are currently viewing. In the normal HEW debugging state this would mean the session is closed. In Synch. session you can have multiple sessions open and the currently selected one on this toolbar is the session you are viewing.
6. This system allows you to debug multiple targets or CPU cores simultaneously. Changing the session changes the views you can see on the screen and the data displayed on these views.

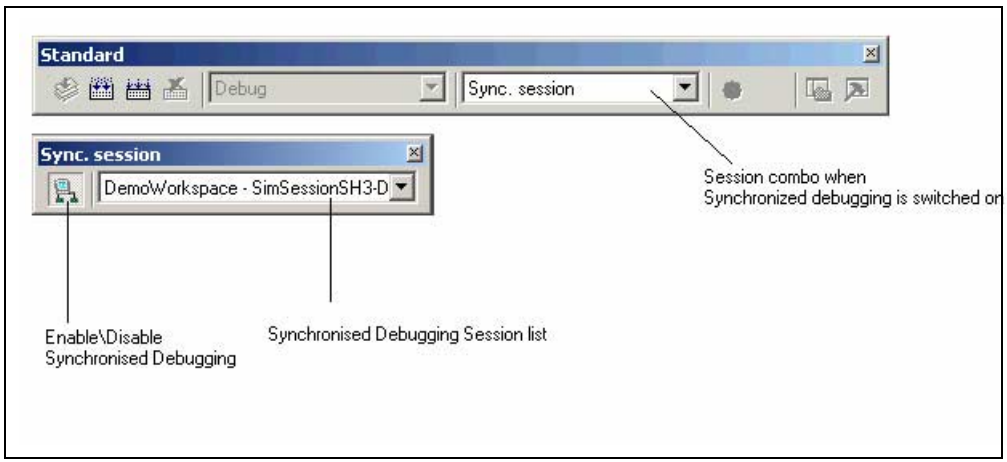


Figure 4.92 Synchronized debug toolbars

Note: There are a number of capabilities that are synchronized when this facility is enabled. The following tables display the capabilities when synchronized debugging is switched on or off. The example shows what happens in the two synchronized sessions.

Debugging function	Target Debugger Session 1	Target Debugger Session 2
User click "GO" in any Session	"GO"	"GO"
User click "STEP Into/out/over" in any Session	"Step"	"Step"
User click "ESC" in any Session	"BREAK"	"BREAK"
	"BREAK" by breakpoints, illegal access or due to illegal User Program	Stop running (Same Effect of pressing ESC)
	Stop running (Same Effect of pressing ESC)	"BREAK" by breakpoints, illegal access or due to illegal User Program

RESET CPU in any Session	RESET CPU	RESET CPU

Figure 4.93 Synchronized debugging on

Debugging function	Target Debugger Session 1	Target Debugger Session 2
User click "GO" in Session 1	"GO"	No activities, "GO" can only be executed manually by User
User click "GO" in Session 2	No activities, "GO" manually executed by User	"GO"
User click "STEP" in session 1	"STEP"	No activities, must "STEP" manually by User
User click "STEP" in session 2	No activities, must "STEP" manually by User	"STEP"
User click "ESC" in session 1	"BREAK"	Still executing, if previously executing User Target Program
User click "ESC" in session 2	Still executing, if previously executing User Target Program	"BREAK"
	"BREAK" by breakpoints, illegal access or due to illegal User Program	Still executing, if previously executing User Target Program
	Still executing, if previously executing User Target Program	"BREAK" by breakpoints, illegal access or due to illegal User Program
RESET CPU in session 1	RESET CPU	No activities
RESET CPU in session 2	No activities	RESET CPU

Figure 4.94 Synchronized debugging off

Note: Another difference to the standard debugging system is that it is possible to view the download modules in the workspace window for all synchronized sessions. This allows modules to be downloaded for any of the available sessions easily.

Section 5 Command Lines

Table 5.1 lists the commands.

Table 5.1 Simulator/Debugger Commands

No.	Command Name	Abbreviation	Function
1	!	-	Comment
2	ADD_FILE	AF	Adds a file to the current project
3	ANALYSIS	AN	Enables or disables performance analysis
4	ANALYSIS_RANGE	AR	Sets or displays performance analysis functions
5	ANALYSIS_RANGE_ DELETE	AD	Deletes a performance analysis range
6	ASSEMBLE	AS	Assembles instructions into memory
7	ASSERT	-	Checks if an expression is true or false
8	BREAKPOINT	BP	Sets a breakpoint at an instruction address
9	BREAK_ACCESS	BA	Specifies a memory range access as a break condition
10	BREAK_CLEAR	BC	Deletes breakpoints
11	BREAK_CYCLE	BCY	Specifies a cycle as a break condition
12	BREAK_DATA	BD	Specifies a memory data value as a break condition
13	BREAK_DISPLAY	BI	Displays a list of breakpoints
14	BREAK_ENABLE	BE	Enables or disables a breakpoint
15	BREAK_REGISTER	BR	Specifies a register data as a break condition
16	BREAK_SEQUENCE	BS	Sets sequential breakpoints
17	BUILD	BU	Performs a build on the current project
18	BUILD_ALL	BL	Performs a build all on the current project
19	CACHE	-	Sets caching on or off
20	CHANGE_CONFIGURA TION	CC	Sets the current configuration
21	CHANGE_PROJECT	CP	Sets the current project
22	COVERAGE	CV	Enables or disables coverage measurement
23	COVERAGE_DISPLAY	CVD	Displays coverage information
24	COVERAGE_LOAD	CVL	Loads coverage information
25	COVERAGE_RANGE	CVR	Sets a coverage range
26	COVERAGE_SAVE	CVS	Saves coverage information
27	DEFAULT_OBJECT_FO RMAT	DO	Sets the default object (program) format
28	DISASSEMBLE	DA	Disassembles memory contents
29	ERASE	ER	Clears the [Command Line] window
30	EVALUATE	EV	Evaluates an expression

Table 5.1 Simulator/Debugger Commands (cont)

No.	Command Name	Abbreviation	Function
31	EXEC_MODE	EM	Sets and displays execution mode.
32	FILE_LOAD	FL	Loads an object (program) file
33	FILE_SAVE	FS	Saves memory to a file
34	FILE_UNLOAD	FU	Unloads an object file from memory
35	FILE_VERIFY	FV	Verifies file contents against memory
36	GENERATE_MAKE_FILE	GM	Generates a build makefile for the current workspace
37	GO	GO	Executes user program
38	GO_RESET	GR	Executes user program from reset vector
39	GO_TILL	GT	Executes user program until temporary breakpoint
40	HALT	HA	Halts the user program
41	INITIALIZE	IN	Initializes the debugging platform
42	LOG	LO	Controls command output logging
43	MAP_DISPLAY	MA	Displays memory resource settings
44	MAP_SET	MS	Allocates a memory area
45	MEMORY_COMPARE	MC	Compares memory contents
46	MEMORY_DISPLAY	MD	Displays memory contents
47	MEMORY_EDIT	ME	Modifies memory contents
48	MEMORY_FILL	MF	Fills a memory area
49	MEMORY_FIND	MI	Finds a string in an area of memory
50	MEMORY_MOVE	MV	Moves a block of memory
51	MEMORY_TEST	MT	Tests a block of memory
52	OPEN_WORKSPACE	OW	Opens a workspace
53	PROFILE	PR	Enables or disables profile
54	PROFILE_DISPLAY	PD	Displays profile information
55	PROFILE_SAVE	PS	Saves the profile information to file
56	QUIT	QU	Exits HEW
57	RADIX	RA	Sets default input radix
58	REGISTER_DISPLAY	RD	Displays register values
59	REGISTER_SET	RS	Changes register contents
60	REMOVE_FILE	RF	Removes a file from the current project
61	RESET	RE	Resets CPU
62	RESPONSE	RS	Sets a window refresh area
63	SLEEP	-	Delays command execution
64	STATUS	STA	Displays the debugging platform status
65	STEP	ST	Steps program (by instructions or source lines)
66	STEP_MODE	SM	Sets the step mode
67	STEP_OUT	SP	Steps out of the current function

Table 5.1 Simulator/Debugger Commands (cont)

No.	Command Name	Abbreviation	Function
68	STEP_OVER	SO	Steps program, not stepping into functions
69	STEP_RATE	SR	Sets or displays rate of stepping
70	SUBMIT	SU	Executes a command file
71	SYMBOL_ADD	SA	Defines a symbol
72	SYMBOL_CLEAR	SC	Deletes a symbol
73	SYMBOL_LOAD	SL	Loads a symbol information file
74	SYMBOL_SAVE	SS	Saves a symbol information file
75	SYMBOL_VIEW	SV	Displays symbols
76	TCL	-	Enables or disables the TCL
77	TRACE	TR	Displays trace information
78	TRACE_ACQUISITION	TA	Enables or disables trace information acquisition
79	TRACE_SAVE	TV	Outputs trace information into a file
80	TRACE_STATISTIC	TST	Analyzes statistic information
81	TRAP_ADDRESS	TP	Sets a system call address
82	TRAP_ADDRESS_ DISPLAY	TD	Displays system call address settings
83	TRAP_ADDRESS_ ENABLE	TE	Enables or disables the system call
84	UPDATE_ALL_ DEPENDENCIES	UD	Updates the current projects build dependencies

The following describes the syntax of each command.

5.1 !(COMMENT)

Abbreviation: None

Description:

Allows a comment to be entered, useful for documenting log files.

Syntax:

! <text>

Parameter	Type	Description
<text>	Text	Output text

Example:

! Start of test routine	Outputs comment 'Start of test routine' into the [Command Line] window (and to the log file, if logging is active).

5.2 ADD_FILE

Abbreviation: AF

Description:

Adds a file to the current project.

Syntax:

af <filename>

Parameter	Type	Description
<filename>	String	File name

Example:

add_file $(PROJDIR) \\sbrk.c	Adds the file sbrk.c to the project from the project directory. Placeholders are valid syntax.

5.3 ANALYSIS

Abbreviation: AN

Description:

Enables/disables performance analysis. Counts are not automatically reset before running.

Syntax:

an [<state>]

Parameter	Type	Description
None		Displays the performance analysis state
<state>	Keyword	Enables or disables performance analysis
	Enable	Enables performance analysis
	Disable	Disables performance analysis
	Reset	Resets performance analysis counts

Examples:

ANALYSIS	Displays performance analysis state.
AN enable	Enables performance analysis.
AN disable	Disables performance analysis.
AN reset	Resets performance analysis counts.

5.4 ANALYSIS_RANGE

Abbreviation: AR

Description:

Sets a function for which the performance analysis is provided, or displays a function for which the performance analysis is provided without parameters.

Syntax:

ar [<function name>]

Parameter	Type	Description
None		Displays all functions for which the performance analysis is provided
<function name>	Character string	Name of function for which the performance analysis is provided

Examples:

ANALYSIS_RANGE sort	Provides the performance analysis for the function sort.
AR	Displays the function for which the performance analysis is provided.

5.5 ANALYSIS_RANGE_DELETE

Abbreviation: AD

Description:

Deletes the specified function, or all functions if no parameters are specified (it does **not** ask for confirmation).

Syntax:

ad [<index>]

Parameter	Type	Description
None		Deletes all functions
<index>	Numeric	Index number of function to delete

Examples:

ANALYSIS_RANGE_DELETE 6	Deletes the function with index number 6.
AD	Deletes all functions.

5.6 ASSEMBLE

Abbreviation: AS

Description:

Assembles mnemonics and writes them into memory. In assembly mode, '.' exits, '^' steps back 2-byte, the Enter key steps forward a byte.

Syntax:

as <address>

Parameter	Type	Description
<address>	Numeric	Address at which to start assembling

Example:

AS H’1000	Starts assembling from H’1000.

5.7 ASSERT

Abbreviation: None

Description:

Checks if an expression is true or false. It can be used to terminate the batch file when the expression is false. If the expression is false, an error is returned. This command can be used to write test harnesses for subroutines.

Syntax:

assert <expression>

Parameter	Type	Description
<expression>	Expression	Expression to be checked

Example:

ASSERT #R0 == 0x100	Returns an error if R0 does not contain 0x100.

5.8 BREAKPOINT

Abbreviation: BP

Description:

Specifies a breakpoint at the address where the instruction is written.

Syntax:

bp <address> [<count>] [<Action>]

Parameter	Type	Description
<address>	Numeric	The address of a breakpoint
<count>	Numeric	The number of times the instruction at the specified address is to be fetched (1 to 16383, default = 1).
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs (saves) data to a file
	Output (O)	Outputs (reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file.
		(optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt
		Interrupt vector number (0 to FF)
<priority>	Numeric	Interrupt priority (optional, default = 0)
		0 to 17

Examples:

BREAKPOINT 0 2	A break occurs when an attempt is made to execute the instruction at address H'0 for the second time.
BP C0 Input in.dat 100 2 8	Eight two-byte data fields are written from file "in.dat" to H'100 when an attempt is made to execute the instruction at address H'C0.

5.9 BREAK_ACCESS

Abbreviation: BA

Description:

Specifies a memory range as a break condition

Syntax:

ba <start address> [<end address>] [<mode>] [<Action>]

Parameter	Type	Description
<start address>	Numeric	The start address of a breakpoint
<end address>	Numeric	The end address of a breakpoint (optional, default = <start address>)
<mode>	Keyword	Access type (optional, default = RW).
	R	A break occurs when the specified range is read.
	W	A break occurs when the specified range is written to.
	RW	A break occurs when the specified range is read or written to.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs (saves) data to a file
	Output (O)	Outputs (reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt Interrupt vector number (0 to H'FF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Examples:

BREAK_ACCESS 0 1000 W	A break occurs when the specified range from address H'0 to address H'1000 is written to.
BA FFFF	A break occurs when address H'FFFF is accessed.

5.10 BREAK_CLEAR

Abbreviation: BC

Description:

Deletes breakpoints.

Syntax:

bc <index>

Parameter	Type	Description
<index>	Numeric	Index of the breakpoint to be canceled. If the index is omitted, all breakpoints are deleted.

Examples:

BREAK_CLEAR 0	The first breakpoint is deleted.
BC	All breakpoints are deleted.

5.11 BREAK_CYCLE

Abbreviation: BCY

Description:

Specifies the number of cycles as a break condition.

Syntax:

by <cycle> [<count>] [<Action>]

Parameter	Type	Description
<cycle>	Numeric	The condition matching the number of cycles <cycle>×n.
<count>	Keyword	The condition satisfying the number of times. (optional, default =ALL)
	All	Break condition is satisfied every time the condition is matched.
	Numeric	1 to H'FFFF Break condition is satisfied only when it matches the specified number of times.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs(saves) data to a file
	Output (O)	Outputs(reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt Interrupt vector number (0 to H'FF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Examples:

BREAK_CYCLE 1000 20	Specifies breaks to occur H'20 times in every H'1000 cycles.
BCY 5000	Specifies a break to occur in every H'5000 cycles.

5.12 BREAK_DATA

Abbreviation: BD

Description:

Specifies a memory data value as a break condition.

Syntax:

bd <address> <data> [<size>] [<option>] [<Action>]

Parameter	Type	Description
<address>	Numeric	The address where the break condition is checked.
<data>	Numeric	Access data
<size>	Keyword	Size (optional, default = B).
	Byte	Byte size
	Word	Word size
	Longword	Longword size
	Single	Single-precision floating-point size
	Double	Double-precision floating-point size
<option>	Keyword	Match or mismatch of data. The default is EQ.
	EQ	A break occurs when the data matches the specified value.
	NE	A break occurs when the data does not match the specified value.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs(saves) data to a file
	Output (O)	Outputs(reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file.
		(optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt
		Interrupt vector number (0 to FF)
<priority>	Numeric	Interrupt priority (optional, default = 0)
		0 to 17

Examples:

BREAK_DATA 0 100 L EQ	A break occurs when H'100 is written to memory address H'0 in longword.
BD C0 FF B NE	A break occurs when a value other than H'FF is written to memory address H'C0 in byte.
BD 4000 10	A break occurs when H'10 is written to memory address H'4000 in byte.

5.13 BREAK_DISPLAY

Abbreviation: BI

Description:

Displays a list of breakpoints.

Syntax:

bi

Parameter	Type	Description
None		Displays a list of breakpoints

Examples:

BREAK_DISPLAY	A list of breakpoints is displayed.
BI	A list of breakpoints is displayed.

5.14 BREAK_ENABLE

Abbreviation: BE

Description:

Enables or disables a breakpoint.

Syntax:

be <flag> [<index>]

Parameter	Type	Description
<flag>	Keyword	Enables or disables a breakpoint
	E	Enable
	D	Disable
<index>	Numeric	Index of the breakpoint to be canceled. If the index is omitted, all breakpoints are deleted.

Examples:

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

5.15 BREAK_REGISTER

Abbreviation: BR

Description:

Specifies a register data as a break condition

Syntax:

br <register name> [<data> <size>] [<option>] [<Action>]

Parameter	Type	Description
<register>	Character string	Register name.
<data>	Numeric	Access data.
<size>	Keyword	Access size. If no size is specified, the size of the specified register is assumed. Note that when data is specified, the size must not be omitted.
	byte	Byte size
	word	Word size
	longword	Longword size
	single	Single-precision floating-point size
	double	Double-precision floating-point size
<option>	Keyword	Match or mismatch of data. The default is EQ.
	EQ	A break occurs when the data matches the specified value.
	NE	A break occurs when the data does not match the specified value.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs(saves) data to a file
	Output (O)	Outputs(reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt Interrupt vector number (0 to H'FF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Examples:

BREAK_REGISTER R0 FFFF W EQ	A break occurs when the lower two bytes of the R0 register change to H'FFFF.
BR R10	A break occurs when the R10 register is written to.

5.16 BREAK_SEQUENCE

Abbreviation: BS

Description:

Sets sequential breakpoints

Syntax:

bs <address1> [<address2> [<address 3> [...]]] [<Action>]

Parameter	Type	Description
<address1> - <address8>	Numeric	Addresses of sequential breakpoints. Up to eight addresses can be specified.
<Action>	Keyword	Action taken when the conditions are satisfied (optional, default = Stop)
	Stop (P)	Halts the execution of the user program
	Input (I)	Inputs(saves) data to a file
	Output (O)	Outputs(reads) data from a file
	Interrupt (T)	Initiates a pseudo-interrupt

Format:

The method of defining each Action are as follows.

Stop

Input <filename> <addr> <size> <count>

Parameter	Type	Description
<filename>	Character string	The name of the file from which data is input
<addr>	Numeric	Address to which the data is read.
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)

Output <filename> <addr> <size> <count> [<option>]

Parameter	Type	Description
<filename>	Character string	The name of the file to which data is saved
<addr>	Numeric	Address from which data is output
<size>	Numeric	Size per data packet (1/2/4/8)
<count>	Numeric	Number of data packets (H'01 to H'FFFFFFFF)
<option>	Keyword	Specifies a new file or appends to an existing file. (optional, makes a new file when abbreviated.)
	A	Adds the data to the existing file.

Interrupt <interrupt type1> [<priority>]

Parameter	Type	Description
<interrupt type1>	Numeric	Type of interrupt Interrupt vector number (0 to H'FF)
<priority>	Numeric	Interrupt priority (optional, default = 0) 0 to 17

Examples:

BREAK_SEQUENCE 1000 2000	A break occurs when addresses H'1000 and H'2000 are passed in this order.
BS 1000	A break occurs when address H'1000 is executed.

5.17 BUILD

Abbreviation: BU

Description:

Starts a build operation on the current project.

Syntax:

bu

Parameter	Type	Description
None		Starts a build operation on the current project

Examples:

build	Starts a standard build process on the current project.

5.18 BUILD_ALL

Abbreviation: BL

Description:

Starts a build all operation on the current project.

Syntax:

bl

Parameter	Type	Description
None		Starts a build all operation on the current project

Examples:

Build_all	Starts a standard build all process on the current project.

5.19 CACHE

Abbreviation: None

Description:

Sets caching on or off, or displays the current caching state.

Syntax:

CACHE [<state>]

Parameter	Type	Description
None		Displays the current caching state
<state>	Keyword	Sets caching on or off
	On	Sets caching on
	Off	Sets caching off

Example:

CACHE	Displays the current caching state.
CACHE on	Turns on caching.

5.20 CHANGE_CONFIGURATION

Abbreviation: CC

Description:

Sets the current configuration.

Syntax:

cc <config name>

Parameter	Type	Description
<config name>	Character string	Configuration name

Example:

CC Debug	Sets the current configuration to Debug.

5.21 CHANGE_PROJECT

Abbreviation: CP

Description:

Sets the current project.

Syntax:

cp <project name>

Parameter	Type	Description
<project name>	Character string	Project name

Example:

CP PROJ2	Sets the current project to PROJ2.

5.22 COVERAGE

Abbreviation: CV

Description:

Enables or disables the coverage range measurement or resets coverage information.

Syntax:

cv [<state>]

Parameter	Type	Description
none		Displays coverage state.
<state>	enable	Enables coverage measurement.
	disable	Disables coverage measurement.
	reset	Resets result of coverage measurement.

Examples:

COVERAGE	Displays coverage state.
CV enable	Enables coverage measurement.
CV r	Resets result of coverage measurement.

5.23 COVERAGE_DISPLAY

Abbreviation: CVD

Description:

Displays coverage information.

Syntax:

cvd

Parameter	Type	Description
none		Displays coverage information.

Example:

COVERAGE_DISPLAY	Displays coverage information.

5.24 COVERAGE_LOAD

Abbreviation: CVL

Description:

Loads the coverage information from a .COV file.
If a wrong file format is specified or the specified file is not found, a warning message will be displayed.

Note: If the coverage range is specified by a source file, the saved .COV file cannot be loaded.

Syntax:

cvl <filename>

Parameter	Type	Description
filename	Character string	File name

Examples:

COVERAGE_LOAD TEST	Loads the coverage information from the TEST.COV file.
CLV COVERAGE.COV	Loads the coverage information from the COVERAGE.COV file.

5.25 COVERAGE_RANGE

Abbreviation: CVR

Description:

Sets the coverage range or displays the range of coverage measurement without parameters.

Syntax:

cvr [<start> <end>]

Parameter	Type	Description
none		Displays the coverage measurement range.
<start>	Numeric	Start address of the coverage measurement range.
<end>	Numeric	End address of the coverage measurement range.

Examples:

COVERAGE_RANGE H'1000 H'10FF	Measures the coverage of addresses between H'1000 and H'10FF.
CVR	Displays the range of coverage measurement.

5.26 COVERAGE_SAVE

Abbreviation: CVS

Description:

Saves the coverage information in a .COV or .TXT file.
If a wrong file extension is specified, an error message will be output.

Syntax:

cvs <filename>

Parameter	Type	Description
filename	Character string	File name

Examples:

COVERAGE_SAVE TEST	Saves the coverage information in the TEST.COV file.
CVS COVERAGE.COV	Saves the coverage information in the COVERAGE.COV file.

5.27 DEFAULT_OBJECT_FORMAT

Abbreviation: DO

Description:

Specifies the default format for loading object (program) files. The format specified with this command is only valid when the format specification is omitted from the FILE_LOAD command.

Syntax:

do [<format>]

Parameter	Type	Description
none		Displays the default format settings.
<format>	Keyword	Object format
	Binary	Binary type
	Elf/Dwarf2	Elf/Dwarf2 type
	IntelHex	Intel Hex type
	S-Record	S type

Example:

DEFAULT_OBJECT_FORMAT	Displays the default format settings.
DO binary	Specifies the binary format as the default format.

5.28 DISASSEMBLE

Abbreviation: DA

Description:

Disassembles memory contents to assembly-language code. The display of disassembled memory is fully symbolic.

Syntax:

da <address> [<length>]

Parameter	Type	Description
<address>	Numeric	Start address
<length>	Numeric	Number of instructions (optional, default = 16)

Examples:

DISASSEMBLE H'100 5	Disassembles 5 lines of code starting at H'100.
DA H'3E00 20	Disassembles 20 lines of code starting at H'3E00.

5.29 ERASE

Abbreviation: ER

Description:

Clears the [Command Line] window

Syntax:

er

Parameter	Type	Description
none		Clears the [Command Line] window

Example:

ER	Clears the [Command Line] window.

5.30 EVALUATE

Abbreviation: EV

Description:

Provides a calculator function, evaluating simple and complex expressions, with parentheses, mixed radices, and symbols. All operators have the same priority but parentheses may be used to change the order of evaluation. The operators have the same meaning as in C/C++. Expressions can also be used in any command where a number is required. Register names may be used, but must always be prefixed by the '#' character. The result is displayed in hexadecimal, decimal, octal, or binary.

Syntax:

ev <expression>

Parameter	Type	Description
<expression>	Expression	Expression to be evaluated

Valid operators:

&&	logical AND	||	logical OR	<<	left arithmetic shift	>>	right arithmetic shift
+	addition	-	subtraction	*	multiplication	/	division
%	modulo	|	bitwise OR	&	bitwise AND	~	bitwise NOT
^	bitwise exclusive OR	!	logical NOT	==	equal to	!=	unequal to
>	greater than	<	less than	>=	greater than or equal to	<=	less than or equal to

Examples:

EV H'123 + (D'73 | B'10)	Result: H'16E D'366 O'556 B'00000000000000000000000101101110
EV #R1 * #R2	Result: H'121 D'289 O'441 B'00000000000000000000000100100001

5.31 EXEC_MODE

Abbreviation: EM

Description:

Specifies the operation of the simulator/debugger when a simulation error occurs.

Syntax:

em [<mode>]

Parameter	Type	Description
None		Displays the current execution mode.
<mode>	Keyword	Execution mode (operation of the simulator/debugger when a simulation error occurs)
	S	Stops operation
	C	Continues operation

Examples:

EXEC_MODE	Displays the current execution mode.
EM c	Specifies to continue operation when a simulation error occurs.

5.32 FILE_LOAD

Abbreviation: FL

Description:

Loads an object code file to memory with the specified offset. Existing symbols are cleared, and the new ones are defined. If an offset is specified this will be added to the symbols. The file extension default is **.MOT**.

Syntax:

fl [<format>] <filename> [<offset>] [<state>]

Parameter	Type	Description
<format>	Keyword	Object format (optional, default = DEFAULT_OBJECT_FORMAT settings)
	Binary	Binary type
	Elf/Dwarf2	Elf/Dwarf2 type
	IntelHex	Intel-Hex type
	S-Record	S type
<filename>	Character string	File name
<offset>	Numeric	Offset to be added to load address (optional, default = 0)
<state>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

Examples:

FILE_LOAD A:\\BINARY\\TESTFILE.A22	Loads Motorola S-Record file "testfile.a22".
FL ANOTHER.MOT H'200	Loads Motorola S-Record file "another.mot" with an offset of H'200 bytes.

5.33 FILE_SAVE

Abbreviation: FS

Description:

Saves the specified memory area data to a file. The user is warned if about to overwrite an existing file. The file extension default is **.MOT**. Symbols are **not** automatically saved.

Syntax:

fs [<format>] <filename> <start> <end>

Parameter	Type	Description
<format>	Keyword	Object format (optional, default = DEFAULT_OBJECT_FORMAT settings)
	Binary	Binary type
	IntelHex	Intel-Hex type
	S-Record	S type
<filename>	Character string	File name
<start>	Numeric	Start address
<end>	Numeric	End address

Examples:

FILE_SAVE TESTFILE 0 H'2013	Saves address range 0-H'2013 as Motorola S-Record file "TESTFILE.MOT".
FS D:\\USER\\ANOTHER.A22 H'4000 H'4FFF	Saves address range H'4000-H'4FFF as Motorola S-Record format file "ANOTHER.A22".

5.34 FILE_UNLOAD

Abbreviation: FU

Description:

Unloads a file from memory at a specified offset. If no offset is supplied then memory will be scanned and the first occurrence of the file will be unloaded.

Syntax:

fu [<format>] <filename> [<offset>]

Parameter	Type	Description
<format>	Keyword	Object format (optional, default = DEFAULT_OBJECT_FORMAT settings)
	Binary	Binary type
	Elf/Dwarf2	Elf/Dwarf2 type
	IntelHex	Intel-Hex type
	S-Record	S type
<filename>	Character string	File name
<offset>	Numeric	Offset at which the file will be removed (optional)

Examples:

FILE_UNLOAD TEST.A22	Unloads the first occurrence of file 'test.a22' from memory.
FU ANOTHER 200	Unloads file 'another' from memory location H'200.

5.35 FILE_VERIFY

Abbreviation: FV

Description:

Verifies file contents against memory contents. The file data must be in a Motorola S-Record format. The file extension default is **.MOT**.

Syntax:

fv <filename> [<offset>]

Parameter	Type	Description
<filename>	Character string	File name
<offset>	Numeric	Offset to be added to file address (optional, default = 0)

Examples:

FILE_VERIFY A:\\BINARY\\TEST.A22	Verifies Motorola S-Record file "TEST.A22" against memory.
FV ANOTHER 200	Verifies Motorola S-Record file "ANOTHER.MOT" against memory with an offset of H'200 bytes.

5.36 GENERATE_MAKE_FILE

Abbreviation: GM

Description:

Generates a makefile for build usage outside of HEW.

Syntax:

gm [<mode>] [<scan>]

Parameter	Type	Description
<mode>	String	
	cccp	Generate make file from the current configuration in the current project (default).
	accp	Generate make file from all the configurations in the current project
	acap	Generate make file from all the configurations all projects
<scan>	String	
	Scan	Scan dependencies (default)
	noscan	noscan: Don't scan dependencies

Example:

GENERATE_MAKE_FILE cccp scan	Generates a makefile for the current configuration and current project. Before the makefile is generated the dependencies are scanned.

5.37 GO

Abbreviation: GO

Description:

Executes object code (the user program). While the user system is halted, a PC value is displayed.

Syntax:

go [<state>] [<address>]

Parameter	Type	Description
<state>	Keyword	Specifies whether or not to continue command processing during user program execution (optional, default = wait)
	wait	Causes command processing to wait until user program stops
	continue	Continues command processing during execution
<address>	Numeric	Start address for PC (optional, default = PC value)

Wait is the default and this causes command processing to wait until user program stops executing.

Continue allows you to continue to enter commands (but they may not work depending on the debugging platform).

Examples:

GO	Executes the user program from the current PC value. Command processing cannot be continued.
GO CONTINUE H'1000	Executes the user program from H'1000. Command processing can be continued.

5.38 GO_RESET

Abbreviation: GR

Description:

Executes the user program starting at the address specified in the reset vector.

While the user program is executing, the [Performance Analysis] window is updated.

Syntax:

gr [<state>]

Parameter	Type	Description
<state>	Keyword	Specifies whether or not to continue command processing during user program execution (optional, default = wait)
	wait	Causes command processing to wait until user program stops
	continue	Continues command processing during execution

Wait is the default and this causes command processing to wait until user program stops executing.

Continue allows you to continue to enter commands (but they may not work depending on the debugging platform)

Example:

GR	Executes the user program starting at the address specified in the reset vector (does not continue command processing).

5.39 GO_TILL

Abbreviation: GT

Description:

Executes the user program from the current PC with temporary breakpoints. This command takes multiple addresses as parameters, and these are used to set temporary PC breakpoints (these breakpoints only exist for the duration of the command).

Syntax:

gt [<state>] <address>...

Parameter	Type	Description
<state>	Keyword	Specifies whether or not to continue command processing during user program execution (optional, default = wait)
	wait	Causes command processing to wait until user program stops
	continue	Continues command processing during execution
<address>...	Numeric	Temporary breakpoint address (list)

Wait is the default and this causes command processing to wait until user program stops executing

Continue allows you to continue to enter commands (but they may not work depending on the debugging platform)

Example:

GO_TILL H'1000	Continues execution until the PC reaches address H'1000.

5.40 HALT

Abbreviation: HA

Description:

Halts the user program. This command can be used after the GO command if the GO command uses continue for option.

Syntax:

ha

Parameter	Type	Description
none		Halts the user program

Example:

HA	Halts the user program.

5.41 INITIALIZE

Abbreviation: IN

Description:

Initializes all breakpoints and memory mapping. It also initializes debugging platform, as if you had reselected the target DLL.

Syntax:

in

Parameter	Type	Description
none		Initializes debugging platform.

Example:

IN	Initializes debugging platform.

5.42 LOG

Abbreviation: LO

Description:

Controls logging of command output to file. If no parameters are specified, logging status is displayed. If an existing file is specified, you will be warned; if you answer 'No', data will be overwritten to the existing file, otherwise the file will be added. Logging is only supported for the command line interface.

Syntax:

lo [<state>|<filename>]

Parameter	Type	Description
none		Displays logging status
<state>	Keyword	Starts or suspends logging
	+	Starts logging
	-	Suspends logging
<filename>	Numeric	Specifies the logging output file

Examples:

LOG TEST	Stores the logging in file TEST.
LO -	Suspends logging.
LOG +	Resumes logging.
LOG	Displays logging status

5.43 MAP_DISPLAY

Abbreviation: MA

Description:

Displays memory resource settings.

Syntax:

ma

Parameter	Type	Description
none		Displays the current memory resource settings

Example:

MA	Displays the current memory resoruce settings.

5.44 MAP_SET

Abbreviation: MS

Description:

Allocates a memory area.

Syntax:

ms <start address> [<end address>] [<mode>]

Parameter	Type	Description
<start address>	Numeric	Specified start address
<end address>	Numeric	Specified end address (optional, default = start address)
<mode>	Keyword	Access type (optional, default = RW)
	R	Read only
	W	Write only
	RW	Displays the current memory mapping

Examples:

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

5.45 MEMORY_COMPARE

Abbreviation: MC

Description:

Compares the memory area (between the start address and the end address) with the memory starting at destination address.

Syntax:

mc <start> <end> <dest> [<access size>]

Parameter	Type	Description
<start>	Numeric	Start address
<end>	Numeric	End address
<dest>	Numeric	Destination address
<access size>	Keyword	Access size (optional, default = byte)
	byte	Displays in byte units
	word	Displays in word units (2 bytes)
	long	Displays in longword units (4 bytes)
	ascii	Displays in ASCII codes
	single	Displays in single-precision floating-point format
	double	Displays in double-precision floating-point format

Examples:

MEMORY_COMPARE H'1000 H'1FFF H'2000 LONG	Compares the memory between H'1000 and H'1FFF with the memory starting at H'2000 (using an access size of longword).

5.46 MEMORY_DISPLAY

Abbreviation: MD

Description:

Displays memory contents.

Syntax:

md <address> [<length>] [<mode>]

Parameter	Type	Description
<address>	Numeric	Start address
<length>	Numeric	Length (optional, default = H'100 bytes)
<mode>	Keyword	Display format (optional, default = byte)
	byte	Displays in byte units
	word	Displays in word units (2 bytes)
	long	Displays in longword units (4 bytes)
	ascii	Displays in ASCII codes
	single	Displays in single-precision floating-point format
	double	Displays in double-precision floating-point format

Examples:

MEMORY_DISPLAY H'C000 H'100 WORD	Displays H'100 bytes of memory starting at H'C000 in word units
MEMORY_DISPLAY H'1000 H'FF	Displays H'FF bytes of memory starting at H'1000 in byte units

5.47 MEMORY_EDIT

Abbreviation: ME

Description:

Allows memory contents to be modified. When editing memory the current location may be modified in a similar way to that described in the ASSEMBLE command description.

When editing, '.' exits edit mode, '^' goes back one data unit, and blank line goes forward without modification.

Syntax:

me <address> [<mode>] [<state>]

Parameter	Type	Description
<address>	Numeric	Address to edit
<mode>	Keyword	Format (optional, default = byte)
	byte	Edits in byte units
	word	Edits in word units
	long	Edits in longword units
	ascii	Edits in ASCII codes
	single	Edits in the single-precision floating-point format
	double	Edits in the double-precision floating-point format
<state>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

Example:

ME H'1000 WORD	Modifies memory contents in word units starting from H'1000 (with verification)

5.48 MEMORY_FILL

Abbreviation: MF

Description:

Modifies the contents in the specified memory area to the specified data value.

Syntax:

mf <start> <end> <data> [<mode>] [<state>]

Parameter	Type	Description
<start>	Numeric	Start address
<end>	Numeric	End address
<data>	Numeric	Data value
<mode>	Keyword	Data size (optional, default = byte)
	byte	Byte
	word	Word
	long	Longword
	single	Single-precision floating-point
	double	Double-precision floating-point
<state>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

Examples:

MEMORY_FILL H'C000 H'C0FF H'55AA WORD	Modifies memory contents in the range from H'C000 to H'C0FF to word data H'55AA.
MF H'5000 H'7FFF H'21	Modifies memory contents in the range from H'5000 to H'7FFF to data H'21.

5.49 MEMORY_FIND

Abbreviation: MI

Description:

Finds a string in a memory range.

Syntax:

mi <start> <end> <string>

Parameter	Type	Description
<start>	Numeric	Start address
<end>	Numeric	End address (including this address)
<string>	Numeric	Data to search for

Example:

MEMORY_FIND H'1000 H'2000 H'12	Finds the first occurrence of the data H'12 between addresses H'1000 and H'2000.

5.50 MEMORY_MOVE

Abbreviation: MV

Description:

Moves data in the specified memory area.

Syntax:

mv <start> <end> <dest> [<state>]

Parameter	Type	Description
<start>	Numeric	Start address
<end>	Numeric	End address (including this address)
<dest>	Numeric	Destination start address
<state>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

Examples:

MEMORY_MOVE H'1000 H'1FFF H'2000	Moves memory contents in the area from H'1000 to H'1FFF into H'2000.
MV H'FB80 H'FF7F H'3000	Moves memory contents in the area from H'FB80 to H'FF7F into H'3000.

5.51 MEMORY_TEST

Abbreviation: MT

Description:

Performs read, write, and verification testing in the specified address range. The original contents of memory have been replaced by the newly written data. The test will access the memory according to the map settings.

This simulator/debugger does not support the MEMORY_TEST command.

Syntax:

mt <start> <end>

Parameter	Type	Description
<start>	Numeric	Start address
<end>	Numeric	End address (including this address)

Examples:

MEMORY_TEST H'8000 H'BFFF	Tests from H'8000 to H'BFFF.
MT H'4000 H'5000	Tests from H'4000 to H'5000.

5.52 OPEN_WORKSPACE

Abbreviation: OW

Description:

Opens a workspace.

Syntax:

ow <filename>

Parameter	Type	Description
filename	Character string	Workspace file name

Example:

OW WKSP.HWS	Opens the WKSP.HWS file.

5.53 PROFILE

Abbreviation: PR

Description:

Enables, disables or sets the profiler display and resets the profiler information.

Syntax:

pr [<state>]

Parameter	Type	Description
None		Displays the profiler information.
<state>	Keyword	Enables, disables or sets the profiler display and resets the profiler information.
	enable	Enables the profiler.
	tree-off	Enables the profiler but does not trace function calls during profile information acquisition.
	disable	Disables the profiler.
	reset	Resets the profiler information.

Examples:

PROFILE ENABLE	Enables the profiler.
pr r	Resets the profiler information.

5.54 PROFILE_DISPLAY

Abbreviation: PD

Description:

Displays the profiler information.

Syntax:

pd [<mode>] [<state1>] [<state2>] [<count>]

Parameter	Type	Description
<mode>	Keyword	Specifies the method of displaying the profiler information. (optional, default=list)
	tree	Displays in tree format
	list	Displays in list format
<state1>	Keyword	Specifies whether or not to include child function information in the parent function cycle information. (optional, default=n)
	i	Specifies child function information to be included in the display.
	n	Specifies child function not to be included in the display.
<state2>	Keyword	Specifies whether or not to control displaying functions that are not executed (optional, default=a).
	e	Displays only executed functions.
	a	Displays all functions.
<count>	Numeric	Specifies the nesting level for calling functions to be displayed. This can be specified only when the <mode> parameter is 'tree' (optional, default = 16).

Examples:

PROFILE_DISPLAY TREE I	Specifies the profiler information to be displayed in tree format and to include child functions.
pd	Specifies the profiler information to be displayed in list format, without child function information.

5.55 PROFILE_SAVE

Abbreviation: PS

Description:

Saves the profiler information to a file. The default file extension is **.PRO**.

Syntax:

ps [<filename>]

Parameter	Type	Description
None		Saves the profiler information on all download modules to files.
<filename>	Character string	Specifies the name of the file to which profiler information is saved.

Example:

PROFILE_SAVE PR_INFO	Saves profiler information to a file named PR_INFO.PRO.

5.56 QUIT

Abbreviation: QU

Description:

Exits HEW. Closes a log file if it is open.

Syntax:

qu

Parameter	Type	Description
None		Exits HEW

Example:

QU	Exits HEW.

5.57 RADIX

Abbreviation: RA

Description:

Sets default input radix. If no parameters are specified, the current radix is displayed. Radix can be changed by using B', H', D', or O' before numeric data.

Syntax:

ra [<mode>]

Parameter	Type	Description
none		Displays current radix
<mode>	Keyword	Sets radix to specified type
	H	Sets radix to hexadecimal
	D	Sets radix to decimal
	O	Sets radix to octal
	B	Sets radix to binary

Examples:

RADIX	Displays the current radix.
RA H	Sets the radix to hexadecimal.

5.58 REGISTER_DISPLAY

Abbreviation: RD

Description:

Displays register contents.

Syntax:

rd [<register>]

Parameter	Type	Description
none		Displays all register contents
<register>	Keyword	Register name

Example:

RD	Displays all register contents
RD R0	Displays the contents of R0

5.59 REGISTER_SET

Abbreviation: RS

Description:

Changes the contents of a register.

Syntax:

rs <register> <value> [<mode>]

Parameter	Type	Description
<register>	Keyword	Register name
<value>	Numeric	Register value
<mode>	Keyword	Data size (optional, default = corresponding register size)
	byte	Byte
	word	Word
	long	Longword
	single	Single-precision floating-point
	double	Double-precision floating-point

Examples:

RS PC_StartUp	Sets the program counter to the address defined by the symbol _StartUp
RS R0 H'1234 WORD	Sets word data H'1234 to R0.

5.60 REMOVE_FILE

Abbreviation: RF

Description:

Removes a from the current project.

Syntax:

rf <filename>

Parameter	Type	Description
<filename>	String	File name

Example:

remove_file $(PROJDIR)\sbrk.c	Removes the file sbrk.c from the project. Placeholders are valid syntax.

5.61 RESET

Abbreviation: RE

Description:

Resets the microprocessor. All register values are set to the initial values of the device. Memory mapping and breakpoints are not initialized.

Syntax:

re

Parameter	Type	Description
none		Resets the microprocessor

Example:

RE	Resets the microprocessor.

5.62 RESPONSE

Abbreviation: RP

Description:

Specifies the frequency of the window update.
When a long refresh interval is specified, the simulation becomes faster but the response, such as for the break button, becomes slower. Set a frequency appropriate for the machine used.

Syntax:

rp [<instruction number>]

Parameter	Type	Description
<instruction number>	Numeric	Specifies the frequency, in number of instruction executions, the window is to update. 1 to 65535 (optional, default=40000)

Example:

RESPONSE 9	Sets the window to refresh every 9 information executions.

5.63 SLEEP

Abbreviation: None

Description:

Delays command execution for a specified period.

Syntax:

sleep <milliseconds>

Parameter	Type	Description
<milliseconds>	Numeric	Delayed time (ms)

The value must always be specified in decimal.

Example:

SLEEP D'9000	Delays 9 seconds.

5.64 STATUS

Abbreviation: STA

Description:

Displays the current status of the debugging platform. The contents of the [Platform] sheet in the [Status] window is displayed.

Syntax:

sta

Parameter	Type	Description
None		Displays the current status of the debugging platform.

Example:

STA	Displays the current status of the debugging platform.

5.65 STEP

Abbreviation: ST

Description:

Single step (in source line or instruction units) execution. Performs a specified number of instructions, from current PC. Default is stepping by lines if source debugging is available. Count default is 1.

Syntax:

st [<mode>] [<count>]

Parameter	Type	Description
<mode>	Keyword	Type of single step (optional)
	instruction	Steps by assembly instruction
	line	Steps by source code line
<count>	Numeric	Number of steps (optional, default = 1)

Example:

STEP 9	Steps code for 9 steps.

5.66 STEP_MODE

Abbreviation: SM

Description:

Selects the step mode.

Syntax:

sm <mode>

Parameter	Type	Description
<mode>	Keyword	Step mode
	Auto	Selects the mode automatically.
	Assembly	Steps by assembly-language instruction
	Source	Steps by source code line

Example:

STEP_MODE auto	Selects the step mode automatically. When the [Editor] window is active, execution steps by source code line. When the [Disassembly] window is active, execution steps by assembly-

language instruction.

5.67 STEP_OUT

Abbreviation: SP

Description:

Steps the program out of the current function. (i.e., a step up). This works for both assembly-language and source level debugging.

Syntax:

sp

Parameter	Type	Description
none		Steps the program out of the current function

Example:

SP	Steps the program out of the current function.

5.68 STEP_OVER

Abbreviation: SO

Description:

Performs a specified number of instructions from current PC.

This command differs from STEP in that it does not perform single step operation in subroutines or interrupt routines. These are executed at full speed.

Syntax:

so [<mode>] [<count>]

Parameter	Type	Description
<mode>	Keyword	Type of stepping (optional)
	instruction	Steps by assembly instruction
	line	Step by source code line
<count>	Numeric	Number of steps (optional, default = 1)

Example:

SO	Steps over 1-step code.

5.69 STEP_RATE

Abbreviation: SR

Description:

Controls the speed of stepping in the STEP and STEP_OVER commands. A rate of 6 causes the fastest stepping. A value of 0 is the slowest.

Syntax:

sr [<rate>]

Parameter	Type	Description
none		Displays the step rate
<rate>	Numeric	Step rate 0 to 6 (6 = fastest)

Examples:

SR	Displays the current step rate.
SR 6	Specifies the fastest step rate.

5.70 SUBMIT

Abbreviation: SU

Description:

Executes a file of emulator commands. This command can be used even in a command file to be processed. Any error aborts the file.

Syntax:

su <filename>

Parameter	Type	Description
<filename>	Character string	File name

Examples:

SUBMIT COMMAND.HDC	Processes the file COMMAND.HDC.
SU A:SETUP.TXT	Processes the file SETUP.TXT on drive A:.

5.71 SYMBOL_ADD

Abbreviation: SA

Description:

Adds a symbol, or changes an existing one.

Syntax:

sa <symbol> <value>

Parameter	Type	Description
<symbol>	Character string	Symbol name
<value>	Numeric	Value

Examples:

SYMBOL_ADD start H'1000	Defines the symbol start at H'1000.
SA END_OF_TABLE 1FFF	Uses current default radix and defines END_OF_TABLE at H'1FFF .

5.72 SYMBOL_CLEAR

Abbreviation: SC

Description:

Deletes a symbol. If no parameters are specified, deletes all symbols (after confirmation).

Syntax:

sc [<symbol>]

Parameter	Type	Description
none		Deletes all symbols
<symbol>	Character string	Symbol name

Examples:

SYMBOL_CLEAR	Deletes all symbols (after confirmation).
SC start	Deletes the symbol 'start'.

5.73 SYMBOL_LOAD

Abbreviation: SL

Description:

Loads symbols from file. File must be in XLINK Pentica-b format (i.e. 'XXXXH name'). The symbols are added to the existing symbol table.

Syntax:

sl <filename>

Parameter	Type	Description
<filename>	Character string	File name

Examples:

SYMBOL_LOAD TEST.SYM	Loads the file TEST.SYM.
SL MY_CODE.SYM	Loads the file MY_CODE.SYM.

5.74 SYMBOL_SAVE

Abbreviation: SS

Description:

Saves symbols to a file in XLINK Pentica-b format. The symbol file extension default is **.SYM**.

Syntax:

ss <filename>

Parameter	Type	Description
<filename>	Character string	File name

Examples:

SYMBOL_SAVE TEST	Saves symbol table to TEST.SYM.
SS MY_CODE.SYM	Saves the symbol table to MY_CODE.SYM.

5.75 SYMBOL_VIEW

Abbreviation: SV

Description:

Displays all defined symbols, or those containing the case sensitive string pattern.

Syntax:

sv [<pattern>]

Parameter	Type	Description
none		Displays all symbols
<pattern>	Character string	Character string that should be contained in the symbols to be displayed

Examples:

SYMBOL_VIEW BUFFER	Displays all symbols containing the word BUFFER.
SV	Displays all the symbols.

5.76 TCL

Abbreviation: None

Description:

Enables or disables the TCL.

Syntax:

tcl [<state>]

Parameter	Type	Description
None		Displays the TCL information.
<state>	Keyword	Enables or disables the TCL
	enable	Enables the TCL
	disable	Disables the TCL

Examples:

TCL	Displays the TCL information.
TCL enable	Enables the TCL.
TCL d	Disables the TCL.

5.77 TRACE

Abbreviation: TR

Description:

Displays the trace buffer contents. The record in the buffer that was executed first is 0; older records have positive offset values.

Syntax:

tr [[<start rec> [<count>]] | [<clear>]]

Parameter	Type	Description
<start rec>	Numeric	Record to start display (optional, default = most recent record - 9)
<count>	Numeric	Number of records to be displayed (optional, default = 10)
<clear>	Keyword	Clears all trace records (optional)
	clear	Clears all trace records

Note: When a negative value is specified for <start rec> (-0 cannot be specified), the value is recognized as a PTR value.

Examples:

TR 0 20	Displays twenty lines of trace buffer contents starting from the top of the buffer.
TR	Displays ten lines of trace buffer contents starting from the end of the buffer (the ten most recently executed lines).
TR –20 10	Displays trace buffer contents for which the PTR value is between –20 and -11.
TR C	Clears all trace records.

5.78 TRACE_ACQUISITION

Abbreviation: TA

Description:

Enables or disables trace information acquisition

Syntax:

ta <mode> [<trace full handling>] [<trace capacity>]

Parameter	Type	Description
<mode>	Keyword	Enables or disables trace information acquisition.
	E	Trace information acquisition is enabled.
	D	Trace information acquisition is disabled.
<trace full handling>	Keyword	Specifies the operation when the trace buffer becomes full.
	C	Continues operation. (default)
	B	Stops operation.
<trace capacity>	Keyword	Specifies the capacity of the trace buffer.
	BUF=1	D'1024 records (default)
	BUF=4	D'4096 records
	BUF=16	D'16384 records
	BUF=32	D'32768 records

Note: The capacity of the trace buffer can be modified only while the trace buffer is empty.

Examples:

TRACE_ACQUISITION E B BUF=32	Trace information acquisition is enabled. Operation stops when the trace buffer becomes full. The capacity of the trace buffer is set to 32768.
TA D	Trace information acquisition is disabled.

5.79 TRACE_SAVE

Abbreviation: TV

Description:

Saves the trace information in files. The files are in text format, so the default file extension is .TXT.

Syntax:

tv<filename> [<mode>]

Parameter	Type	Description
<filename>	Character string	File name
<mode>	Keyword	File writing mode
	A	Appends information to the file.
	O	Overwrites the file. (default)

Examples:

TRACE_SAVE TEST	Saves the trace information in TEST.TXT.
TV TRACE.TXT	Saves the trace information in TRACE.TXT

5.80 TRACE_STATISTIC

Abbreviation: TST

Description:

Analyzes the statistic information under the specified conditions.

Syntax:

tst <item> <string>

Parameter	Type	Description
<item>	Character string	Statistic information item to be analyzed
<string>	Character string	Character string that specifies conditions

Example:

TST CODE1 E630	Analyzes the statistic information under condition CODE1 = E630.

5.81 TRAP_ADDRESS

Abbreviation: TP

Description:

Specifies the system call address for standard I/O and file I/O functions from the user program. For details, refer to section 4.22, Standard I/O and File I/O Processing.

Syntax:

tp <address>

Parameter	Type	Description
<address>	Numeric	System call address

Example:

TP 1000	Sets the system call address to 1000.

5.82 TRAP_ADDRESS_DISPLAY

Abbreviation: TD

Description:

Displays the current system call address for standard I/O and file I/O functions from the user program. For details, refer to section 4.22, Standard I/O and File I/O Processing.

The system call address is displayed in the following format:

H'xxxxxxxx Y

H'xxxxxxx: System call address (8-digit hexadecimal value)
Y: E: System call enabled
D: System call disabled

Syntax:

td

Parameter	Type	Description
None		Displays the current system call address and whether the system call is enabled or disabled.

Example:

TD	Displays the system call address value and whether the system call is enabled or disabled.

5.83 TRAP_ADDRESS_ENABLE

Abbreviation: TE

Description:

Enables or disables the system call for standard I/O and file I/O functions from the user program. For details, refer to section 4.22, Standard I/O and File I/O Processing.

Syntax:

te <mode>

Parameter	Type	Description
<mode>	Keyword	Enables or disables the system call.
	E	Enables the system call.
	D	Disable the system call.

Example:

TE E Enables the system call.

5.84 UPDATE_ALL_DEPENDENCIES

Abbreviation: UD

Description:

Updates the dependencies for the current project.

Syntax:

ud

Parameter	Type	Description
None		Updates the dependencies for the current project

Example:

update_all_dependencies Starts an update all dependencies operation on the current project.

Section 6 Messages

6.1 Information Messages

The simulator/debugger outputs information messages as listed in table 6.1 to notify users of execution status.

Table 6.1 Information Messages

Message	Contents
Break Access	The break access condition was satisfied and execution has stopped.
Break Cycle	The break cycle condition was satisfied and execution has stopped
Break Data	The break data condition was satisfied and execution has stopped.
Break Register	The break register condition was satisfied and execution has stopped.
Break Sequence	The break sequence condition was satisfied and execution has stopped.
PC Breakpoint	The breakpoint condition was satisfied and execution has stopped.
Sleep	Execution has been stopped by the SLEEP instruction.
Step Normal End	The step execution succeeded.
Stop	Execution has been stopped by the [Stop] button.
Trace Buffer Full	Since the Break mode was selected by [Trace buffer full handling] in the [Trace Acquisition] dialog box and the trace buffer became full, execution was terminated.

6.2 Error Messages

The simulator/debugger outputs error messages to notify users of the errors of user programs or operation. Table 6.2 lists the error messages.

Table 6.2 Error Messages

Message	Contents
Address Error	One of the following states occurred: • A PC value was an odd number. • An instruction was read from the internal I/O area. • Word data was accessed to an address other than address 2n. • Longword data was accessed to an address other than address 2n. Correct the user program to prevent the error from occurring.
Exception Error	An error occurred during exception processing. Correct the user program to prevent the error from occurring.
File Open Error	An error occurred during opening a file with the break of file-input/output action. Correct the file setting.
File Input Error	An error occurred during reading a file with the break of file-input/output action. Correct the file setting.
File Output Error	An error occurred during writing to a file with the break of file-input/output action. Correct the file setting.
Illegal Instruction	Either of the following states occurred: • A code other than an instruction was executed. • MOV.B Rn,@-SP or MOV.B @SP+,Rn was executed. Correct the user program to prevent the error from occurring.
Illegal Operation	Either of the following states occurred: • The relationship between flags C and H of CCR and the value before compensation was illegal in the DAA or DAS instruction. • A division by zero or an overflow occurred during DIVXU or DIVXS instruction execution. Correct the user program to prevent the error from occurring.
Memory Access Error	One of the following states occurred: • A memory area that had not been allocated was accessed. • Data was written to a memory area having the write protect attribute. • Data was read from a memory area having the read disable attribute. • A memory area in which memory does not exist was accessed. Allocate memory, change the memory attribute, or correct the user program to prevent the memory from being accessed.
System Call Error	System call error occurred. Modify the incorrect contents of registers R0, R1, and parameter block.
I/O area not exist	The I/O area was deleted. Be sure to set the I/O area.
The memory resource has not been set up	The memory resource was set outside the range of memory mapping. Modify the memory resource settings so that no error will occur.

Appendix A: Trouble Shooting

? ***I have clicked the "Stop Build" button or selected [Build->Stop Build] to abort a build but the build process has not stopped.***

The HEW will stop the build after it has built the current file (or executed the current single phase). If the builder does not respond after a longer period of time you can select **[Build->Terminate Current Tool]** which will attempt to forcibly terminate the current process. Do NOT assume that any output from the tool you terminated is valid. It is recommended that you delete any output files produced and ensure that the phase is executed again.

? ***I have a text file in the editor but it does not show any syntax coloring.***

Ensure that you have named the file (i.e. saved it) and that the "Enable syntax coloring" check box is set on the "Editor" tab of the "Tools Options" dialog, which is launched via **[Tools->Options...]**. HEW checks a file group to which the file's extension belongs and decides whether the file is colored. To view the current defined extensions and their file groups, use the "File Extensions" dialog which is launched via **[Project->File Extensions...]**. To view coloring information, use the "Format" tab of the "Tools Options" dialog, which is launched via **[Tools->Options...]**. (See the "Syntax Coloring" section in chapter 4, "Using the Editor" for details.)

? ***I want to change the settings of a tool but the [Tools->Administration…] menu option cannot be selected.***

[Tools->Administration...] cannot be selected while a workspace is open. To enter the "Tool Administration" dialog first close the current workspace.

? ***I did not edit any of the project files. But some of files were built again when I selected [Build->Build].***

A file is built again if any of the following conditions holds:

1. The options for the current file have been modified since the last build.
2. If any of the output files are missing.
3. Any of the source files have a newer date than the phase's output files.
4. Any of the dependent files have a newer date than the phase's output files.
5. In a user-defined custom build phase, the "Don't check for input file(s) existence before executing" check box is checked. To see the status of this check box, select **[Options->Build Phases…]**, select the phase in the "Build Phase Order" list in the "Build Order" tab, and click the "Modify…" button. Then you will see the "Don't check for input file(s) existence before executing" check box on the "Command" tab.
6. A subcommand file is selected in Optimizing Linkage Editor.

? ***I want to exclude a file in a project from build temporarily.***

Push the right button of your mouse on the file in the "Projects" tab of the "Workspace" window, and select **[Exclude Build <file>]**, where <file> is the selected file. Then the file is excluded from build. To include the file in build again, push the right button of your mouse on the file, and select **[Include Build <file>]**.

? ***I opened a workspace from my PC, and one of my colleagues opened the same workspace simultaneously from another PC. I changed the settings of the workspace and saved it. He or she saved the workspace after me. I opened the workspace again and found that the settings of the workspace differed from what I had set before.***

The settings saved last are enforced. Once HEW opens a workspace, it is updated inside the memory. HEW does not save the settings into a file until a user saves the workspace intentionally.

? ***I want HEW to automatically specify default options to a file, which is newly added to a project.***

You can specify default options to a phase if the phase is a multiple build phase. Select a phase from the **[Options]** menu. If the phase is a multiple build phase, a file list is displayed on the left-hand side of the options dialog (figure A.1). In the file list, open a folder of a file group to which you want to specify your default options. You will find a "Default Options" icon in the folder. Select the icon, specify options on the right-hand side of the options dialog, and click "OK". The options specified here will be given to any new file of that file group when it is first added to the project.

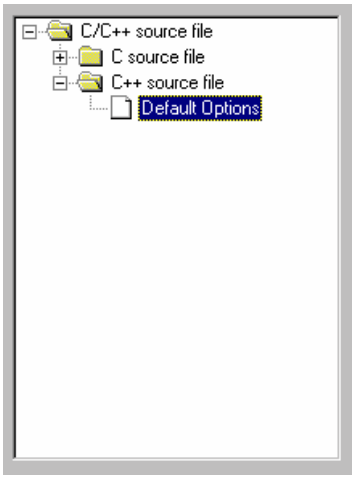


Figure A.1: Options Dialog File List

? ***I selected [Build->Build All], but the dependencies in the "Projects" tab of the "Workspace" window is not updated while selecting [Build->Build] updates the dependencies.***

[Build->Build All] does not update the dependencies in the "Projects" tab of the "Workspace" window. To update the dependencies, select **[Build->Update All Dependencies]**.

Appendix B: Regular Expressions

The High-performance Embedded Workshop editor allows you to include special characters in search strings when performing a find or replace operation. These characters are listed in table B.1 and are detailed in the following pages.

Table B.1: Regular Expression Characters

Character	Function
?	Matches any single character (except newline)
*	Matches any number of occurrences (0 or more) of any character except a newline
\n	Matches a new line character
\t	Matches a tab character
[ ]	Matches any one character or range listed within the brackets
\	Overrides any following regular expression character

- **Symbol: ?**
  Meaning: This character matches any single character, except the newline character.
  Example: t?p matches "top", "tip" but not "trap".

- **Symbol: ***
  Meaning: This character matches any number of occurrences (0 or more) of any character except a newline. Thus, this character will not match across new lines. The * character will match as few occurrences as are necessary to make the rest of the pattern match.
  Example 1: t*o matches the "to" of "too", the "tro" of "trowel" and the "ty o" of "sporty orange" but not "smart orange" because the * character does not match across a new line.

- **Symbol: \n**
  Meaning: This character matches the newline character.
  \n would be used to search for line endings or in patterns that cross line boundaries.
  Example 1: ;\n
  matches every occurrence of a newline following a semicolon
  Example 2: ;\nif
  searches for a semicolon, a new line and a line beginning with "if".

- **Symbol:** **\t**

  Meaning: This character matches the tab character.

  Example 1: \t8

  Finds every occurrence of a tab character followed by an 8.

  Example 2: init\t

  Finds every occurrence of a tab character following "init".

- **Symbol:** **[ ]**

  Meaning: This matches any one character or a range of single characters listed within the brackets. Brackets cannot be nested.

  [-] specifies a range of characters e.g. [a-z] or [0-9]. The beginning character in the range must have a lower ASCII value than the ending character of the range.

  [~] matches a single character if it is not any one of the characters between [~ and]. This pattern also matches newline characters, unless the newline character is included within the brackets.

  Example 1: [AEIOU]

  Finds every uppercase vowel.

  Example 2: [<>?]

  Finds a literal <, > or ?.

  Example 3: [A-Za-z0-9_]

  Matches an upper or lowercase letter, a digit or an underscore.

  Example 4: [~0-9]

  Matches any character except a digit.

  Example 5: [\t\n]

  Matches a space, a tab or newline.

  Example 6: [\]]

  Matches a literal] if] is placed after \.

- **Symbol:** \

  Meaning: This is the regular expression override character. If the character following the backslash is a regular expression character, it is treated as a normal character. The backslash is ignored if it is followed by a normal (non-regular expression) character.

  Example 1: *

  Searches for every occurrence of an asterisk.

  Example 2: \\

  Searches for every occurrence of a backslash.

Appendix C: Placeholders

This appendix describes how to use the placeholders, a feature provided by several of the High-performance Embedded Workshop components.

C.1 What is a Placeholder?

A placeholder is a special string, inserted into text, which is replaced at some subsequent time for the actual value. For example, one of the HEW placeholders is $(FULLFILE) which represents a file with a full path. Suppose that you have an editor in c:\myedit\myeditor.exe, which can take the file to edit as a parameter. When invoking the editor the following shortcut could be made, e.g.:

c:\myedit\myeditor.exe c:\files\file1.c

if you wanted to open FILE1.C from the directory c:\files. However, what happens if you want the HEW to open any file through this editor? The problem is that the command above is specific to "c:\files\file1.c". What we want to be able to do is to tell the HEW to use the editor specified but to open the file that I have chosen at that time. To do this, you can replace the specific name of the file for a general placeholder, i.e.:

c:\myedit\myeditor.exe **$(FULLFILE)**

Now whenever the HEW launches the editor with a file, it knows that it has to replace $(FULLFILE) with the file you have selected.

C.2 Inserting a Placeholder

Placeholders can only be entered into three specific edit fields within the HEW (figures C.1, C.2 and C.3). There are four ways a placeholder can be entered:

In the first example, place the insertion cursor at the point you would like to insert the placeholder and then select the required placeholder from the pop-up menu to the right of the edit field.

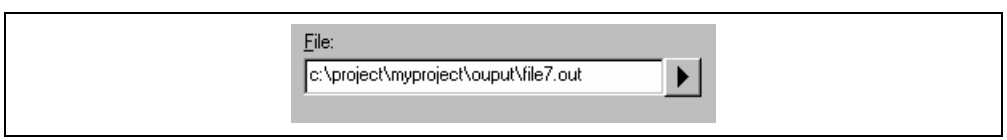


Figure C.1: Placeholder Pop-up Menu

In the second example, select the required placeholder other than "Custom directory" from the drop-down list box and specify a sub-directory relative to the directory shown by the placeholder. If you select "Custom directory", specify an absolute directory path in the "Sub-Directory" field.

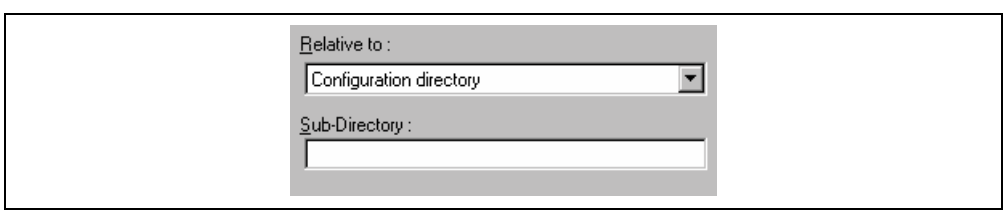


Figure C.2: Placeholder Drop-down List and Sub-Directory Field

In the third example, place the insertion cursor at the point you would like to insert the placeholder, select the required placeholder from the drop-down list box and then click the "Insert" button.

Figure C.3: Placeholder Drop-down List

In the fourth example, type the placeholder into the field directly. Ensure that you type the placeholder name in uppercase and that it is preceded by $(and followed by), i.e.

This is correct:

$(FILEDIR)

These are incorrect:

$(Filedir)
$(FILEDIR)
$FILEDIR

C.3 Available Placeholders

Table C.1 lists the placeholders and their meanings.

Table C.1: Placeholders

Placeholder	Meaning
$(FULLFILE)	Filename (including full path)
$(FILEDIR)	File directory
$(FILENAME)	Filename (excluding path including extension)
$(FILELEAF)	Filename (excluding path and extension)
$(EXTENSION)	File extension
$(WORKSPDIR)	Workspace directory
$(WORKSPNAME)	Workspace name
$(PROJDIR)	Project directory
$(PROJECTNAME)	Project name
$(CONFIGDIR)	Configuration directory
$(CONFIGNAME)	Configuration name
$(HEWDIR)	HEW installation directory
$(TCINSTALL)	Toolchain install directory (on option dialog)
$(TOOLDIR)	Tool installation directory (Tools Administration)
$(TEMPDIR)	Temp directory
$(WINDIR)	Windows® directory
$(WINSYSDIR)	Windows® system directory
$(EXEDIR)	Command directory
$(USERNAME)	User login (Version control)
$(PASSWORD)	User password (Version control)
$(VCDIR)	"Virtual" version control directory
$(COMMENT)	Comment (Version control)
$(LINE)	Line number of an error/warning

For example, the placeholders will be expanded as shown in table C.2.

Table C.2: Placeholder Expansions (Example)

Placeholder	Expanded placeholder (example)
$(FULLFILE)	c:\hew\workspace\project\file.src
$(FILEDIR)	c:\hew\workspace\project
$(FILENAME)	file.src
$(FILELEAF)	file
$(EXTENSION)	src
$(WORKSPDIR)	c:\hew\workspace
$(WORKSPNAME)	workspace
$(PROJDIR)	c:\hew\workspace\project
$(PROJECTNAME)	project
$(CONFIGDIR)	c:\hew\workspace\project\debug
$(CONFIGNAME)	debug
$(HEWDIR)	c:\hew
$(TCINSTALL)	c:\hew\toolchains\renesas\sh\511
$(TOOLDIR)	c:\hew\toolchains\renesas\sh\511
$(TEMPDIR)	c:\Temp
$(WINDIR)	c:\Windows
$(WINSYSDIR)	c:\Windows\System
$(EXEDIR)	v:\vc\win32
$(USERNAME)	JHARK
$(PASSWORD)	214436
$(VCDIR)	"c:\project" is mapped to "x:\vc\project"
$(COMMENT)	"Please Enter Comment" dialog is invoked
$(LINE)	12

In table C.2, we are assuming that

- a file path is "c:\hew\workspace\project\file.src"
- a workspace named "workspace" is located at "c:\hew\workspace"
- a project named "project" is located at "c:\hew\workspace\project".
- a configuration named "debug" has a configuration directory located at "c:\hew\workspace\project\debug".
- HEW.EXE is installed in "c:\hew".
- a *.HRF file of a toolchain (i.e. compiler, assembler, linker) is located in "c:\hew\toolchain\renesas\sh\511". This is referred to as $(TCINSTALL) on the option setting dialogs of the **[Options]** menu and as $(TOOLDIR) on the "Tools Administration" dialog.
- the Windows® operating system is installed in "c:\Windows" and the Windows® system directory is "c:\Windows\System".
- a version control executable path is "v:\vc\win32\ss.exe", a user name and its password to login the version control system are "JHARK" and "214436" respectively, $(COMMENT) is specified in a command line to the version control executable, and "c:\project" is mapped to "x:\vc\project" on the "Projects" tab of the "Version Control Setup" dialog, which is invoked via **[Tools->Version Control->Configure…]**.
- an error of compiler or assembler occurred at line 12.

Note: Not all of the placeholders are relevant in every field. For example, the $(LINE) placeholder has no meaning when specifying a dependent files location. $(USERNAME), $(PASSWORD), $(VCDIR), and $(COMMENT) placeholders are acceptable only in version control. If you enter a placeholder into an edit field where it is not acceptable you might be informed.

C.4 Placeholder Tips

Placeholders are there to allow you to create flexible paths to the various files used by the system.

- If there is a placeholder pop-up menu () next to an edit field into which you are about to enter a path or file, you should consider how you can use a placeholder to make that path or file definition flexible.
- If you use several configurations, then the $(CONFIGDIR) placeholder is very useful to ensure that files can be written to and from the current configuration's directory.
- Wherever possible, use a placeholder. They can always be removed or added later so don't be afraid to experiment.

Appendix D: I/O File Format

HEW formats the [IO] window based on information it finds in an I/O Register definition file. When you select a debugging platform, HEW will look for a "<*device*>.IO" file corresponding to the selected device and load it if it exists. This file is a formatted text file that describes the I/O modules and the address and size of their registers. You can edit this file, with a text editor, to add support for memory mapped registers or peripherals you may have specific to your application e.g. registers in an ASIC device mapped into the microcomputer's address space.

D.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.

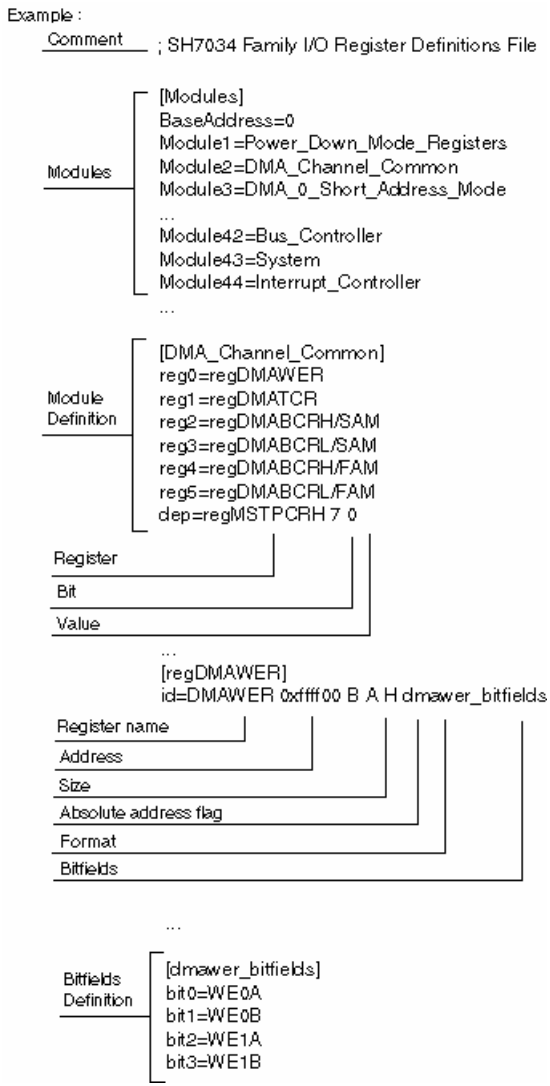
The [Register] definition entry is entered in the format id=<name> <address> [<size> [<absolute>[<format>[<bitfields>]]]].

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.



Appendix E: 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
```

H8S, H8/300 Series
High-performance Embedded Workshop 3 User's Manual

Publication Date: Rev.1.00, July 3, 2003
Published by: Sales Strategic Planning Div.
Renesas Technology Corp.
Edited by: Technical Documentation & Information Department
Renesas Kodaira Semiconductor Co., Ltd.

H8S, H8/300 Series
High-performance Embedded Workshop 3
User's Manual

Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan

REJ10B0026-0100H