

To all our customers

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

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.)

Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

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

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

Cautions

Keep safety first in your circuit designs!

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

Notes regarding these materials

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

Hitachi Microcomputer Development Environment System

High-performance Embedded Workshop

(for Windows[®] 95/98 and Windows NT[®] 4.0)

User's Manual



ADE-702-201A

Rev. 2.0

02/18/00

Hitachi, Ltd.

HS6400EWIW1S

Cautions

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

Trademarks

Microsoft, MS-DOS, Windows, Windows NT are registered trademarks of Microsoft Corporation.
Visual SourceSafe is a trademark of Microsoft Corporation.

IBM is a registered trademark of International Business Machines Corporation.

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

Document Information

Product Code: S32HEWM

Version: 1.1

Copyright © Hitachi Micro Systems Europe Ltd. 1999. All rights reserved.

Copyright © Hitachi, Ltd. 1999. All rights reserved.

Introduction

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

- A configurable build engine which 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 configure the environment to run your own tools.
- Version control support.
- Attach other specially designed Hitachi Embedded Workshop extensions such as Hitachi Debugging Interface (HDI).

The Hitachi 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. It is intended to be used in tandem with the Hitachi Debugging Interface (HDI) to provide a complete development tool suite.

About This Manual


This manual describes how to use the HEW system. The first chapter, “*Overview*” presents the main concepts of the HEW system. Chapter two, “*Build Basics*” explains how the build system works, how it manipulates files and how options are specified. “*Using the Editor*”, chapter three, describes in detail the features of the editor whilst chapter four, “*Advanced Build Features*” returns to the topic of the build system for a much more in depth explanation of its features and customizability. Chapter five, “*Tool Administration*” details the tool management philosophy of the HEW and then chapter six, “*Customizing the Environment*”, shows which environmental features can be configured. Finally, chapter seven, “*Version Control*” and chapter eight, “*Using the Custom Version Control System*”, explain how to set-up and configure a version control tool for use with the HEW.

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. |
| <u>"enter this string"</u> | 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

| | | |
|-----------|--|----|
| Section 1 | Overview | 1 |
| 1.1 | Workspaces, Projects and Files | 1 |
| 1.2 | The Main Window | 2 |
| 1.2.1 | The Title Bar | 2 |
| 1.2.2 | The Menu Bar | 3 |
| 1.2.3 | The Toolbars | 3 |
| 1.2.4 | The Workspace Window | 5 |
| 1.2.5 | The Editor Window | 8 |
| 1.2.6 | The Output Window | 9 |
| 1.2.7 | The Status Bar | 10 |
| 1.3 | The Help System | 11 |
| 1.4 | Launching the HEW | 12 |
| 1.5 | Creating a New Workspace | 13 |
| 1.6 | Opening a Workspace | 14 |
| 1.7 | Saving a Workspace | 16 |
| 1.8 | Closing a Workspace | 16 |
| 1.9 | Exiting the HEW | 16 |
| Section 2 | Build Basics | 17 |
| 2.1 | The Build Process | 17 |
| 2.2 | Project Files | 18 |
| 2.2.1 | Adding Files to a Project | 19 |
| 2.2.2 | Removing Files from a Project | 20 |
| 2.2.3 | Excluding a Project File from Build | 22 |
| 2.2.4 | Including a Project File in Build | 22 |
| 2.3 | File Extensions and File Groups | 22 |
| 2.4 | Specifying How to Build a File | 28 |
| 2.5 | Build Configurations | 29 |
| 2.5.1 | Selecting a Configuration | 30 |
| 2.5.2 | Adding and Deleting Configurations | 31 |
| 2.6 | Building a Project | 32 |
| 2.6.1 | Building a Project | 32 |
| 2.6.2 | Building Individual Files | 32 |
| 2.6.3 | Stopping a Build | 33 |
| 2.6.4 | The Output Window | 33 |
| 2.6.5 | Controlling the Content of the Output Window | 33 |
| 2.7 | File Dependencies | 34 |
| 2.8 | Configuring the Workspace Window | 35 |
| 2.8.1 | Show dependencies under each file | 35 |

| | | |
|---------------------------------|---|----|
| 2.8.2 | Show standard library includes..... | 36 |
| 2.8.3 | Show file paths | 37 |
| 2.8.4 | Show file groups in separate folders..... | 37 |
| 2.9 | Setting the Current Project..... | 38 |
| 2.10 | Inserting a project into a workspace | 38 |
| 2.11 | Specifying dependencies between projects | 40 |
| 2.12 | Removing a project from a workspace | 42 |
| 2.13 | Loading/unloading a project into/from a workspace | 42 |
| 2.14 | Renaming a workspace or a project..... | 43 |
| Section 3 Using the Editor..... | | 45 |
| 3.1 | The Editor Window | 45 |
| 3.2 | Working with Multiple Files..... | 46 |
| 3.3 | The Editor Toolbars | 47 |
| 3.3.1 | Editor Toolbar Buttons | 47 |
| 3.3.2 | Search Toolbar Buttons | 48 |
| 3.3.3 | Bookmarks Toolbar Buttons..... | 49 |
| 3.3.4 | Templates Toolbar Buttons..... | 49 |
| 3.4 | Standard File Operations | 49 |
| 3.4.1 | Creating a New File..... | 49 |
| 3.4.2 | Saving a File | 50 |
| 3.4.3 | Saving all Files | 50 |
| 3.4.4 | Opening a File | 50 |
| 3.4.5 | Closing Files | 51 |
| 3.5 | Editing a File..... | 52 |
| 3.6 | Searching and Navigating Through Files..... | 53 |
| 3.6.1 | Finding Text | 53 |
| 3.6.2 | Finding Text in Multiple Files | 54 |
| 3.6.3 | Replacing Text..... | 55 |
| 3.6.4 | Jumping to a Specified Line | 57 |
| 3.7 | Bookmarks..... | 57 |
| 3.8 | Printing a File | 58 |
| 3.9 | Configuring Text Layout | 59 |
| 3.9.1 | Page Set-up..... | 59 |
| 3.9.2 | Changing Tabs | 60 |
| 3.10 | Auto Indentation | 61 |
| 3.11 | Splitting a Window | 62 |
| 3.12 | Configuring Text | 63 |
| 3.12.1 | Changing the Editor Font..... | 63 |
| 3.13 | Syntax Coloring | 64 |
| 3.14 | Templates..... | 67 |
| 3.14.1 | Defining a Template | 67 |
| 3.14.2 | Inserting a Template | 68 |

| | | |
|--|---|------------|
| 3.15 | Brace Matching..... | 69 |
| Section 4 Advanced Build Features | | 71 |
| 4.1 | The Build Process Revisited | 71 |
| 4.1.1 | What is a build? | 71 |
| 4.2 | Creating a Custom Build Phase | 74 |
| 4.3 | Ordering Build Phases | 78 |
| 4.3.1 | Build Phase Order..... | 79 |
| 4.3.2 | Build File Phase Order..... | 82 |
| 4.4 | Setting Custom Build Phase Options | 83 |
| 4.4.1 | Options Tab | 84 |
| 4.4.2 | Output Files Tab | 85 |
| 4.4.3 | Dependent Files Tab | 87 |
| 4.5 | File Mappings | 89 |
| 4.6 | Controlling the Build | 91 |
| 4.7 | Logging Build Output | 92 |
| 4.8 | Changing Toolchain Version | 93 |
| 4.9 | Using the Hitachi Debugging Interface (HDI) | 94 |
| 4.10 | Generating a Makefile | 95 |
| Section 5 Tool Administration..... | | 97 |
| 5.1 | Tool Locations | 98 |
| 5.2 | HEW Registration Files (*.HRF)..... | 98 |
| 5.3 | Registering Components | 100 |
| 5.3.1 | Searching Drives for Components | 100 |
| 5.3.2 | Registering a Single Component | 101 |
| 5.4 | Unregistering Components | 101 |
| 5.5 | Viewing and Editing Component Properties | 102 |
| 5.6 | Uninstalling Components | 104 |
| 5.7 | Technical Support Issues | 106 |
| Section 6 Customizing the Environment | | 107 |
| 6.1 | Customizing the Toolbar | 107 |
| 6.2 | Customizing the Tools Menu | 109 |
| 6.3 | Configuring the Help System | 112 |
| 6.4 | Specifying Workspace Options..... | 114 |
| 6.4.1 | Open last workspace at start-up | 114 |
| 6.4.2 | Restore the files on opening workspace..... | 115 |
| 6.4.3 | Display workspace information dialog on opening workspace | 115 |
| 6.4.4 | Save workspace before executing any phases..... | 116 |
| 6.4.5 | Prompt before saving workspace..... | 116 |
| 6.4.6 | Default directory for new workspaces | 116 |
| 6.5 | Using an External Editor..... | 118 |

| | | |
|--|--|-----|
| 6.6 | Customizing File Save | 119 |
| 6.6.1 | Save files before executing any tools..... | 120 |
| 6.6.2 | Prompt before saving files | 120 |
| Section 7 Version Control | | 122 |
| 7.1 | Selecting a Version Control System | 123 |
| Section 8 Using the Custom Version Control System..... | | 126 |
| 8.1 | Defining Version Control Menu Options | 126 |
| 8.1.1 | System menu options and toolbar buttons | 128 |
| 8.1.2 | User menu options | 130 |
| 8.2 | Defining Version Control Commands | 132 |
| 8.2.1 | Executable return code | 132 |
| 8.3 | Specifying Arguments | 133 |
| 8.3.1 | Specifying File Locations | 134 |
| 8.3.2 | Specifying Environment | 138 |
| 8.3.3 | Specifying Comments..... | 139 |
| 8.3.4 | Specifying a User Name and Password | 140 |
| 8.4 | Controlling Execution..... | 142 |
| 8.4.1 | Prompt before executing command | 142 |
| 8.4.2 | Run in DOS Window | 142 |
| 8.4.3 | Use forward slash '/' as version control directory delimiter | 142 |
| 8.5 | Importing and Exporting a Set-up | 143 |
| Appendix A Regular Expressions..... | | 144 |
| Appendix B Placeholders | | 146 |
| B.1 | What is a Placeholder?..... | 146 |
| B.2 | Inserting a Placeholder..... | 146 |
| B.3 | Available Placeholders | 148 |
| B.4 | Placeholder Tips | 150 |
| Appendix C Using Visual SourceSafe..... | | 151 |
| C.1 | Overview | 151 |
| C.2 | Setting up the Visual SourceSafe Project | 152 |
| C.3 | Identifying Commands to be Used | 152 |
| C.4 | Defining Commands | 153 |
| C.4.1 | Defining the "History" command | 153 |
| C.4.2 | Command | 155 |
| C.4.3 | File..... | 155 |
| C.4.4 | User..... | 155 |
| C.4.5 | Setting up directory mappings | 156 |
| C.4.6 | Setting environment variables | 159 |

| | | |
|-----------------------------------|-----------------------------|-----|
| C.4.7 | And finally..... | 161 |
| C.5 | Executing Commands | 162 |
| Appendix D HMAKE User Guide | | 163 |
| D.1 | Command line..... | 163 |
| D.1.1 | Basic structure | 163 |
| D.1.2 | Exit codes | 163 |
| D.1.3 | Parameters | 163 |
| D.2 | File syntax..... | 164 |
| D.2.1 | Variable declarations | 164 |
| D.3 | Description blocks | 165 |
| D.3.1 | Basic outline | 165 |
| D.3.2 | Sub command files | 166 |
| D.4 | Comments..... | 167 |
| D.5 | Message commands | 167 |
| Index | | 168 |

Section 1 Overview

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

1.1 Workspaces, Projects and Files

Just as a wordprocessor allows you to create and modify documents, the Hitachi Embedded Workshop allows you to create and modify workspaces. A workspace can be thought of as a container of projects and, similarly, a project can be thought of as a container of project files. Thus, each workspace contains one or more projects and each project contains one or more files. Figure 1.1 illustrates this graphically.

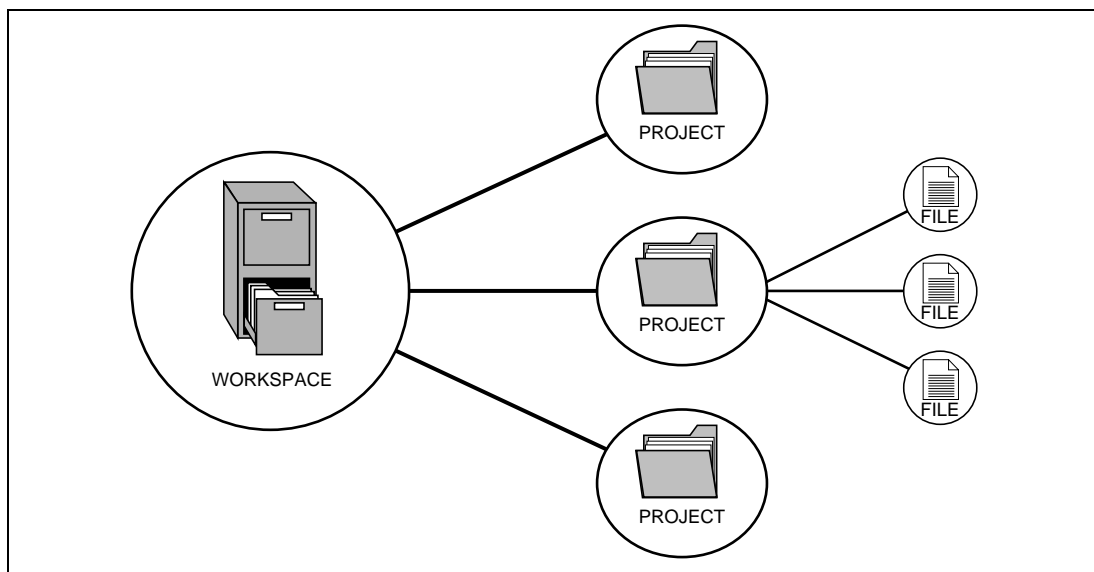


Figure 1.1 Workspaces, Projects and Files

Workspaces allow you to group related projects together. For example, you may have an application which 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.

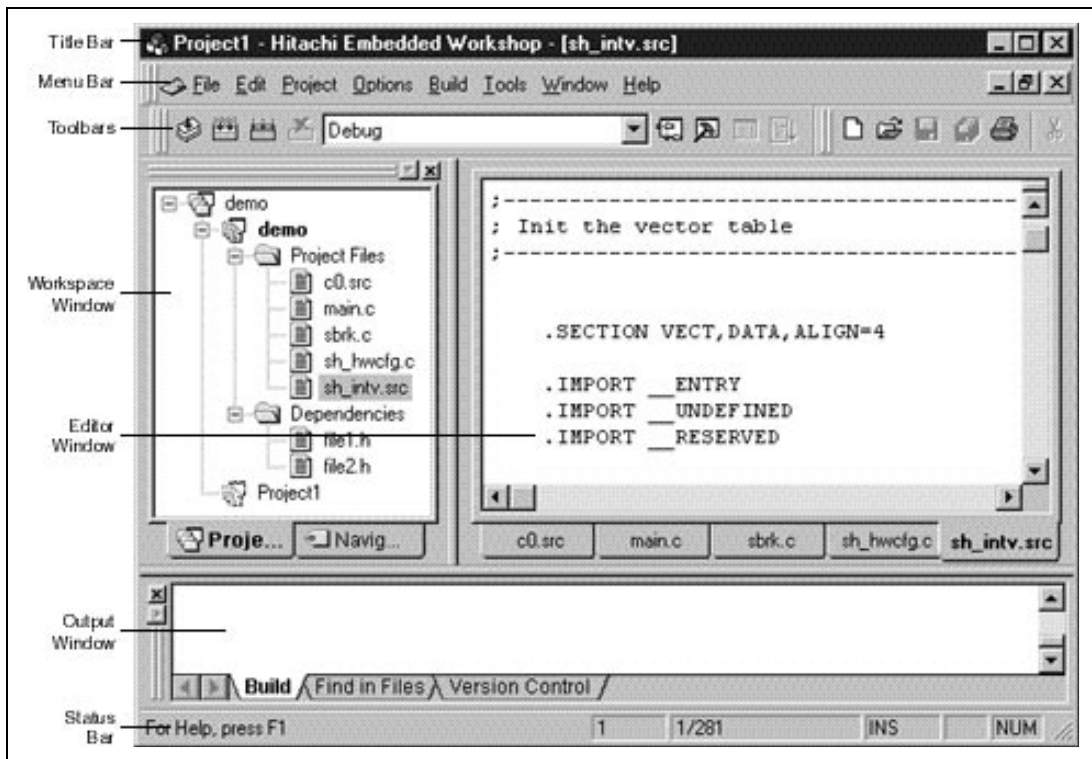


Figure 1.2 HEW Main Window

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 iconify the HEW. 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 8 menus: File, Edit, Project, Options, Build, Tools, Window and Help. All of the menu options are grouped logically under these headings. For instance, if you want to open a file then the file menu is where you will find the right menu option, if you want to set-up a tool then the tools menu is the correct selection. The function of the various menu options will be covered by the following sections 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 six default toolbars: Editor, Standard, Search, Bookmarks, Templates and Version Control (as shown in figures 1.3-1.8). Toolbars can be created, modified and removed via the **[Tools->Customize...]** menu option (see chapter 6, “*Customizing the Environment*”, for further information).

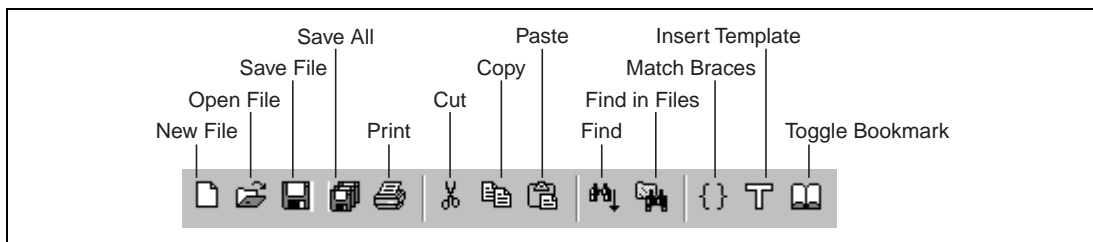


Figure 1.3 Editor Toolbar

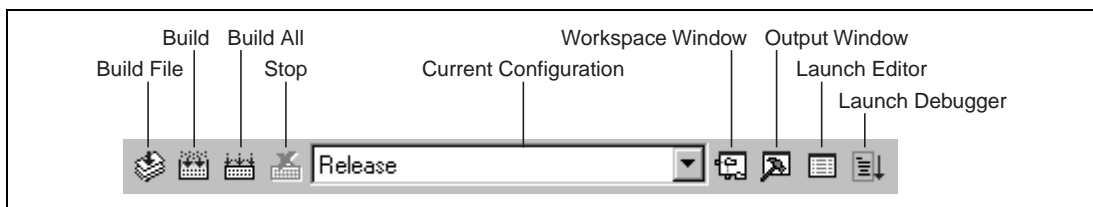


Figure 1.4 Standard Toolbar

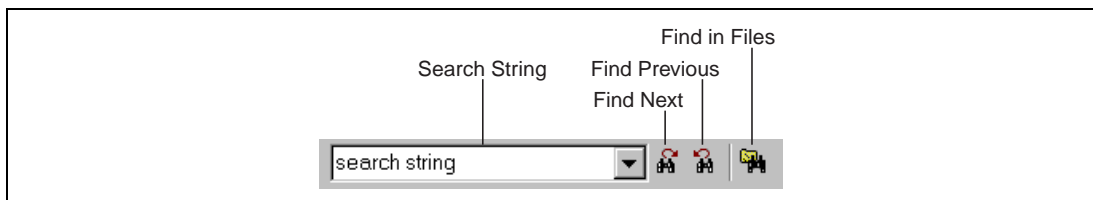


Figure 1.5 Search Toolbar

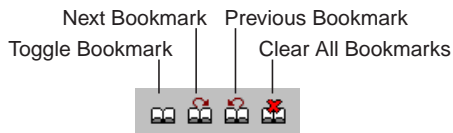


Figure 1.6 Bookmarks Toolbar

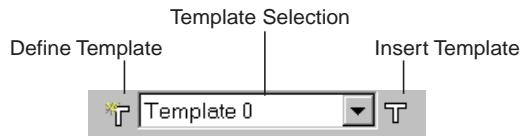


Figure 1.7 Templates Toolbar

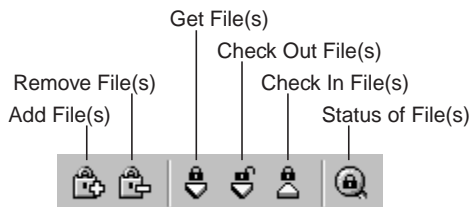


Figure 1.8 Version Control Toolbar

When a menu bar or a toolbar is docked, it has a control bar as shown in figure 1.9.i. If you want to move a docked menu bar or toolbar, click and drag its control bar to the new location. Figure 1.9.i shows the menu bar when it is docked and figure 1.9.ii shows the menu bar when it is floating.

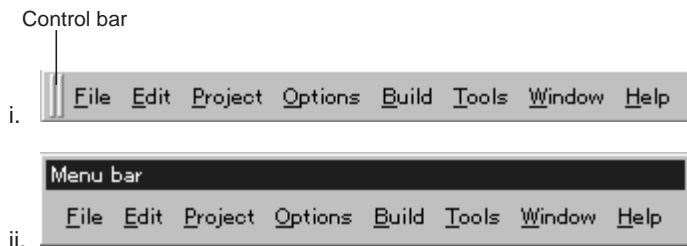


Figure 1.9 Menu bar, Docked and Floating

Figure 1.10.i shows standard toolbar when it is docked and figure 1.10.ii shows the standard toolbar when floating.

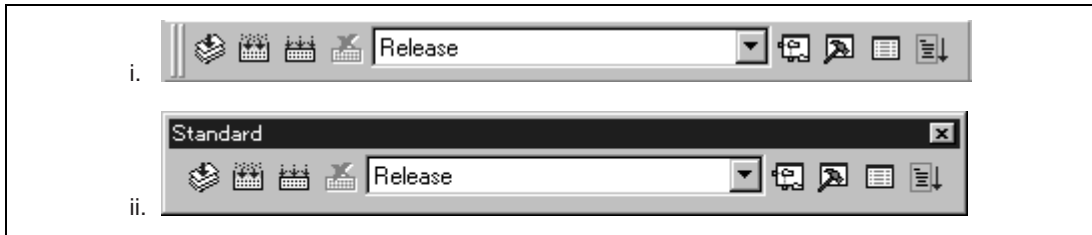


Figure 1.10 Standard Toolbar, Docked and Floating

- 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 two tabs. The “Projects” tab shows the current workspace, projects and files (figure 1.11). You can quickly open any project file or dependent file by double clicking on its corresponding icon.

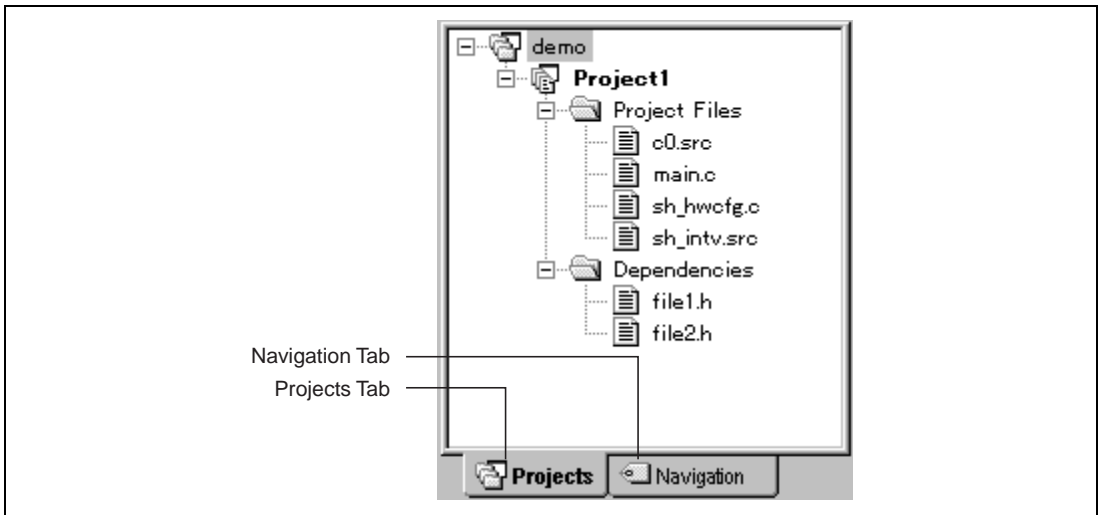


Figure 1.11 Workspace Window Projects Tab

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 1.12 shows ANSI C functions. See chapter 2, “*Build Basics*”, for more information on the “Workspace” window.

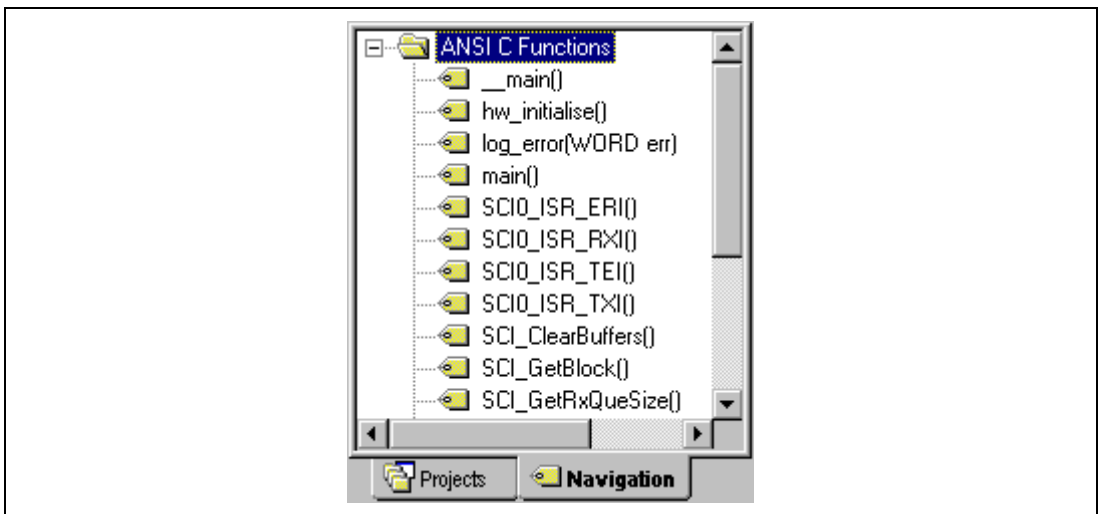


Figure 1.12 Workspace Window Navigation Tab

If docking is allowed on the “Workspace” window or the Output” window, the window can be docked or can be floating. Otherwise, the window is a normal window inside the main window.

- ☞ 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 1.13.i shows a docked “Workspace” window, and figure 1.13.ii shows a floating “Workspace” window.

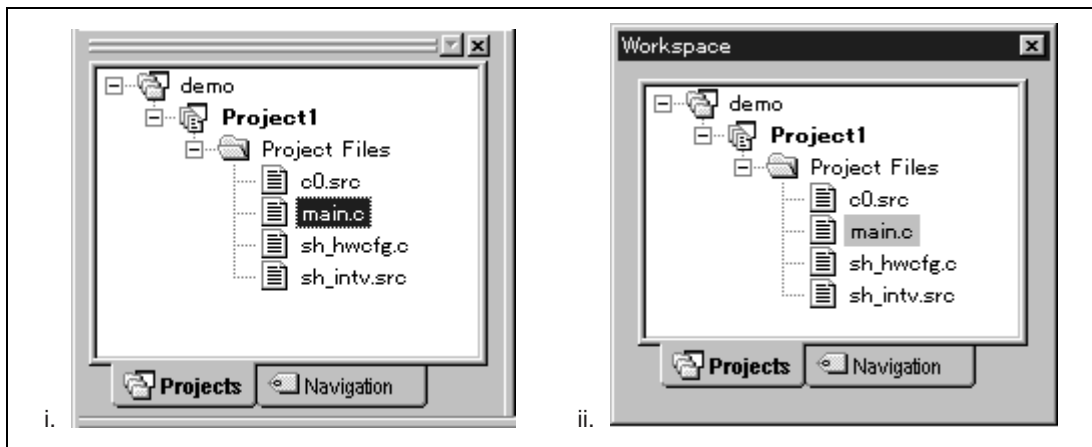


Figure 1.13 Workspace Window, Docked and Floating

When the “Workspace” window or the “Output” window is docked, it has a control bar as shown in figure 1.14. If you want to move a docked window, click and drag its control bar to the new location.



Figure 1.14 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 **[Window->Workspace]** or **[Window->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 1.15).

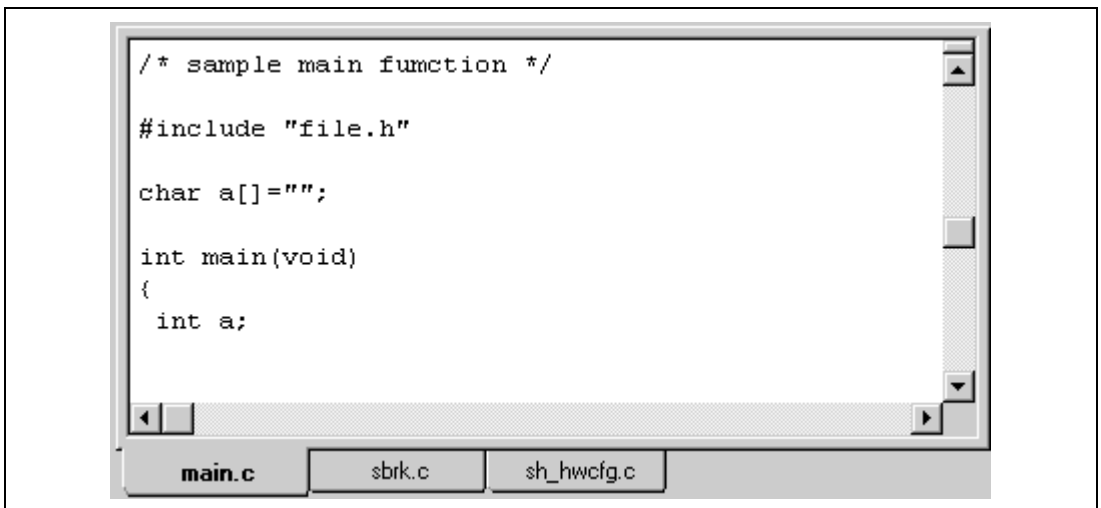


Figure 1.15 Editor Window

The editor window can be customized via the “Tools Options” dialog which can be invoked via the [**Tools->Options...**] menu option. This dialog allows you to configure fonts, colors, tabs and so on. If you would prefer to use your favourite editor rather than the HEW internal editor then specify your alternative in the “Tools Options” dialog. For further details on how to use and configure the editor, refer to chapter 3, “*Using the Editor*”.

1.2.6 The Output Window

The “Output” window (figure 1.16) has, at most, three 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.

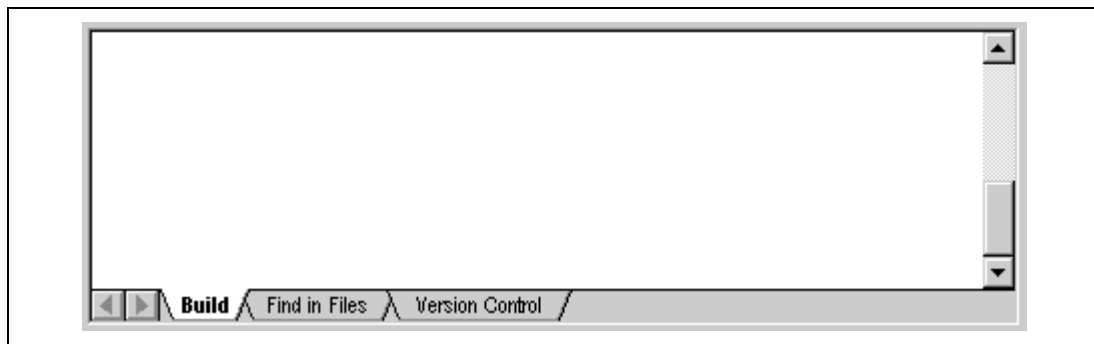


Figure 1.16 Output Window

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 3, “*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*”.

You can clear the contents displayed on the current tab, copy a text string on the current tab to the Windows® clipboard, and set a font used on the “Output” window.

- To clear the contents of the current tab of the “Output” window:
Push the right mouse button anywhere inside the current tab of the “Output” window and select [**Clear**] on the pop-up menu (figure 1.17).
- To copy a text string on the current tab of the “Output” window to the Windows® clipboard:
Select a text string on the current tab. Push **CTRL+C**. Or, push the right mouse button anywhere on the current tab and select [**Copy**] on the pop-up menu (figure 1.17).



Figure 1.17 Output Window Pop-up Menu, Version Control Tab

- ➡ To set a font displayed the “Output” window:

Push the right mouse button anywhere on the current tab and select **[Set Font...]** on the pop-up menu (figure 1.17). Then the “Set Font” dialog (figure 1.18) will be displayed. Select a font from the “Font” drop-down list and select the size of the font from the “Size” drop-down list. Click the “OK” button to make the settings effective.

Note: On Japanese Windows®, even though you select an English font, the default Japanese font will be effective.

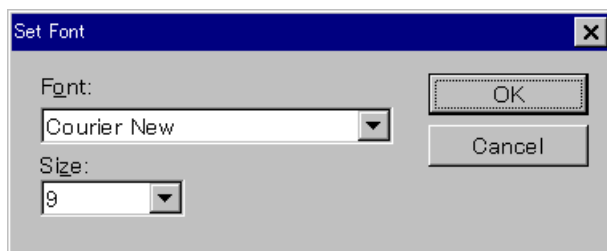


Figure 1.18 Set Font Dialog

1.2.7 The Status Bar

The status bar displays information as to the current state of the HEW. Figure 1.19 shows the six sections of the status bar.

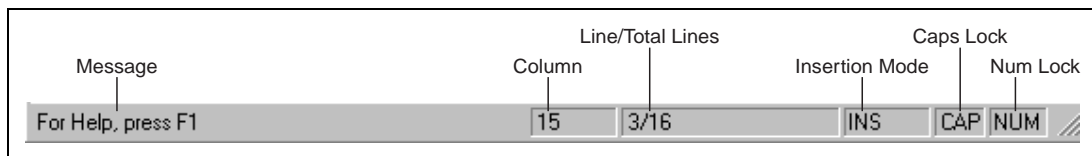


Figure 1.19 Status Bar

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 (as shown in figure 1.20).

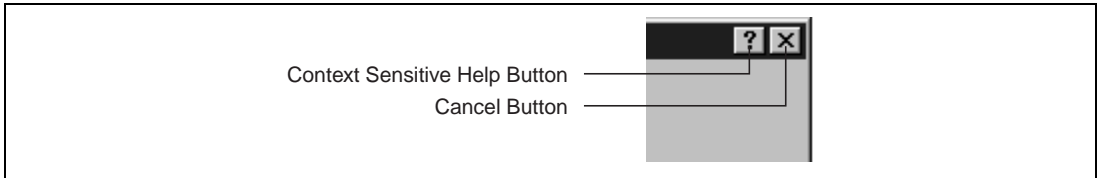


Figure 1.20 Help Button

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 that you require assistance on.

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

1.4 Launching the HEW

To run the HEW, open the “Start” menu of Windows®, select “Programs”, select “Hitachi Embedded Workshop” and then select the shortcut of the Hitachi Embedded Workshop. By default, the “Welcome!” dialog shown in figure 1.21 will be displayed.

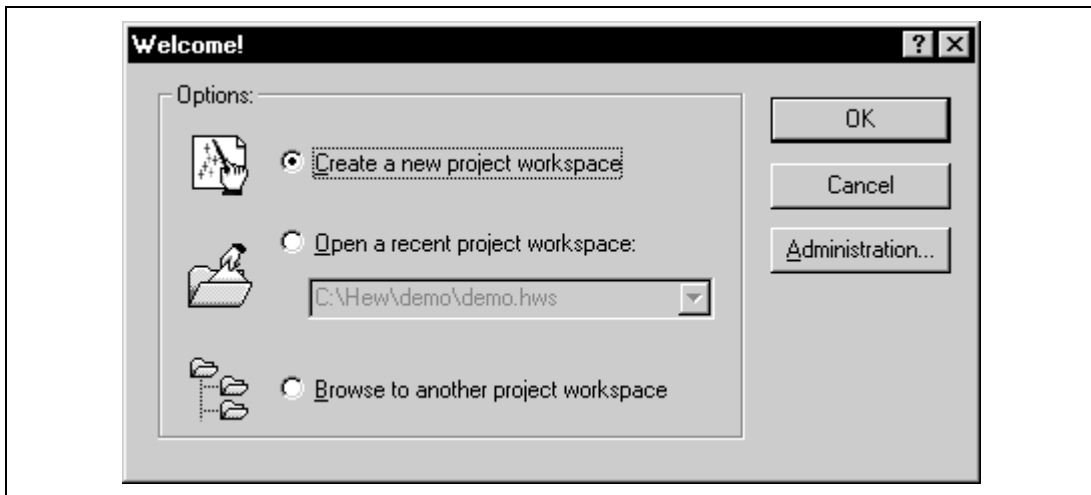


Figure 1.21 Welcome! Dialog

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”. 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 Creating a New Workspace

➡ To create a new workspace:

1. Select the “Create a new project workspace” option from the “Welcome!” dialog (figure 1.21) and press “OK” or select [**File->New Workspace...**]. The “New Project Workspace” dialog will be displayed (figure 1.22).

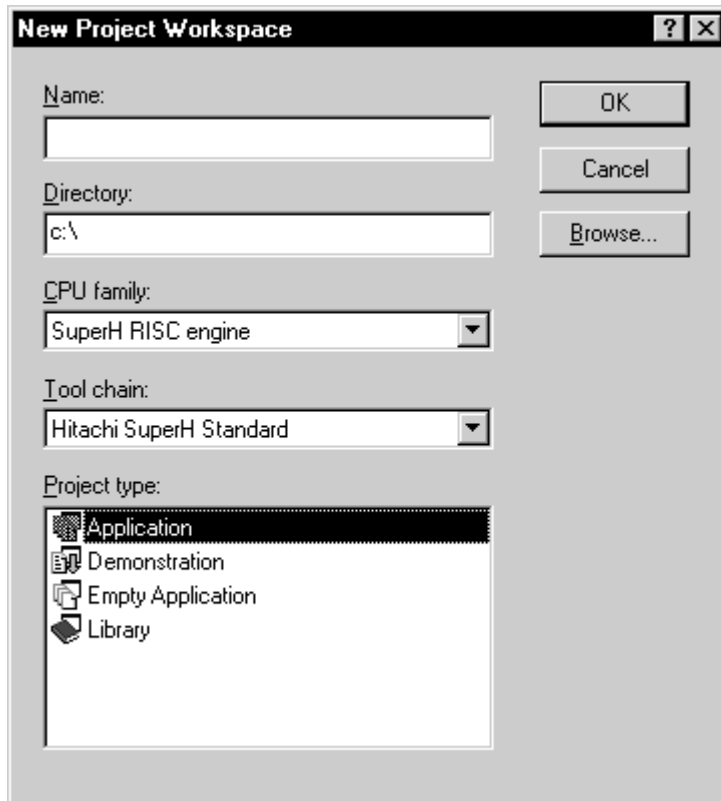


Figure 1.22 New Project Workspace Dialog

2. 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 workspace name the HEW will add a subdirectory for you automatically. This can be changed if desired. Use the “Browse...” button to graphically select the directory in which you would like to create the workspace. Alternatively, you can type the directory into the “Directory” field manually.
3. Select the CPU family and tool chain upon which you would like to base the workspace. Note that this cannot be changed once the workspace has been created.

4. When a new workspace is created, HEW will also create a project with the same name and place it inside the new workspace automatically. The “Project types” list displays all of the available project types (e.g. application, library etc.). Select the type of project that you want to create from this list.
5. Click “OK” to create the new workspace and project.

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

1.6 Opening a Workspace

➡ To open a workspace:

1. Select “Browse to another project workspace” option from the “Welcome!” dialog (figure 1.21) and press “OK” or select [**File->Open Workspace...**]. The “Open Project Workspace” dialog will be invoked.
2. Select the workspace file that you want to open (.HWS files only).
3. Click “Open” to open the workspace.

If the HEW is set-up to display information when a workspace is opened then a workspace properties dialog will be invoked (figure 1.23). Press the “OK” button to open the workspace.

Then, if the version of the toolchain used to build a project of the workspace is missing, the “Toolchain Missing Version” dialog will be launched (figure 1.24). To open the workspace using an existing version of the toolchain, select a version from the “Available versions” drop-down list and click the “OK” button. If you do not want to open the workspace, click the “Cancel” button.

Finally, the workspace will be opened if you have not cancelled opening it.

Note that whether the workspace properties dialog is shown depends on the setting of either the “Show workspace information on workspace open” check box on the workspace properties dialog or the “Display workspace information dialog on opening workspace” check box on the “Workspace” tab of the “Tools Options” dialog. This dialog is described in the “Specifying Workspace Options” section in chapter 6, “*Customizing the Environment*”. Click “OK” to open the workspace. Click “Cancel” to stop opening the workspace.

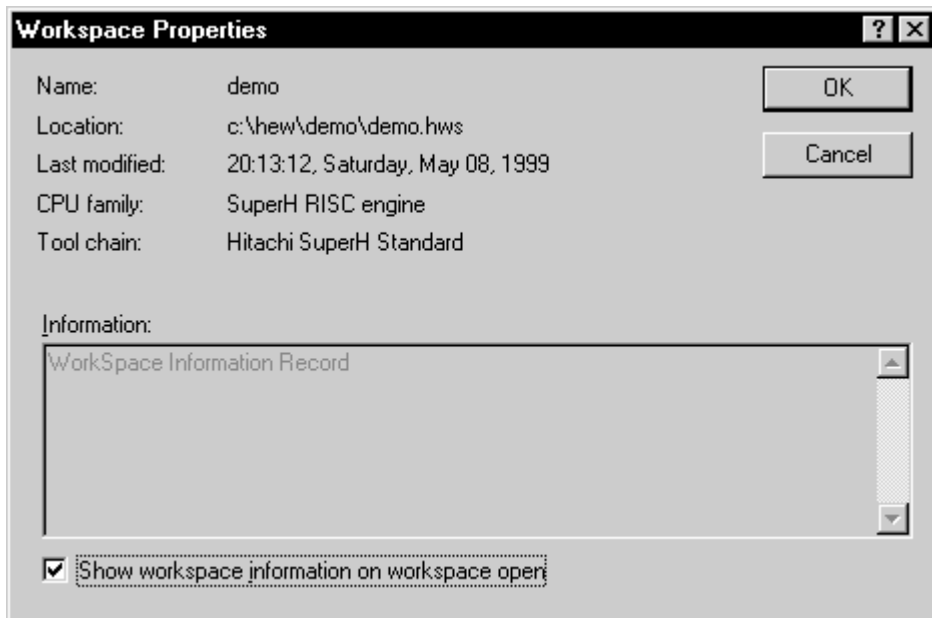


Figure 1.23 Workspace Properties Dialog

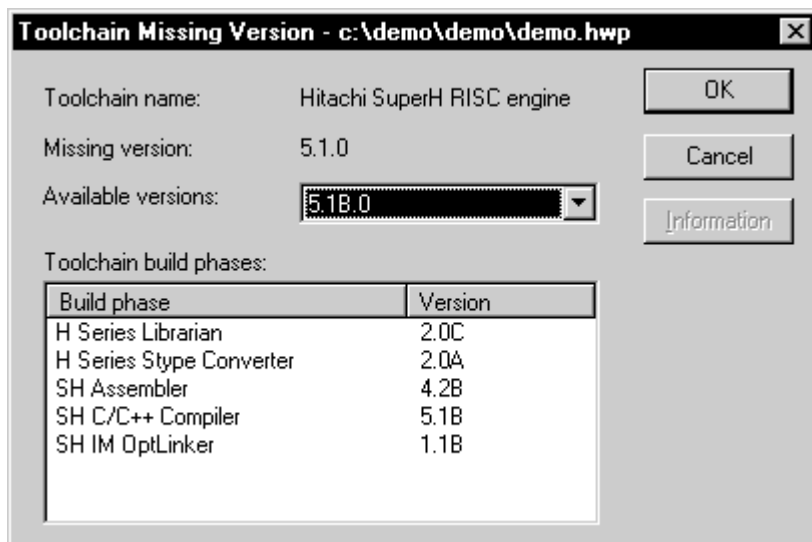


Figure 1.24 Toolchain Missing Version Dialog

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

- ☞ To open a recently used workspace:
1. Select “Open a recent project workspace” from the “Welcome!” dialog, select the name of the workspace from the drop down list and then click “OK”.
- or:
2. Select the [**File->Recent Workspaces**] menu option and from this sub-menu select the name of the workspace.

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

1.7 Saving a Workspace

A HEW workspace can be saved by selecting [**File->Save Workspace**].

1.8 Closing a Workspace

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

1.9 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.

Section 2 Build Basics

This chapter explains the general functions of the HEW whilst the more advanced features can be found in chapter 4, “*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 which 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.

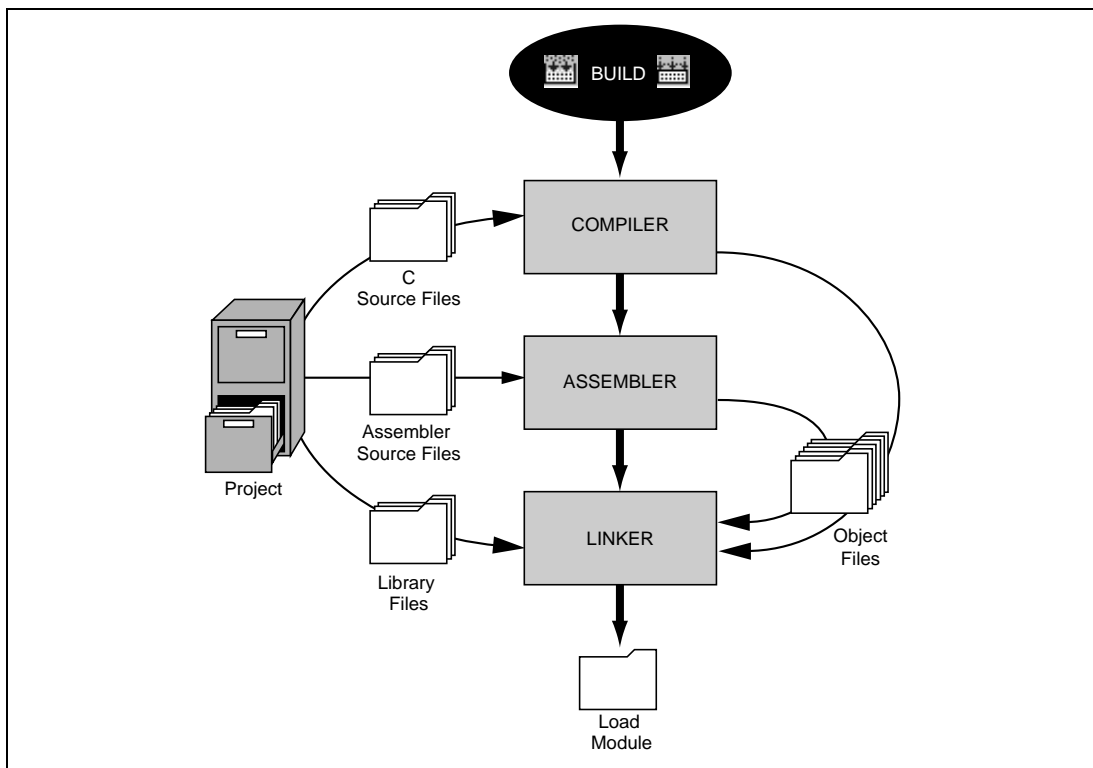


Figure 2.1 Typical Build Process

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 and during the linker phase all library files and output files from the compiler and assembler phases are linked together to produce the load module.

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 4, “*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).

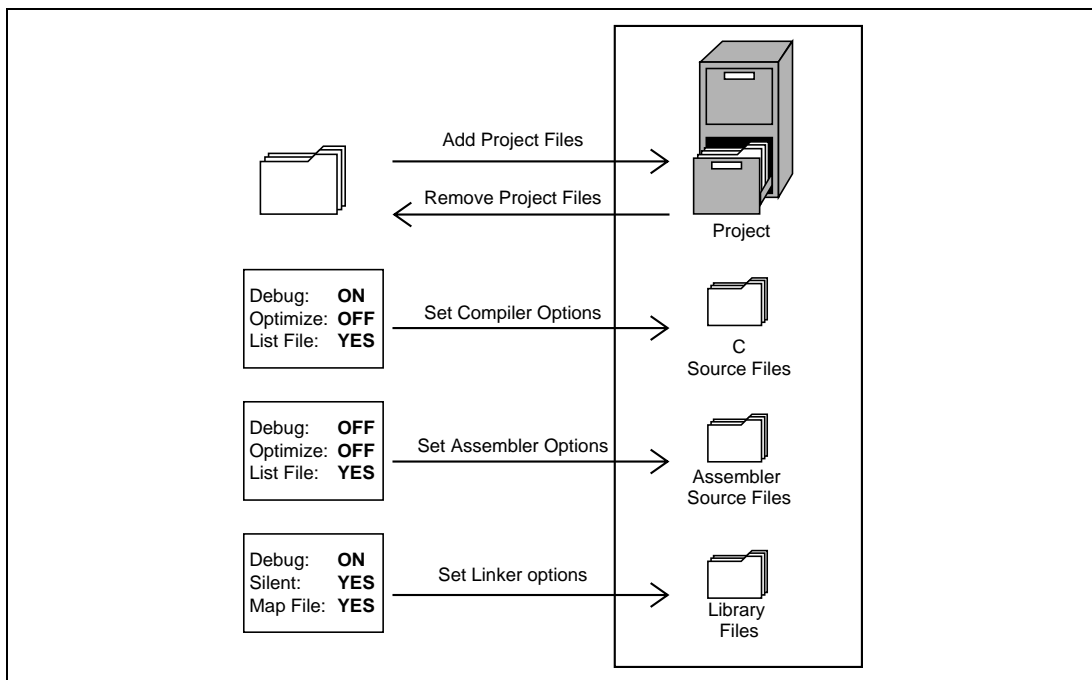


Figure 2.2 Editing a Project

2.2.1 Adding Files to a Project

Before you can build your application you must first inform the Hitachi 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.

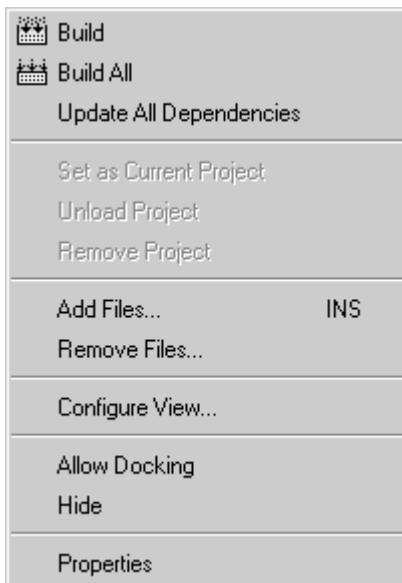


Figure 2.3 Project Pop-up Menu

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

Note: You may only add a file to a project if it is of a recognized type. When you request that a file should be added, the HEW checks the file’s extension and decides whether it is permitted. 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 (see figure 2.4). The “Remove Project Files” dialog will be displayed (figure 2.5).

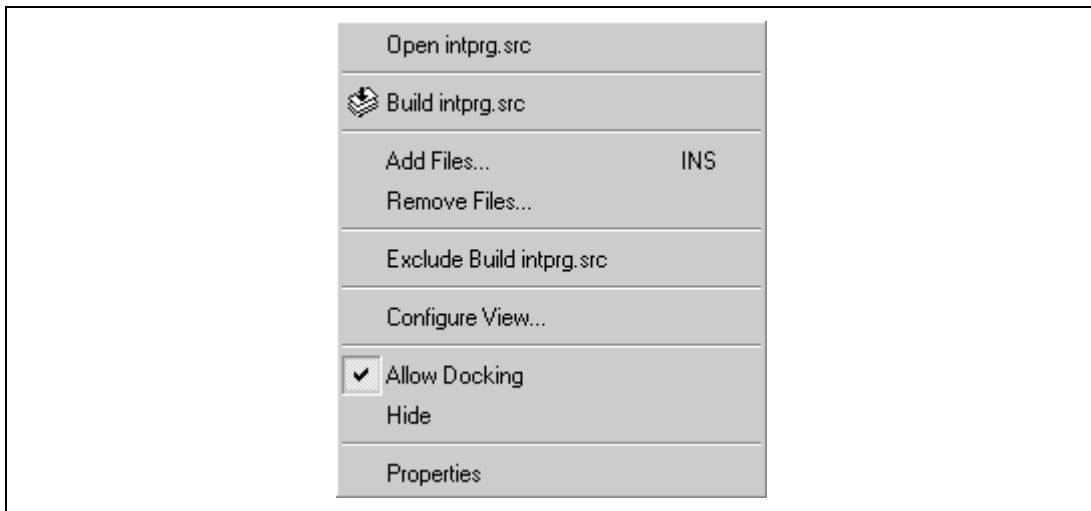


Figure 2.4 Projects Tab Pop-up Menu

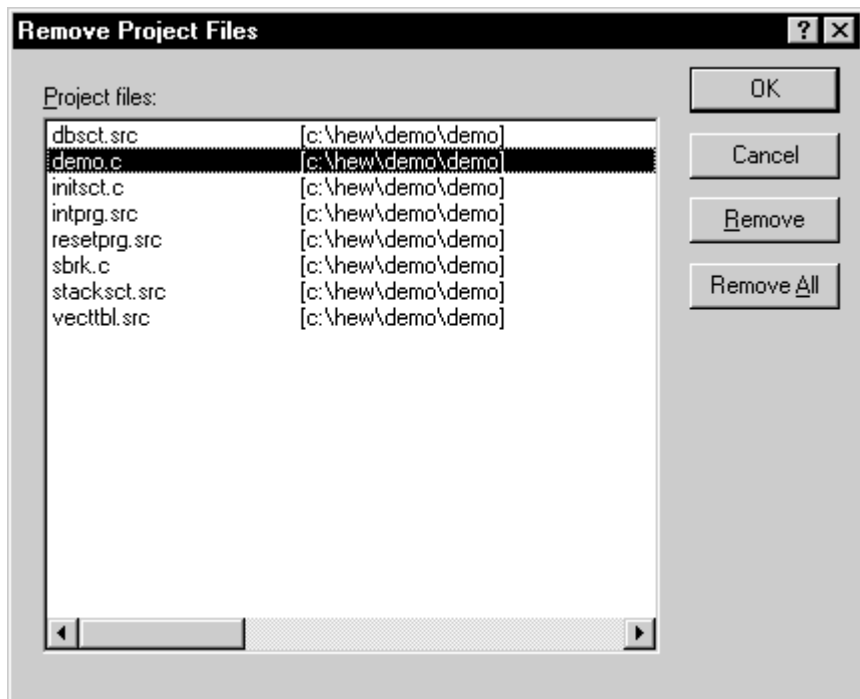


Figure 2.5 Remove Project Files Dialog

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 2.4). 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 (see figure 2.1). 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 for example and if you use a different extension (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 2.6). 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.

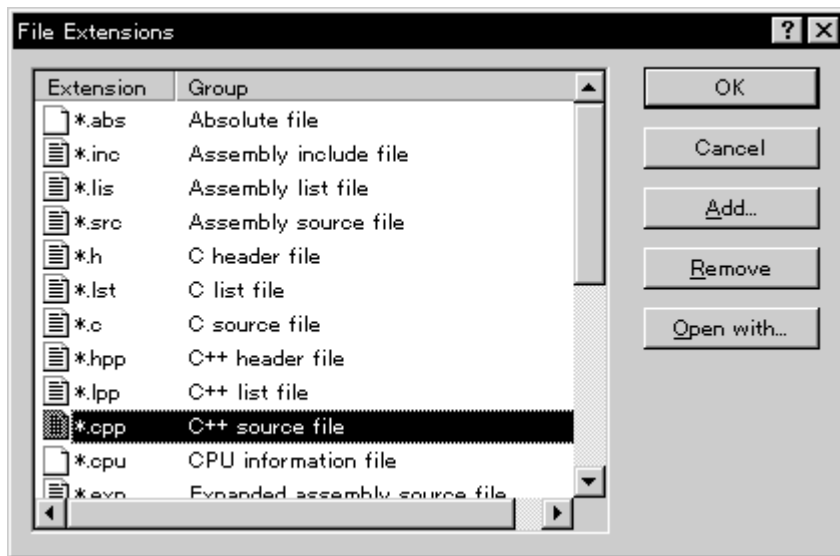


Figure 2.6 File Extensions Dialog

The “File Extensions” list shown in figure 2.6 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 2.7.

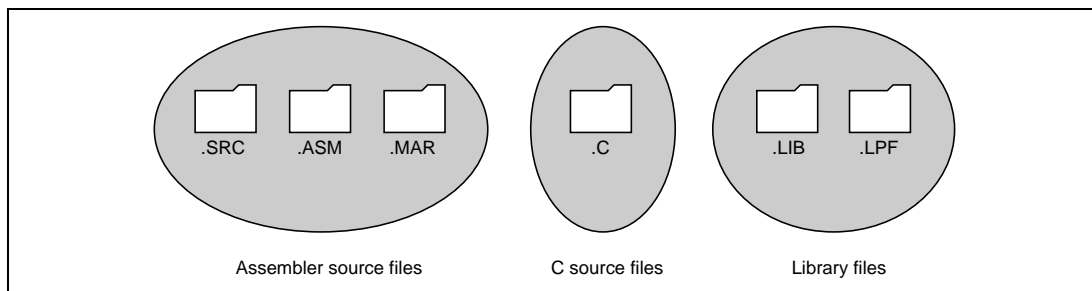


Figure 2.7 File Extensions and Groups

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 2.6).
 2. Click the “Add...” button. The “Define File Extension” dialog will be displayed (figure 2.8).
 3. Enter the extension which you want to define into the “File extension” field. It is not necessary to type the period (.) character.
 4. Select the “New group” option and enter a description which defines this new file group.
 5. Check the “Text based file” if the type of file which has the extension you are defining is text based or leave it unchecked if not.
 6. Click “OK” to add the extension to the “File Extensions” list.

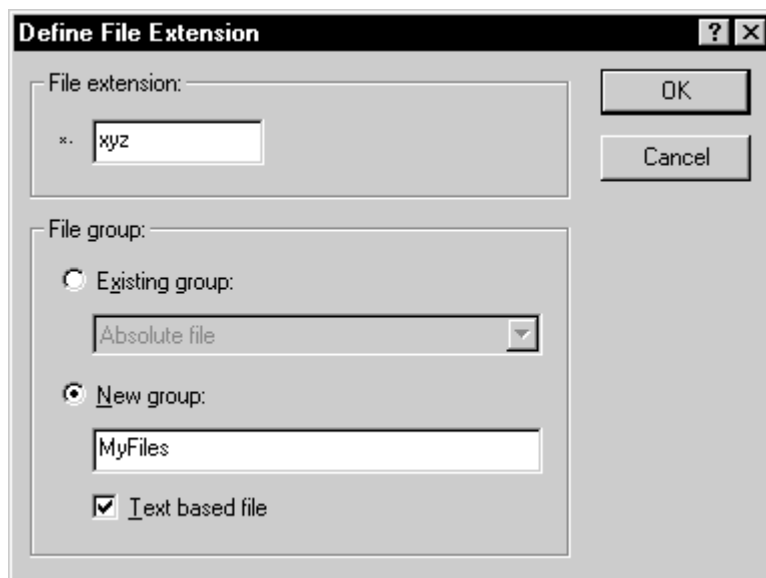


Figure 2.8 Define File Extension Dialog (New Group)

If you want to create a new extension because your files use a different extension from those accepted by the HEW for a given phase (e.g. your assembler source files are .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 2.6).
 2. Click the “Add...” button. The “Define 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.
 4. Select the “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.

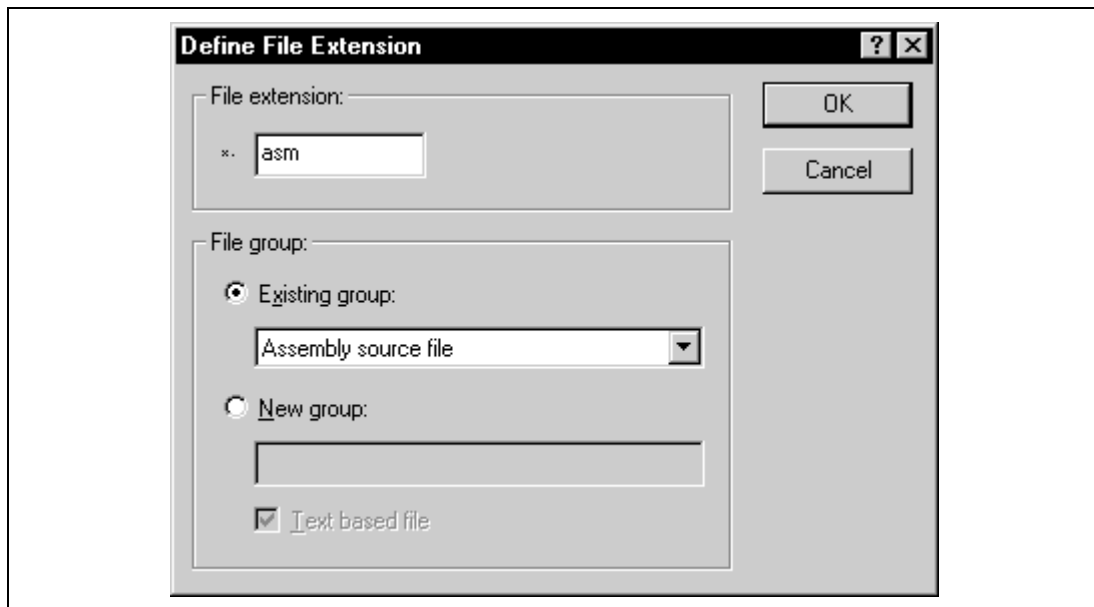


Figure 2.9 Define File Extension Dialog (Existing Group)

When creating a new file group, check the “Text based file” check box to associate files in this group with the editor (i.e. double clicking on the file in the “Projects” tab of the “Workspace” window will open the file in the editor) or do not set it to indicate that the file cannot be opened in the editor. 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 2.10 shows the association between a word processor and the the extension .DOC.

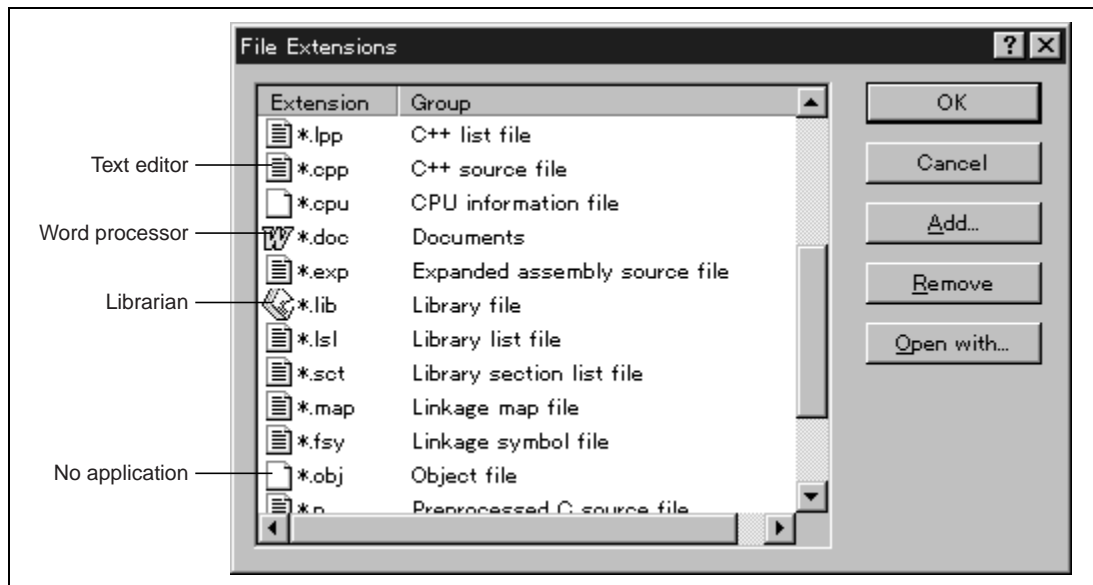


Figure 2.10 File Groups and Applications

- ➔ To associate an application with a file group:
 1. Select the file group to be associated from the “File Extensions” dialog (figure 2.10).
 2. Click the “Open with...” button. The “Associate Application” dialog will be displayed (figure 2.11).

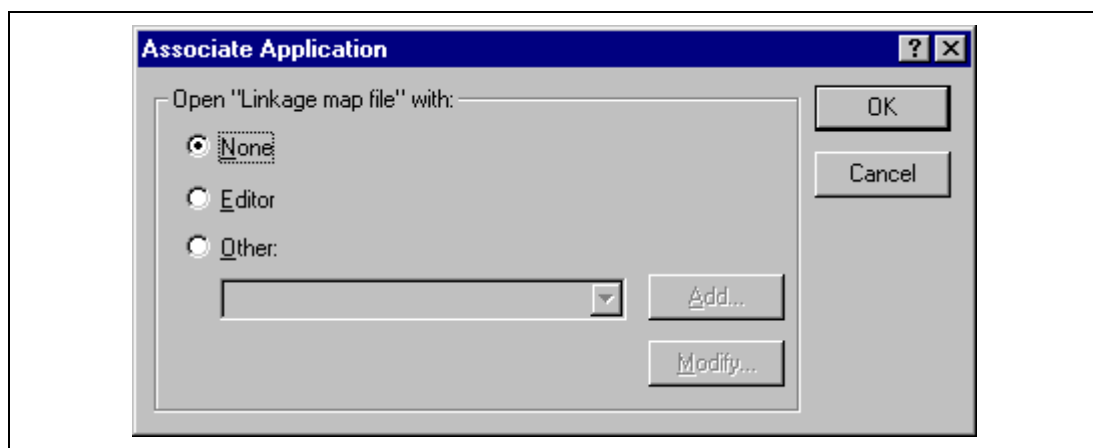


Figure 2.11 Associate Application Dialog

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 2.12).

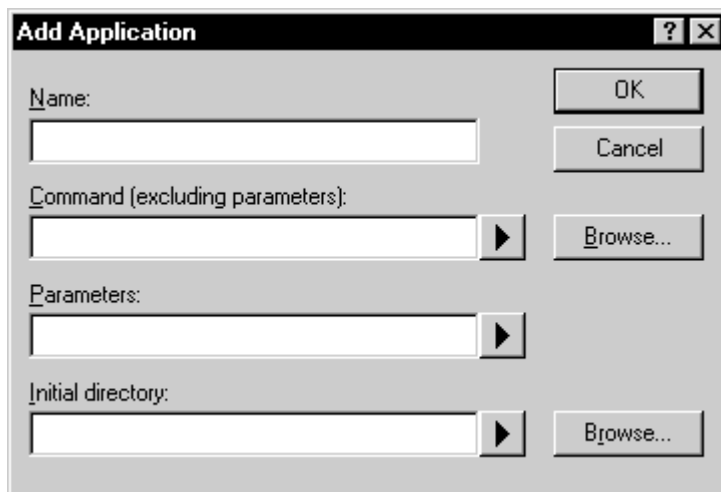


Figure 2.12 Add Application Dialog

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 B, “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 2.13). 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.

Figure 2.13 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.

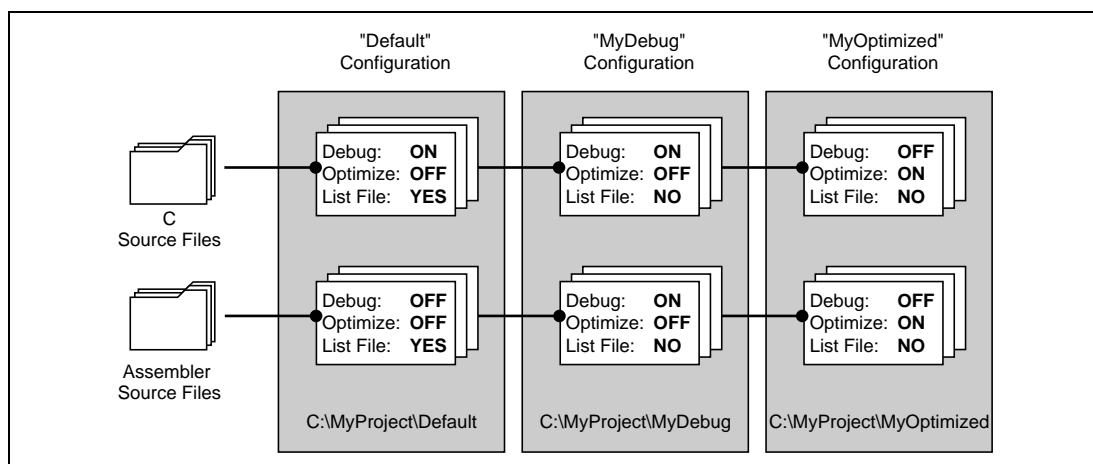


Figure 2.13 Configurations and File Options

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 2.14) in the toolbar.

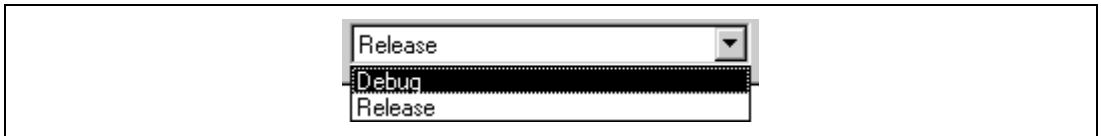


Figure 2.14 Toolbar Selection

or:

1. Select [**Options->Build Configurations...**]. This will invoke the “Build Configurations” Dialog (figure 2.15).

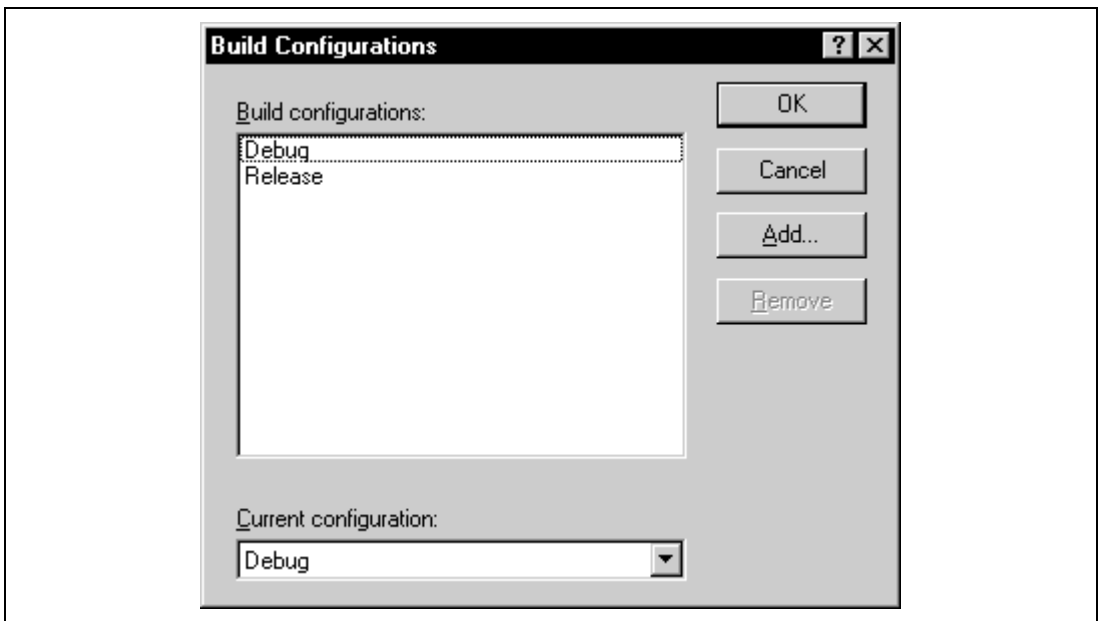


Figure 2.15 Build Configurations Dialog

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 2.15).
2. Click the “Add...” button. The “Add Configuration” dialog will be invoked (figure 2.16).

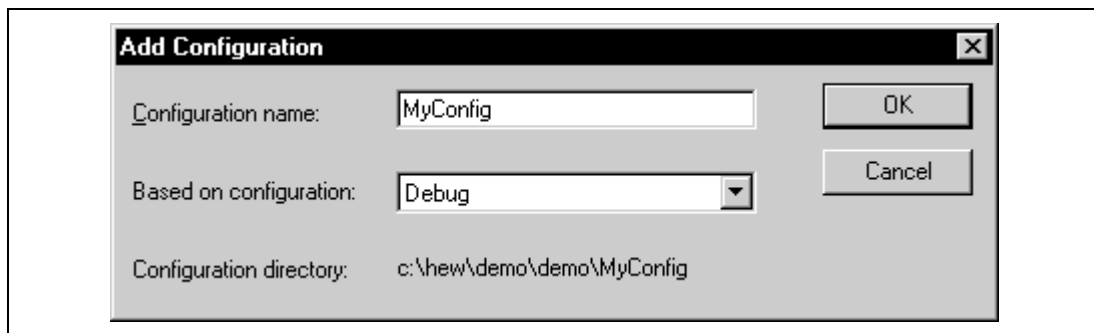


Figure 2.16 Add Configuration Dialog

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 2.15).
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 Hitachi 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 Hitachi 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 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.

2.6.5 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 “Tools Options” dialog will be displayed.
2. Select the “Build” tab (figure 2.17).
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.

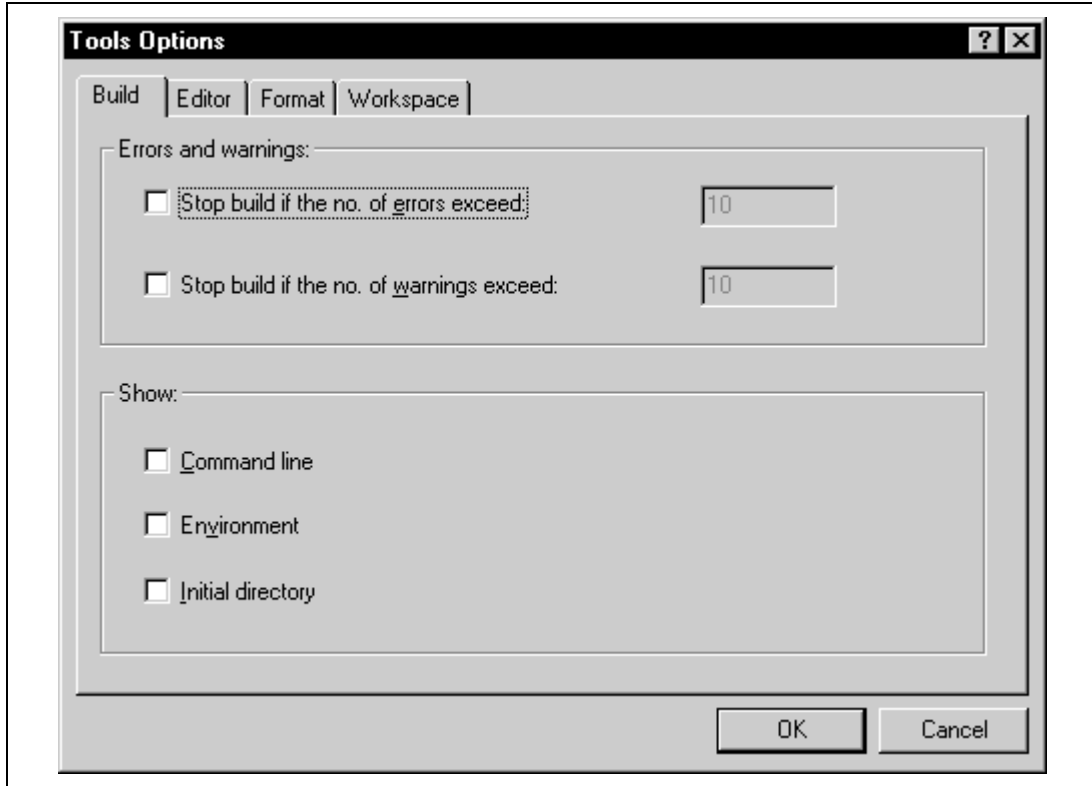


Figure 2.17 Tools Options Dialog, Build Tab

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 2.18.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 2.18.ii). If this option is not selected then a separate folder contains all dependencies (figure 2.18.i).

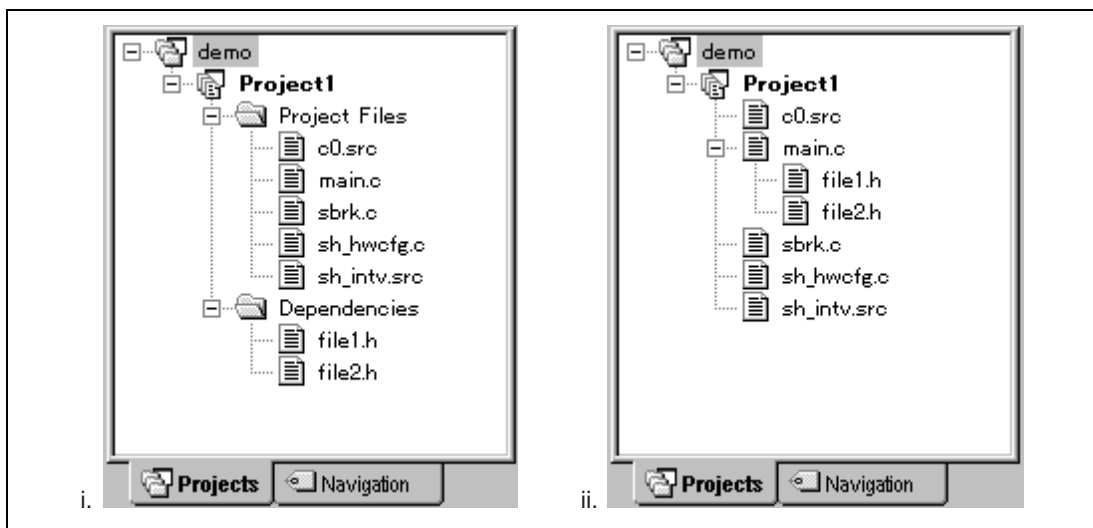


Figure 2.18 Dependencies Under Each File

2.8.2 Show standard library includes

By default, any dependent files found in standard include paths will not be shown (figure 2.19.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 2.19.ii).

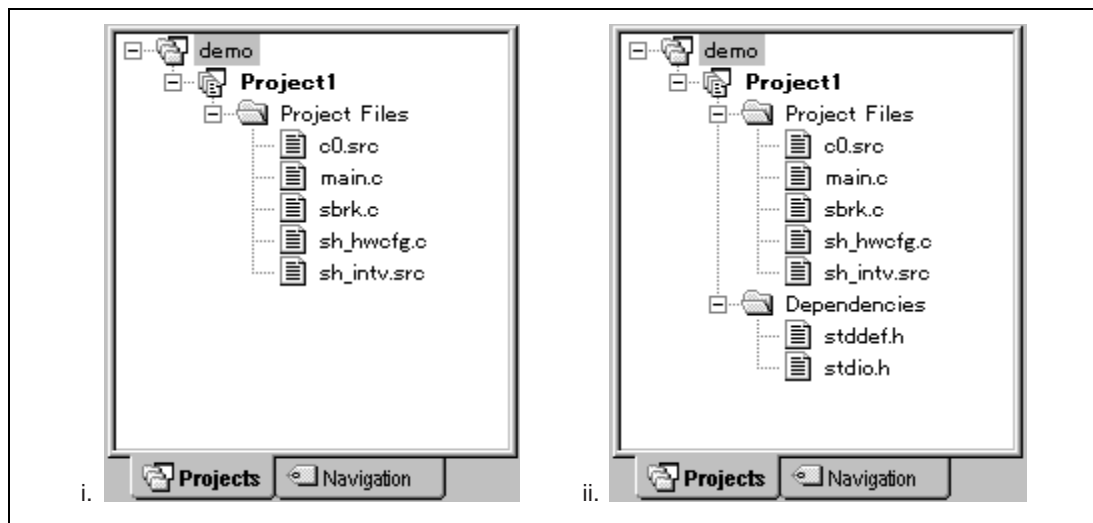


Figure 2.19 Standard Library Includes

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 2.20).

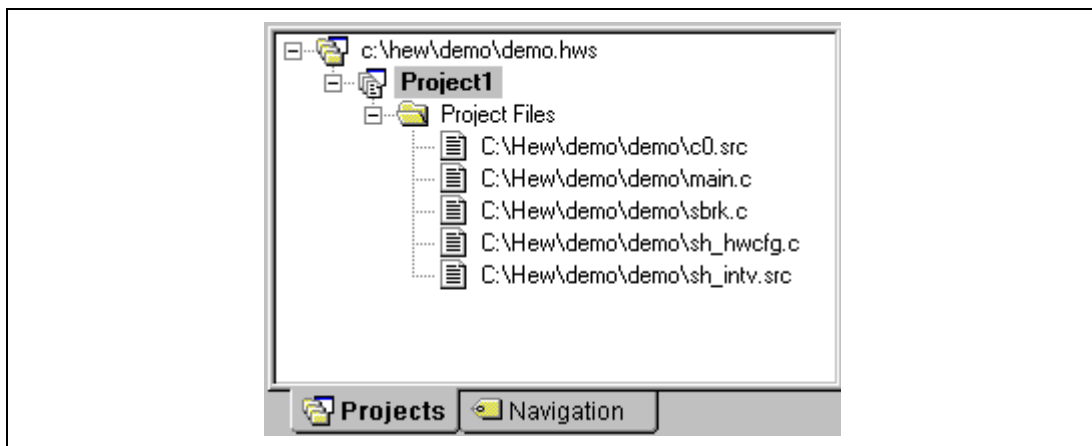


Figure 2.20 File Paths Shown

2.8.4 Show file groups in separate folders

Setting “Show file groups in separate folders”, you can choose to arrange the project files displayed in the workspace so that each file type has its own folder (figure 2.21.ii) or you can choose to display all files in one folder (figure 2.21.i).

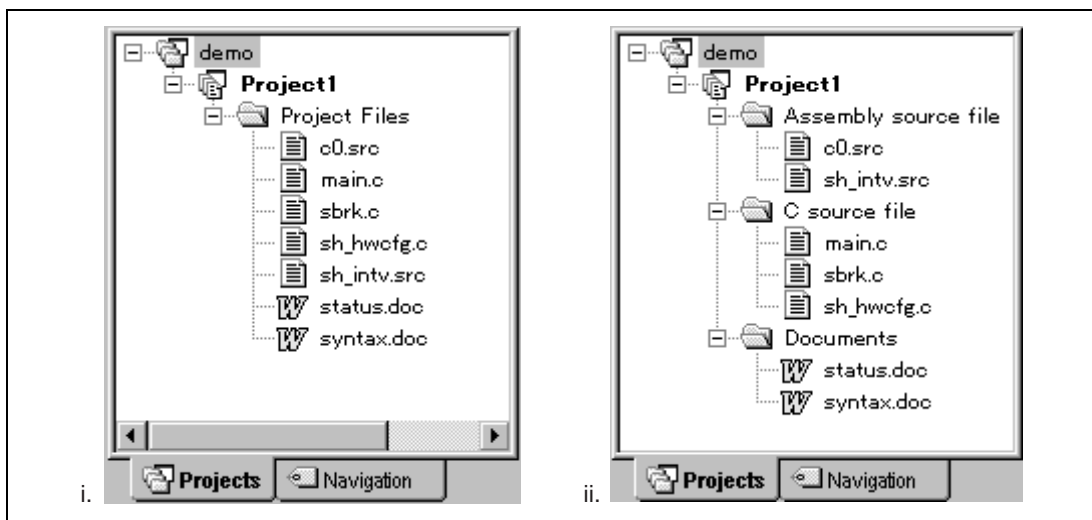


Figure 2.21 Files and Folders

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. When a project is active, its folder can be opened, its project files can be viewed, it can be built and so forth. You can set which project is active in a workspace, and the name of the current project will be shown in bold letters.

☞ To set a project as the current project:

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

or:

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

2.10 Inserting a project into a workspace

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

☞ To insert a new project into a workspace:

1. Select **[Project->Insert Project...]**. The “Insert Project” dialog (figure 2.22) will be displayed.
2. Set the “New Project” option.
3. Click OK. The “Insert New Project” dialog (figure 2.23) 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 and 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).

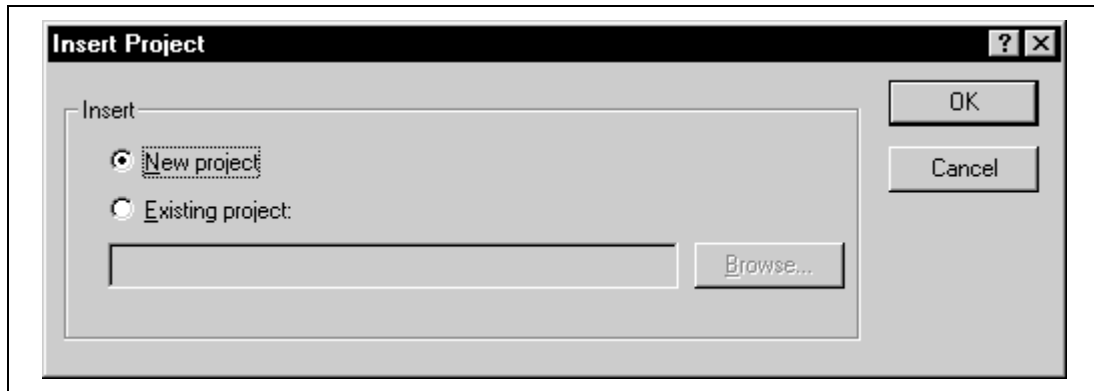


Figure 2.22 Insert Project Dialog



Figure 2.23 Insert New Project Dialog

- ➡ To insert an existing project into a workspace:
1. Select [**Project->Insert Project...**]. The “Insert Project” dialog (figure 2.22) 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 such that when one project is built, all its dependent projects are built first. This is useful if one of the projects in a workspace is used by another. For example, imagine that a workspace contains two projects. The first is a library project which is being developed with the second project, which is an application using the library. In this scenario, the library project could be specified as a dependent (i.e. child) project of the application project so that when the application is built, the library would be built first if it is out-of-date.

When a dependent project is built, the configuration whose name is the same as the current configuration of the current project is selected. If the same configuration name is not in a dependent project, a configuration specified as the current configuration when the project was last closed is selected. In the scenario of the above example, if the application project is the current project, building the application project as the “Debug” configuration will also build the library project as the “Debug” configuration if it exists. If the library project does not have the “Debug” configuration, the configuration selected when it was last closed will be selected.

- ➡ To make projects depend upon another:
1. Select [**Project->Dependent Projects**]. The “Dependent Projects” dialog (figure 2.24) will be displayed.
 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.

Note: Only one level of dependency relative to the current project is effective. If the current project is built, only child projects can be built and grandchild projects will not be built.

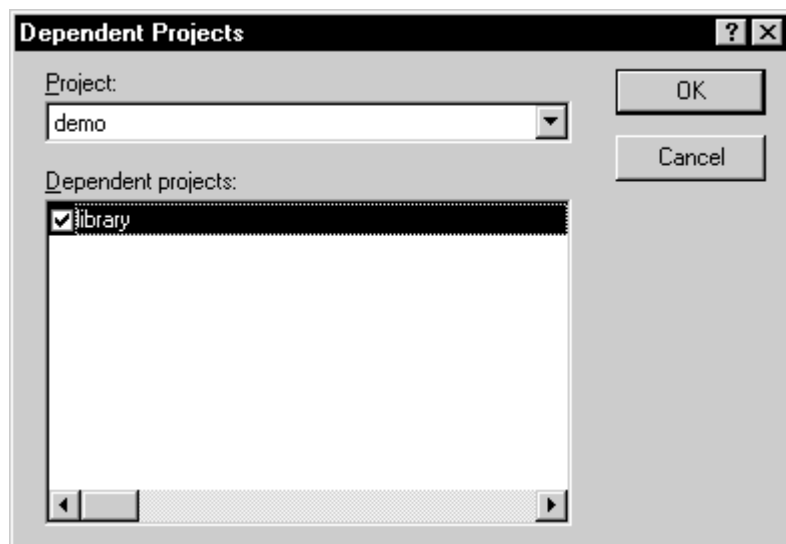


Figure 2.24 Dependent Projects Dialog

2.12 Removing a project from a workspace

➡ To remove a project from a workspace:

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

or:

1. Select the project from the “Projects” tab of the “Workspace” window.
2. Press the **DEL** key.

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

2.13 Loading/unloading a project into/from a workspace

Folders in a project other than the current one can be opened and its project files one can be viewed on the “Projects” tab of the “Workspace” window if you load the project into a workspace. A loaded project can be unloaded from a workspace if the project is neither modified nor set as a current project. The icon of an unloaded project is grayed in the “Projects” tab of the “Workspace” windows.

➡ To load a project into a workspace:

1. Click the right mouse button on the project you want to load on the “Projects” tab of the “Workspace” window. Then a pop-up menu (figure 2.25.i) will be displayed.
2. Select **[Load Project]** from the pop-up menu.

Note: The **[Load Project]** menu item is active only if the project has not been loaded. The icon of such project is grayed.

➡ To unload a project from a workspace:

1. Click the right mouse button on the project you want to unload on the “Projects” tab of the “Workspace” window. Then a pop-up menu (figure 2.25.ii) will be displayed.
2. Select **[Unload Project]** from the pop-up menu.

Note: The **[Unload Project]** menu item is active if the project is loaded and if the project is neither modified nor set as a current project.



Figure 2.25 Project Pop-up Menu

2.14 Renaming a workspace or a project

A workspace or a project can be renamed once you have created it.

☞ To rename a workspace:

1. Click the right mouse button on the workspace in the “Projects” tab of the “Workspace” window. Then a pop-up menu will be displayed.
2. Select **[Properties]** from the pop-up menu. Then the “Workspace Properties” dialog will be launched (figure 2.26).
3. Modify the name of the workspace in the “Name” edit box. The name can be up to 32 characters in length and contain letters, numbers and the underscore character.
4. Click “OK” to rename the workspace.

Note: Even though you rename the workspace, the directory name of the workspace will not be renamed.

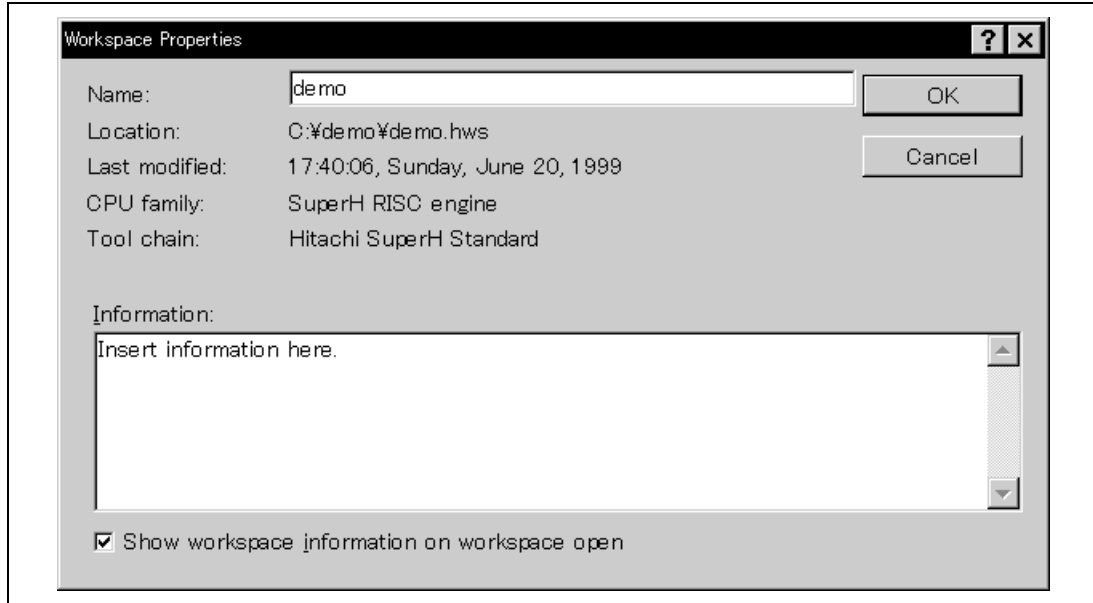


Figure 2.26 Workspace Properties Dialog

- ➡ To rename a project:
1. Click the right mouse button on the project in the “Projects” tab of the “Workspace” window. Then a pop-up menu will be displayed.
 2. Select **[Properties]** from the pop-up menu. Then the “Properties” dialog will be launched (figure 2.27).
 3. Modify the name of the project in the “Name” edit box. The name can be up to 32 characters in length and contain letters, numbers and the underscore character.
 4. Click “OK” to rename the workspace.

Note: Even though you rename the project, the directory name of the project will not be renamed.

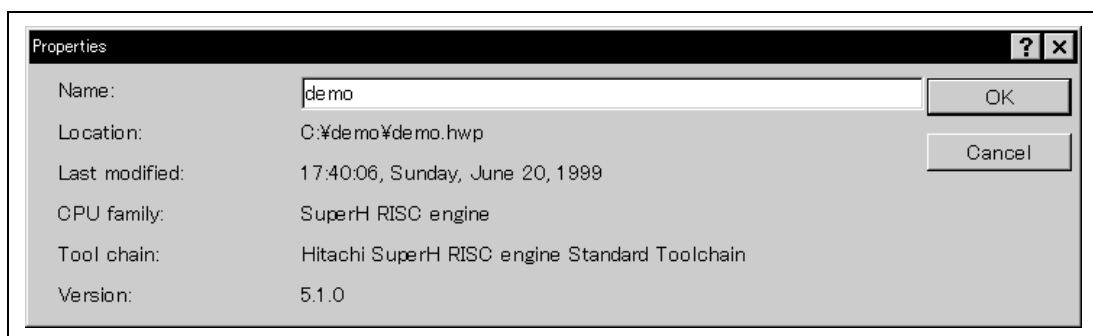


Figure 2.27 Project Properties Dialog

Section 3 Using the Editor

This chapter describes how to use the editor that is provided with the Hitachi Embedded Workshop.

3.1 The Editor Window

The editor window (figure 3.1) 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 (“sbrk.c” is the active window in figure 3.1). All text operations such as typing, pasting text and so forth only affect the active window. To switch to another window (i.e. to make some other window the active window) 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 or 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.

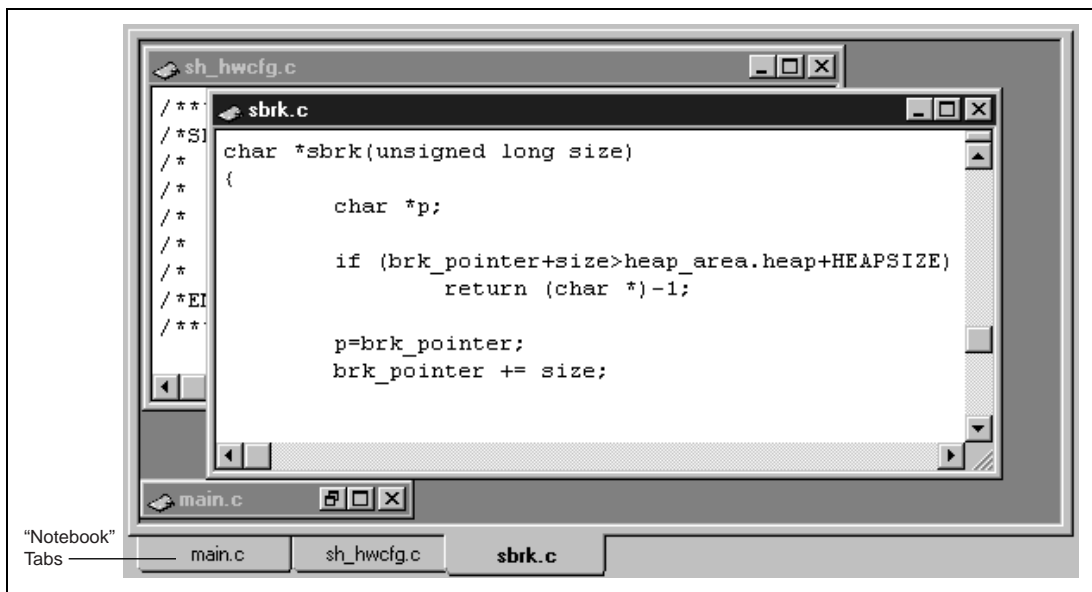


Figure 3.1 Editor Window

3.2 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 which 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]**
Arrange all open windows so that they occupy the entire editor window, without any overlaps.
- **[Window->Arrange Icons]**
Line up all iconized 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 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.

3.3 The Editor Toolbars

The editor has four related toolbars: Editor, Search, Bookmarks and Templates (figures 1.3, 1.5-1.7). They provide a shortcut to the functions of the editor which you will use most often. The following sections describe each buttons function.

3.3.1 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 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 will be invoked. Select a template number and then click OK. For further information on templates, refer to the “*Templates*” section later in this chapter.

Toggle Bookmark

The Hitachi Embedded Workshop editor provides standard bookmark capabilities. To set a bookmark, select the line to mark and click this button (the line will be highlighted in green by default). To remove a bookmark, select the line to remove a bookmark and click this button (the line will return to normal text). For further information on bookmarks, refer to the “*Bookmarks*” section later in this chapter.

3.3.2 Search Toolbar Buttons

Find Next

Finds the next occurrence of the current search string.

Find Previous

Finds the previous occurrence of the current search string.

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.

3.3.3 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.

3.3.4 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.

3.4 Standard File Operations


3.4.1 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.


3.4.2 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 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 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 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.

3.4.3 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 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 will be displayed).

3.4.4 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 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 Hitachi 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).

3.4.5 Closing Files

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

3.5 Editing a File

The Hitachi 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 3.1 outlines the basic operations that are provided by the editor.

Table 3.1 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 |

3.6 Searching and Navigating Through Files

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

3.6.1 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 (img). The "Find" dialog will be displayed (figure 3.2).

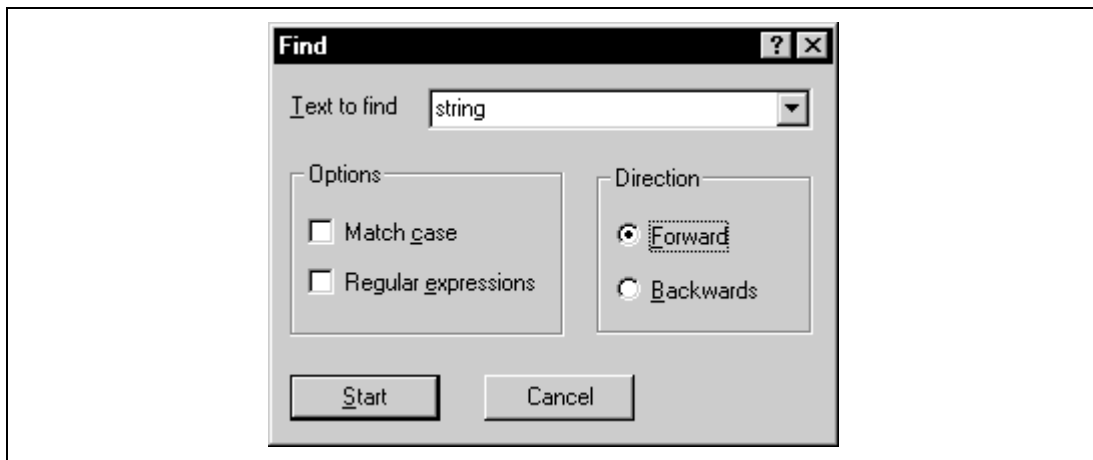


Figure 3.2 Find Dialog

4. Enter the text that you want to search for into the "Text to 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 "Text to find" field.
5. 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.
6. If your search string uses regular expressions then check the "Regular expressions" check box. Refer to Appendix A, "*Regular Expressions*" for further information.
7. The "Direction" radio buttons allow you to select the direction of the search. Selecting "Forward" means that the search will be performed from the insertion cursor towards the bottom of the file. Selecting "Backwards" means that the search will be performed from the insertion cursor towards the top of the file.



- Click the “Start” button to begin the search. If the string is found the find strip dialog will be displayed (figure 3.3). Click “Find Next” or the find next toolbar button () to move on to the next occurrence of the search string. Click “Find Previous” or the find previous toolbar button () to move to the previous occurrence of the search string. Click “Cancel” to stop the find action.




Figure 3.3 Find Strip Dialog

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

3.6.2 Finding Text in Multiple Files

➡ To search for text in many files:

- 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 will be displayed (figure 3.4).

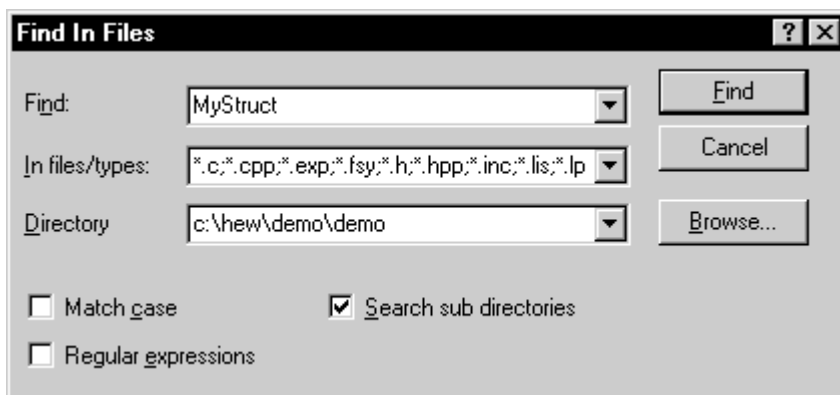


Figure 3.4 Find in Files Dialog

- 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.
- 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).
- 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 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 A, “*Regular Expressions*” for further information.
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.

3.6.3 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 will be displayed (figure 3.5).
4. Enter the text that you want to search for into the “Text to find” 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 “Text to find” field.
5. Enter the text that you want to replace the search string with, or select a previous replace string from the drop-down list box.

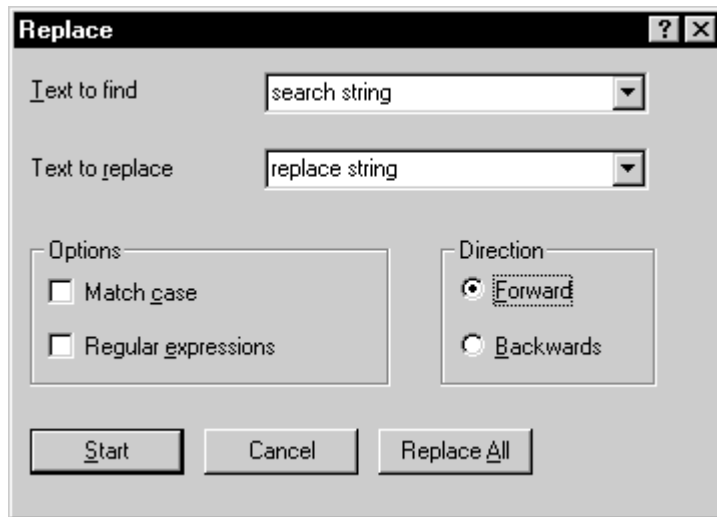


Figure 3.5 Replace Dialog

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 A, “*Regular Expressions*” for further information.
8. The “Direction” radio buttons allow you to select the direction of the search. Selecting “Forward” means that the search will be performed from the insertion cursor towards the bottom of the file. Selecting “Backwards” means that the search will be performed from the insertion cursor towards the top of the file.
9. Click the “Start” button to run the search interactively (i.e. to be prompted upon each replace) or click “Replace All” to replace all occurrences of the search string (with no prompts).
10. If you clicked “Start”, the editor will search for the first occurrence of the search string. If it finds an occurrence then the replace confirm dialog is displayed (figure 3.6). Click “Ignore” if you do not want to replace the current 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.

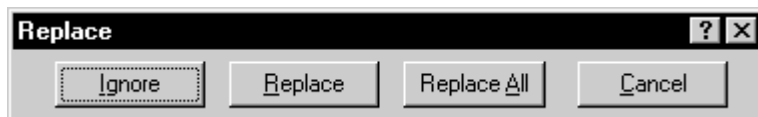


Figure 3.6 Replace Confirm Dialog

3.6.4 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 will be displayed (figure 3.7).
3. Enter into the dialog 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.

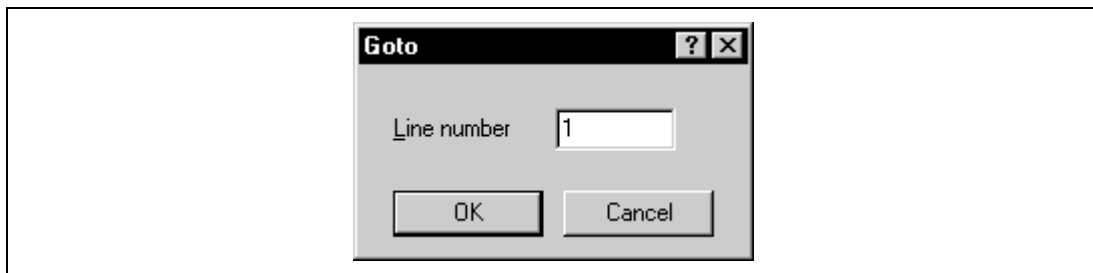



Figure 3.7 Goto Line Dialog


3.7 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. The line will become highlighted to indicate that it is 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 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 (.

3.8 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**.

3.9 Configuring Text Layout

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

3.9.1 Page Set-up

When you print a file from the Hitachi Embedded Workshop editor, the settings in the print dialog 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 3.8).
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.

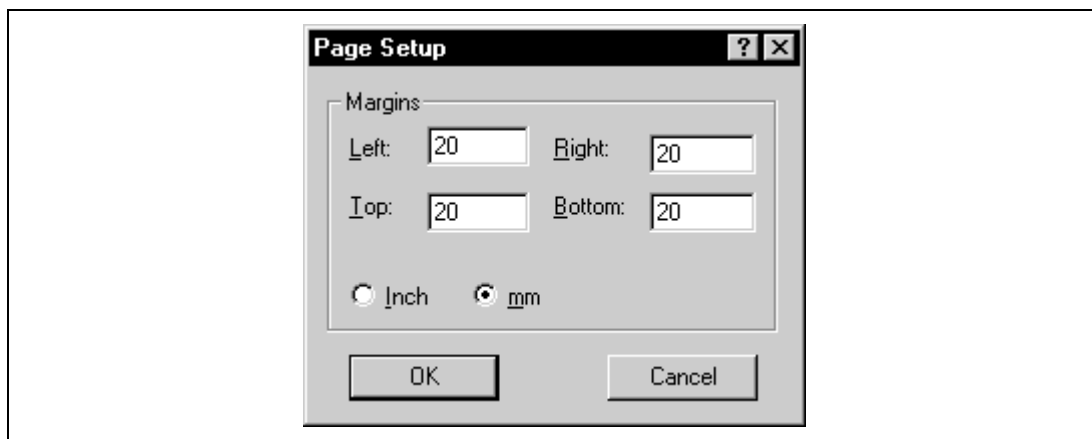


Figure 3.8 Page Setup Dialog

3.9.2 Changing Tabs

➡ To change tab size:

1. Select [**Tools->Options...**]. The tools options dialog will be displayed. Select the “Editor” tab (figure 3.9).
2. Enter into the “Tab size” field the number of desired tabs.
3. Click “OK” for the tab setting specified to take effect.

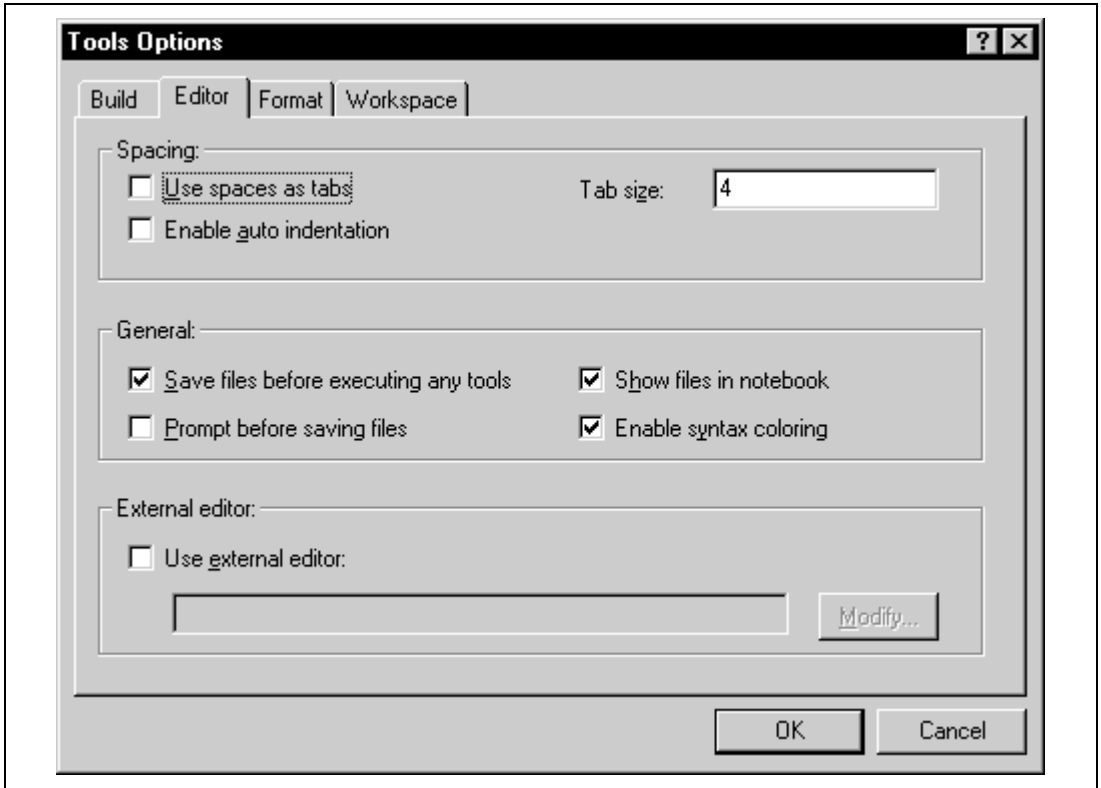


Figure 3.9 Tools Options Dialog, Editor Tab

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 “Tools Options” dialog.

➡ To use spaces as tabs:

1. Select [**Tools->Options...**]. The tools options dialog will be displayed. Select the “Editor” tab (figure 3.9).
2. Set the “Use spaces as tabs” check box as appropriate.
3. Click “OK” for the tab setting specified to take effect.

3.10 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 3.10 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).

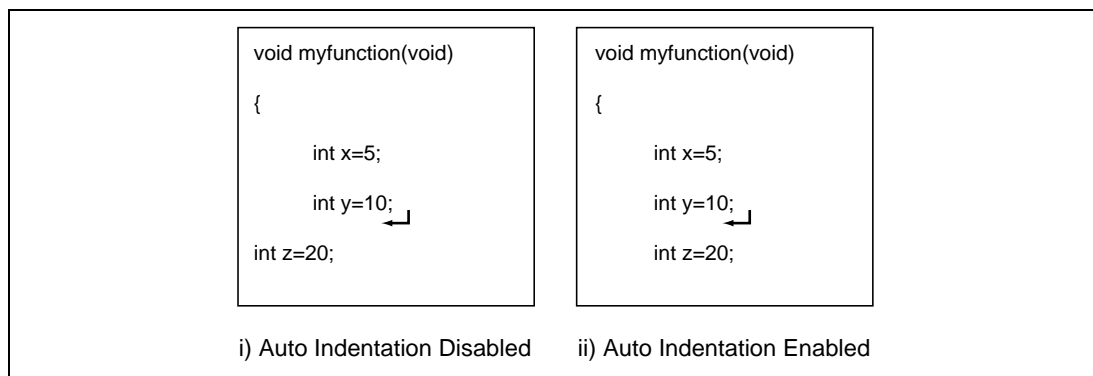


Figure 3.10 Effect of Auto Indentation

☞ To enable/disable Auto Indentation:

1. Select [**Tools->Options...**]. The "Tools Options" dialog will be displayed. Select the "Editor" tab (figure 3.9).
2. Set the "Enable auto indentation" check box accordingly.
3. Click "OK" for the setting of the auto indentation check box to take effect.

3.11 Splitting a Window

The Hitachi Embedded Workshop editor allows you to split a text window into two. Figure 3.11 shows the split bar button which is located just underneath the maximize button at the top right hand corner of any text window.

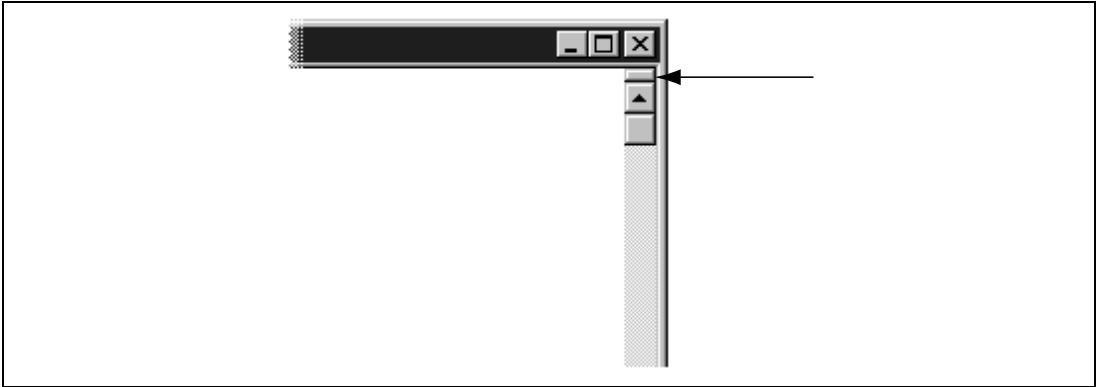


Figure 3.11 Split Bar Button

- 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, 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.

3.12 Configuring Text

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

3.12.1 Changing the Editor Font

The Hitachi 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->Options...**]. The “Tools Options” dialog will be displayed. Select the “Format” tab (figure 3.12).
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.

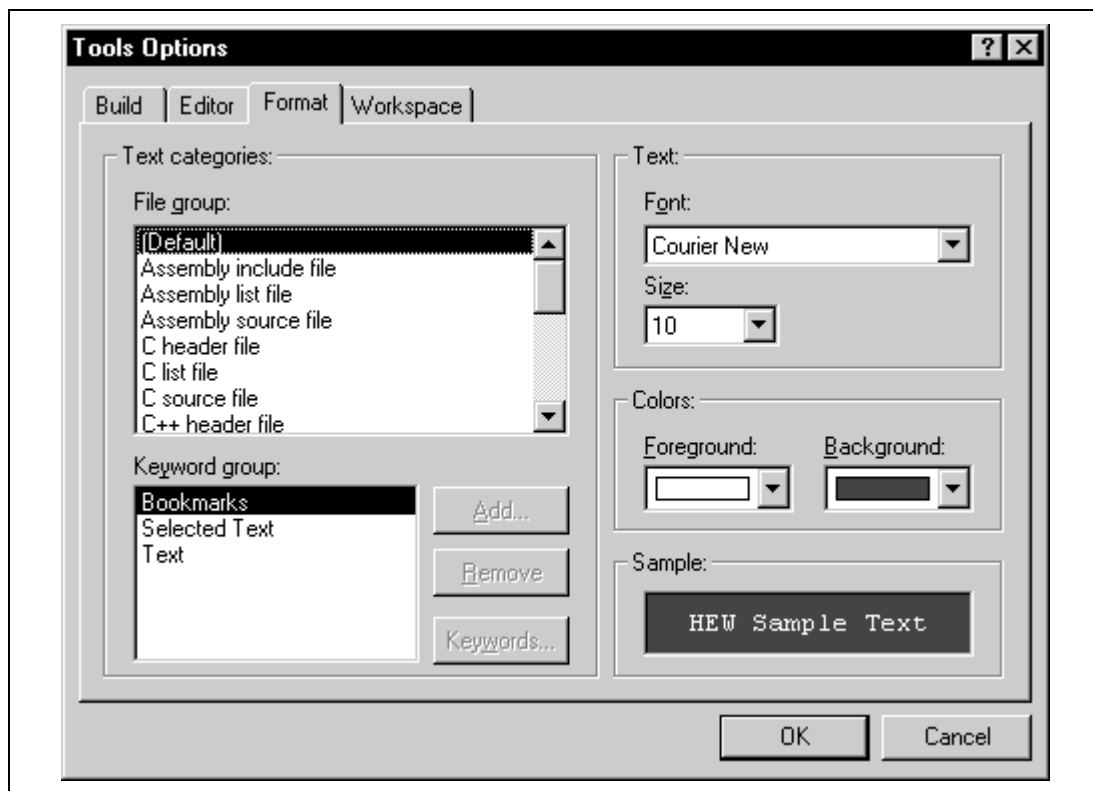


Figure 3.12 Tools Options Dialog, Format Tab

3.13 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.

☞ To enable syntax coloring:

1. Select [**Tools->Options...**]. The “Tools Options” dialog will be displayed. Select the “Editor” tab.
2. Check the “Enable syntax coloring” checkbox.
3. Click “OK” to enable syntax coloring.

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->Options...**]. The “Tools Options” dialog will be displayed. Select the “Format” tab (figure 3.12).
2. Select the file type for which you want to edit syntax coloring from the “File group” list. As you do this, the content of the “Keyword group” list will change to reflect the current groups.
3. Select the keyword group from the “Keyword group” list. The selections in the “Foreground” and “Background” lists will change to reflect the current colors for the selected keyword group.
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 and keywords:

1. Select [**Tools->Options...**]. The “Tools Options” dialog will be displayed. Select the “Format” tab (figure 3.12).
2. Select the file type for which you want to edit syntax coloring from the “File group” list. As you do this, the content of the “Keyword group” list will change to reflect the current groups.
3. Click “Add...” to create a new keyword group. Then the “Define Keywords” dialog will be displayed (figure 3.13).



Figure 3.13 Define Keywords Dialog

4. Enter the name of the group into the “Keyword group” field.
5. Click the “Add...” button to add a keyword. Then the “Add Keyword” dialog (figure 3.14) will be launched. Specify a keyword in the “Keyword” field and click “OK” to close the dialog. To remove a keyword from the “Keywords” list of the “Define Keywords” dialog, select the keyword and click the “Remove” button.

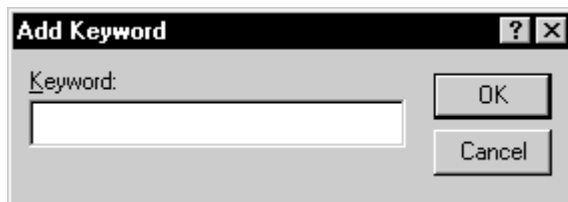


Figure 3.14 Add Keyword Dialog

6. Click “OK” on the “Define Keyword” dialog to confirm the changes to the keyword group.
7. Selecting the new keyword group, modify the “Foreground” and “Background” color lists as desired on the “Format” tab. The color “System” refers to the current window foreground and background settings in control panel.
8. Click “OK” on the “Tools Options” dialog to confirm all of the changes to the syntax coloring.

- ➡ To modify keywords of an existing keyword group:
1. Select [**Tools->Options...**]. The “Tools Options” dialog will be displayed. Select the “Format” tab (figure 3.12).
 2. Select the file type for which you want to edit syntax coloring from the “File group” list. As you do this, the content of the “Keyword group” list will change to reflect the current groups.
 3. Select the desired keyword group to be modified and click the “Keywords...” button. Then the “Define Keywords” dialog will be displayed (figure 3.15).

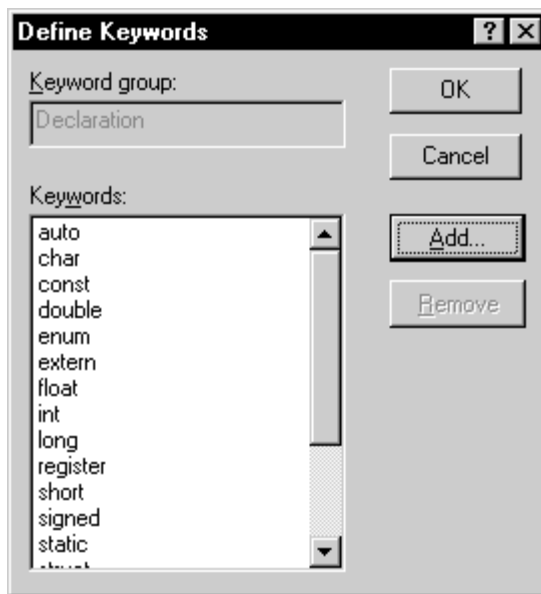


Figure 3.15 Define Keywords Dialog

4. Click the “Add...” button to add a keyword. Then the “Add Keyword” dialog (figure 3.14) will be launched. Specify a keyword in the “Keyword” field and click “OK” to close the dialog. To remove a keyword from the “Keywords” list of the “Define Keywords” dialog, select the keyword and click the “Remove” button.
5. Click “OK” on the “Define Keyword” dialog to confirm the changes to the keyword group.
6. Click “OK” on the “Tools Options” dialog to confirm all of the changes to the syntax coloring.


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 “Tools Options” dialog will be displayed. Select the “Editor” tab (figure 3.9).
 2. Set the “Enable syntax coloring” check box as necessary and then click “OK”.

3.14 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 Hitachi 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.

3.14.1 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 the define template bookmark toolbar button (). The dialog shown in figure 3.16 will be displayed.
 2. Use the “Template number” drop down menu to select which template you want to modify (a maximum of 10 templates can be defined).
 3. 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**.
 4. Enter the following keywords to insert special information when the template is inserted:

| | |
|-------------|---|
| [TIME] | The current time |
| [DATE_DMY] | The current date in day/month/year form |
| [DATE_MDY] | The current date in month/day/year form |
| [DATE_YMD] | The current date in year/month/day form |
| [DATE_TEXT] | The current date in textual form |
 5. Enter the ^ character to specify where the insertion cursor is to be placed after the template has been inserted. If this is not specified then the insertion cursor will be placed after the last character in the template (as in a normal paste operation).

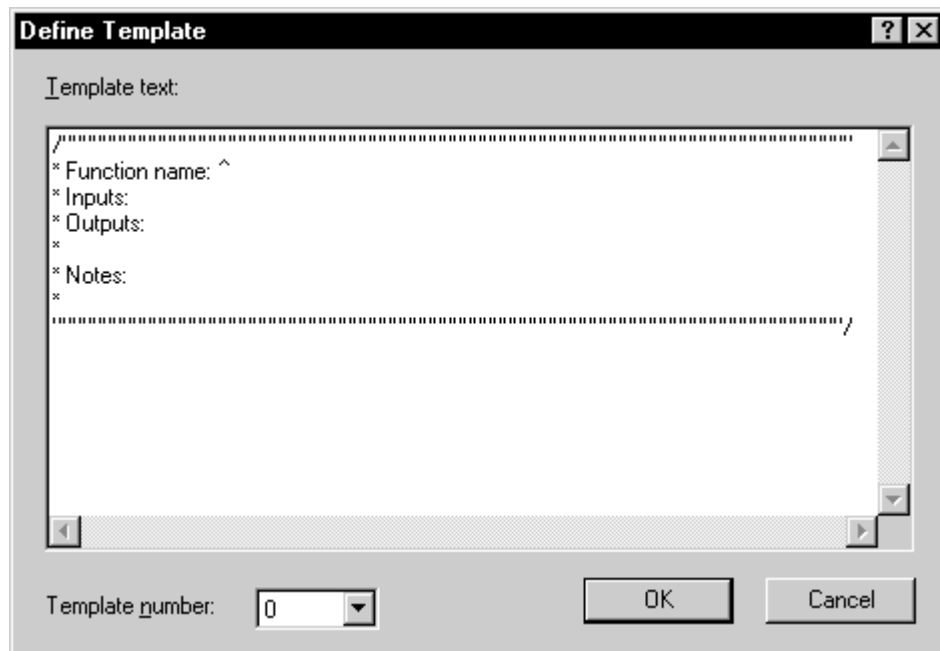


Figure 3.16 Define Template Dialog

3.14.2 Inserting a Template

➡ To insert a template:

1. Click the insert template toolbar button (📄), select [**Edit-> Templates->Insert Template...**] or select [**Templates-> Insert Template...**] from the local menu. The “Insert Template” dialog will be displayed (figure 3.17). You can also insert a template by pressing **ALT** along with the number of the template to insert (e.g. **ALT+4** to insert template 4). Finally, you can insert a template by selecting it from the drop-down list on the templates toolbar and then clicking the insert template toolbar button (📄).

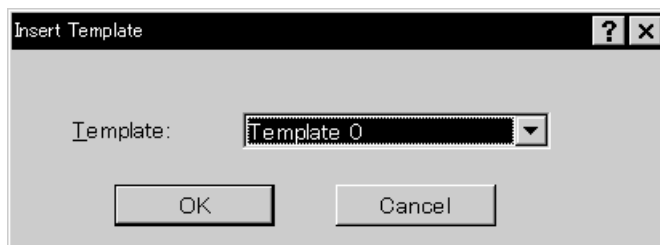


Figure 3.17 Insert Template Dialog

2. Use the “Template” drop down list to select the number of the template and then click “OK”.

3.15 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 Hitachi Embedded Workshop editor provides a match brace feature which highlights text between braces of type { }, () and [] (see figure 3.18).

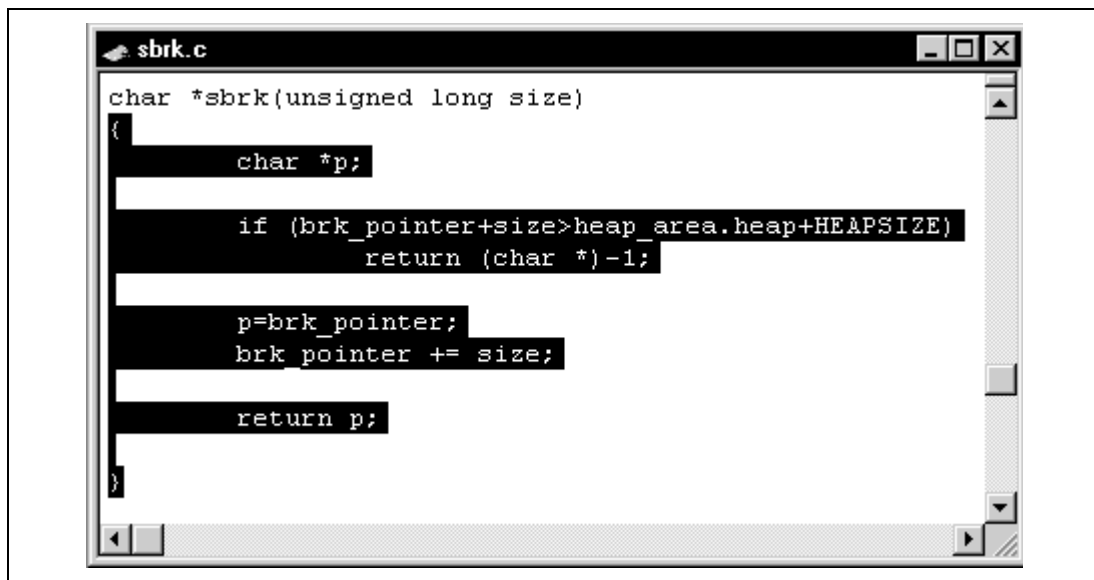



Figure 3.18 Brace Match

- ☞ 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.

Section 4 Advanced Build Features

This chapter explains the more advanced build concepts.

4.1 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 Hitachi 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.

4.1.1 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 4.1 presents another view of the build process.

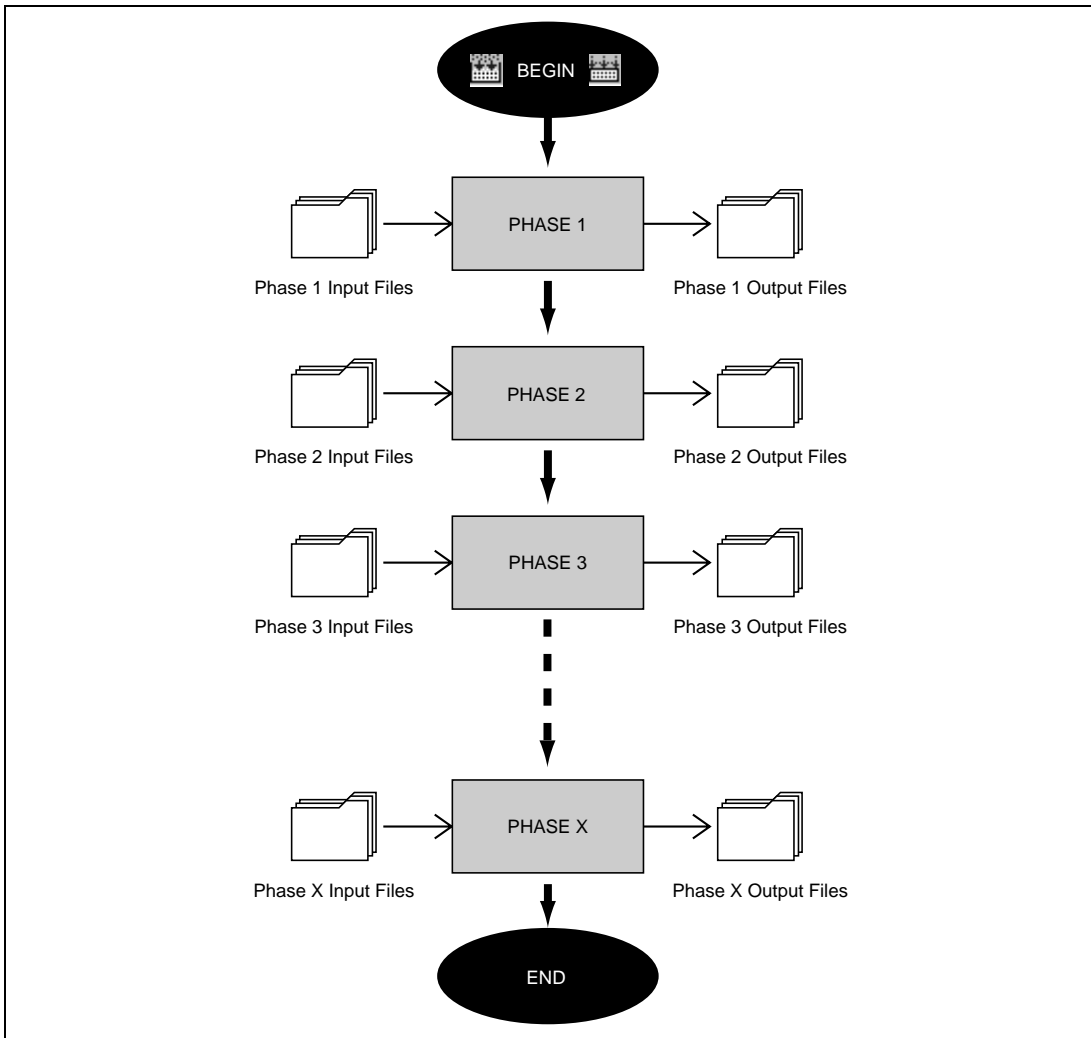


Figure 4.1 Build Process

The Hitachi 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 4.2). On the left-hand side are the phases which are defined in the current project (figure 4.2 shows a standard set of build phases). The remainder of this chapter details the various functions that the “Build Phases” dialog provides.

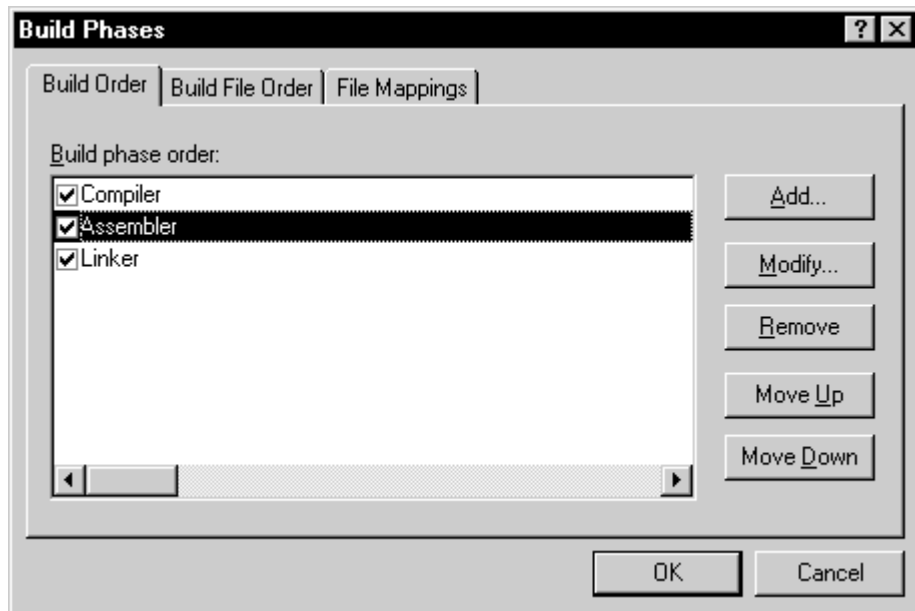


Figure 4.2 Build Phases Dialog

4.2 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 4.2) and then click the “Add...” button. This will invoke the new build phase wizard dialog (figure 4.3a).

The first step (as shown in figure 4.3a) 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.

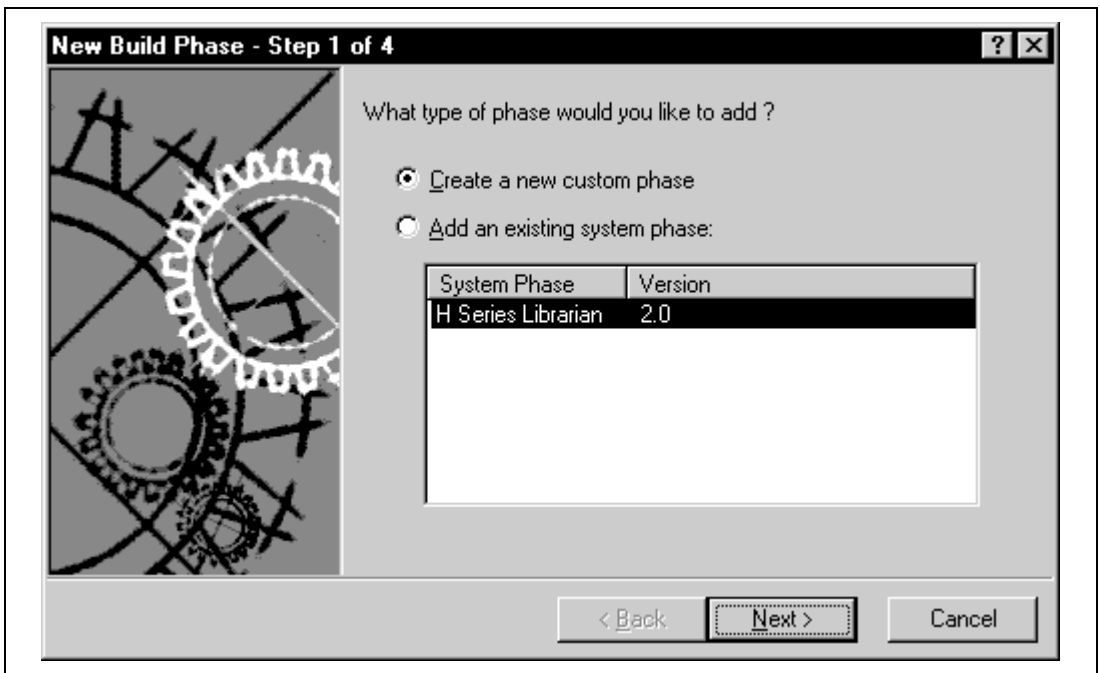


Figure 4.3a New Build Phase Dialog (Step 1)

The second step (figure 4.3b) 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.

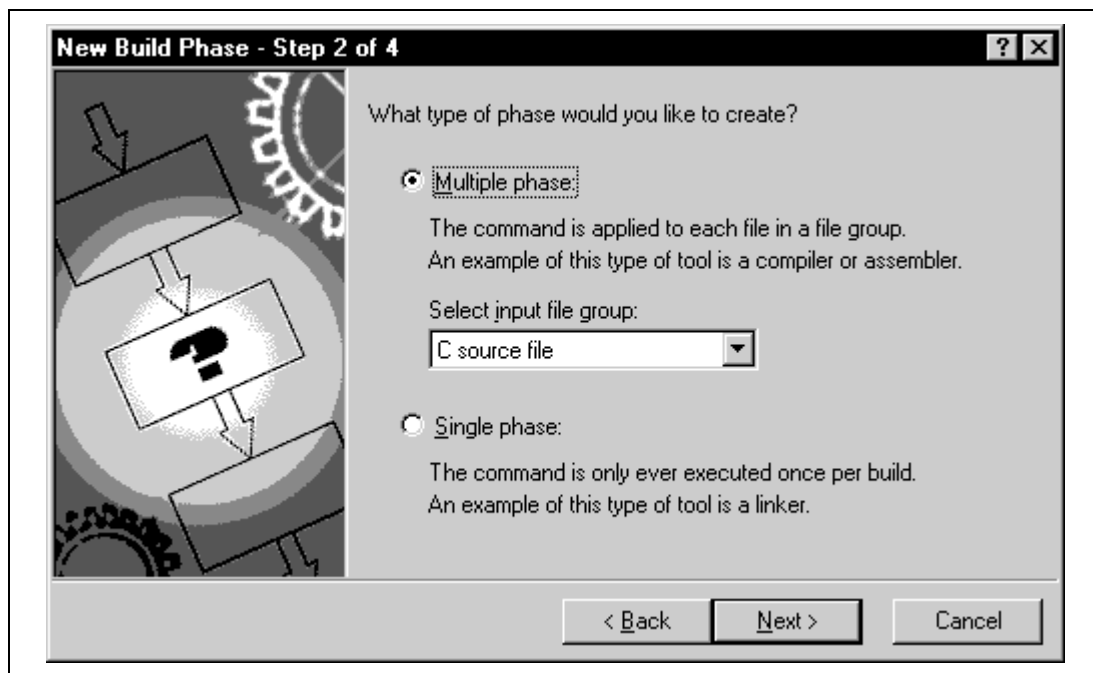


Figure 4.3b New Build Phase Dialog (Step 2)

The third step (figure 4.3c) 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.

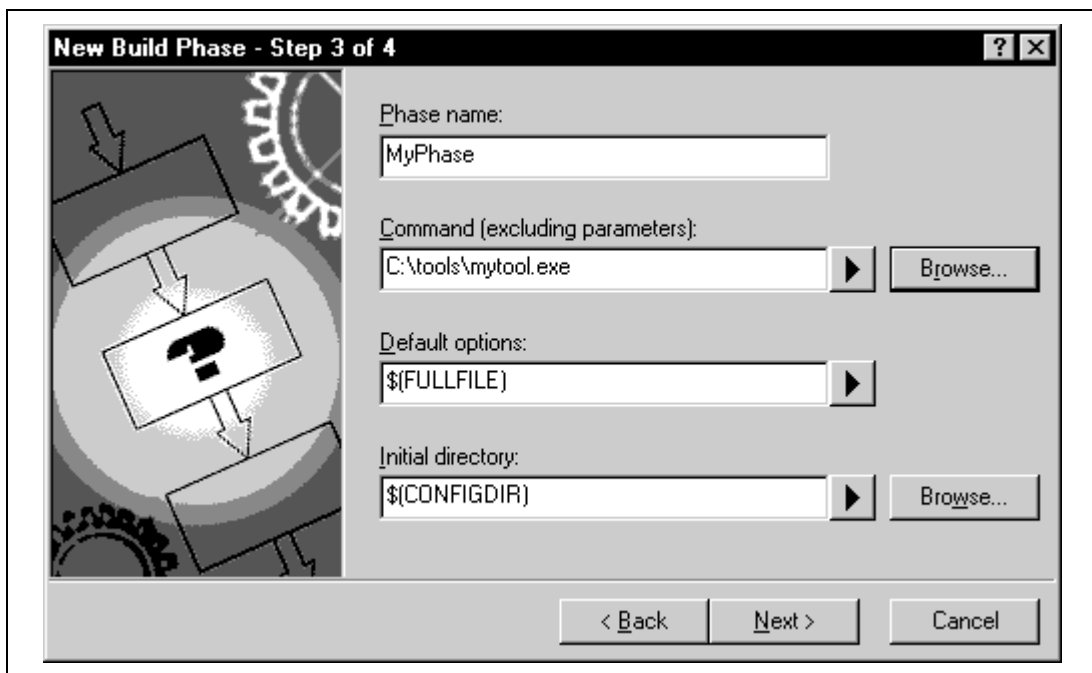


Figure 4.3c New Build Phase Dialog (Step 3)

The fourth and final step (figure 4.3d) allows you to specify any environment variables which the phase requires.

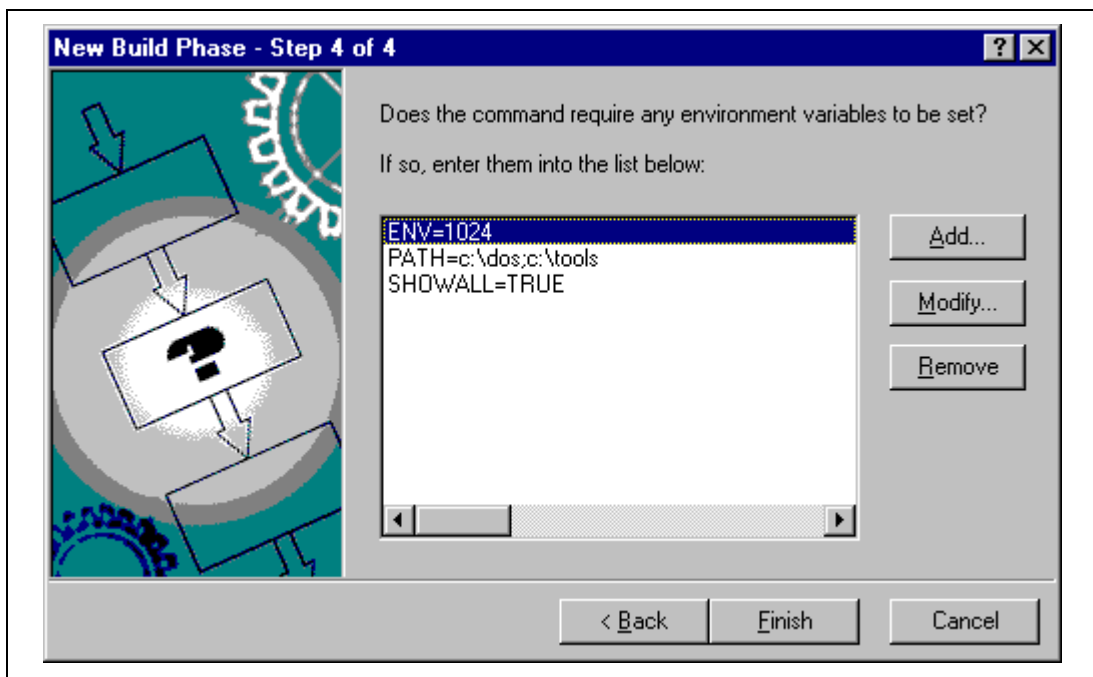


Figure 4.3d New Build Phase Dialog (Step 4)

To add a new environment variable click the “Add...” button (the dialog shown in figure 4.4 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 variable select the variable that you want to modify from this 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 an environment variable select the variable that you want to remove from the list and then click the “Remove” button.

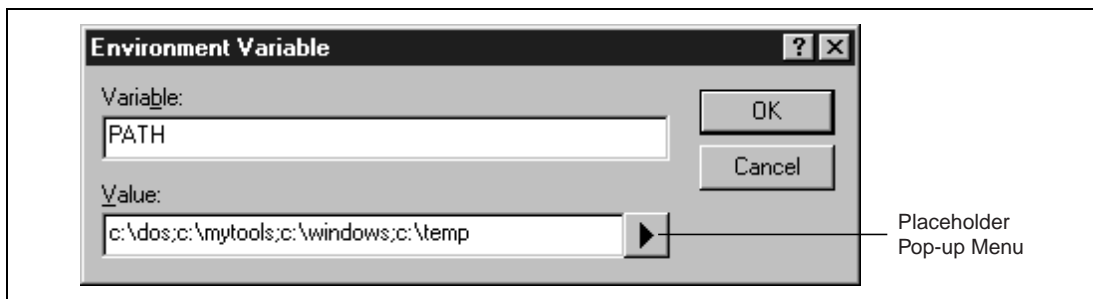


Figure 4.4 Environment Variable Dialog

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 4.2).

4.3 Ordering Build Phases

In a standard build (shown in figure 4.5), 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.

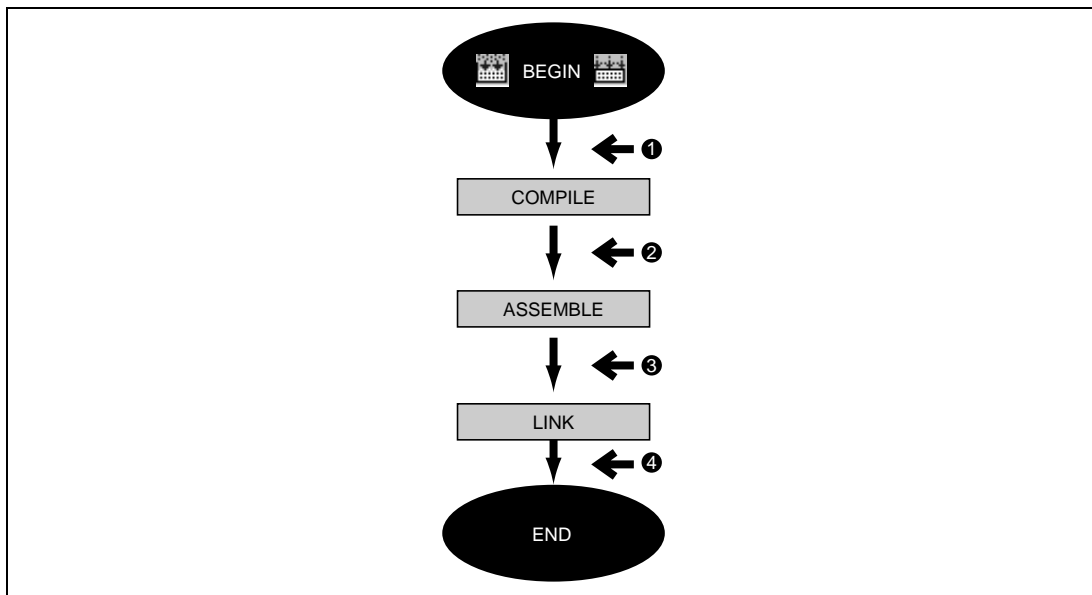


Figure 4.5 Typical Build Process

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”.

4.3.1 Build Phase Order

The “Build Order” tab (figure 4.6) 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.

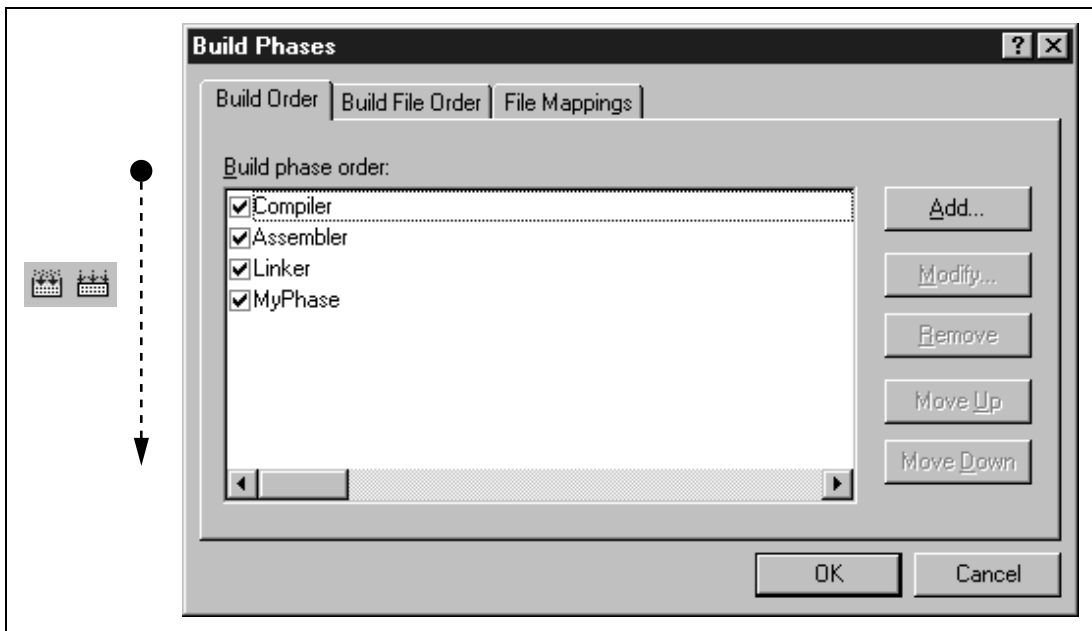


Figure 4.6 Build Phases Dialog, Build Order Tab

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 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 4.7).
 3. Change the contents of the fields as appropriate.
 4. Set the “Don’t check for input file(s) existence before executing” check box if you don’t want the HEW to abort the execution of the phase if any of the input files don’t exist.

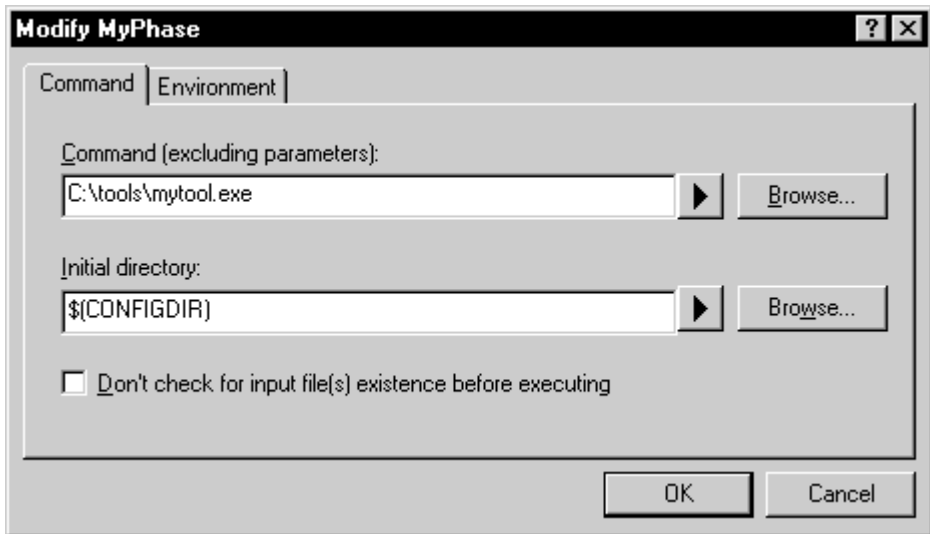


Figure 4.7 Modify Phase Dialog, Command Tab

5. Select the “Environment” tab (figure 4.8) to edit the environment settings for the phase.
6. Use the “Add...”, “Modify...” and “Remove” buttons to add, modify and remove environment variables. The operation is the same as discussed in the previous section.
7. Click “OK” when all modifications have been made.

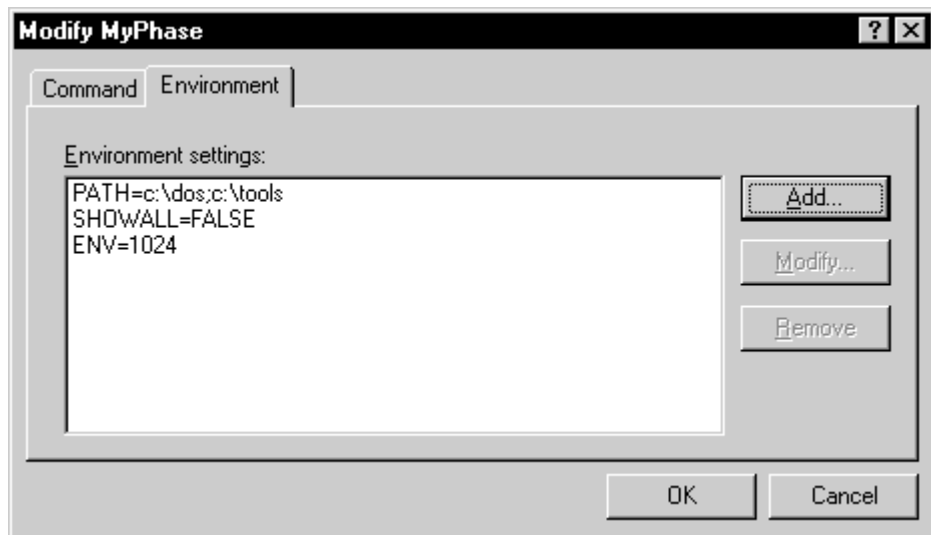



Figure 4.8 Modify Phase Dialog, Environment Tab

4.3.2 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 project 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 4.9).

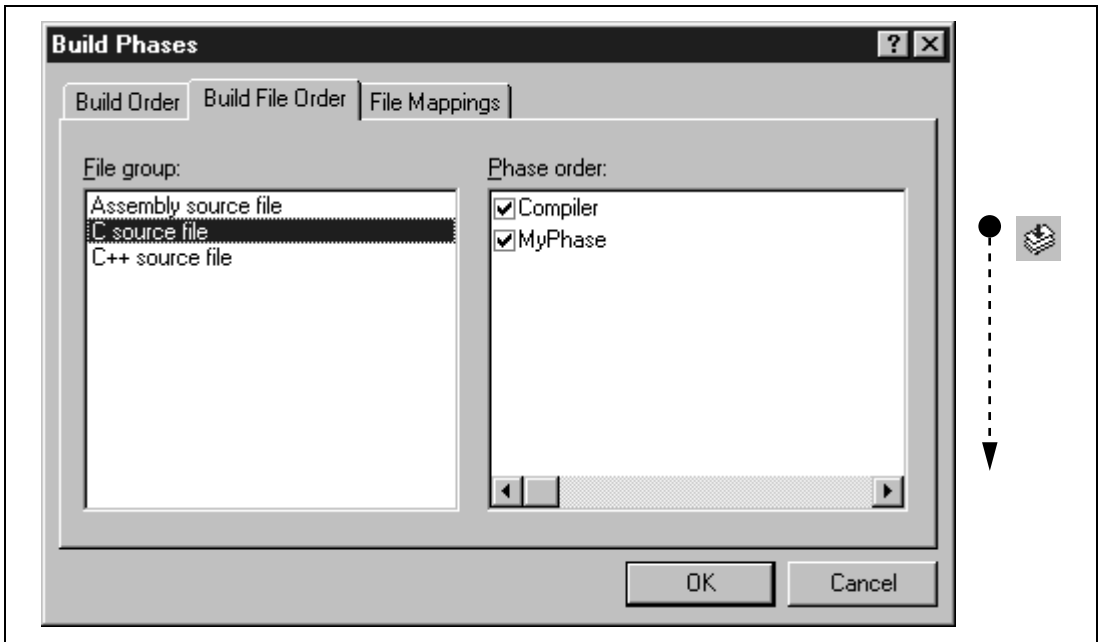


Figure 4.9 Build Phases Dialog, Build File Order Tab

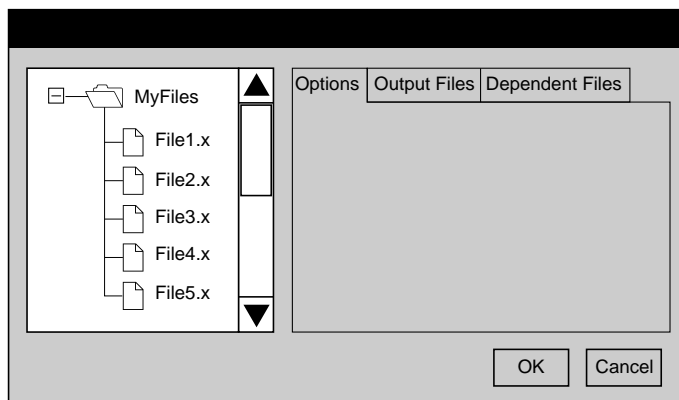
The list displays all of the current phases that will be executed when the build file operation is selected upon the file group shown in the “File group” list box. In figure 4.9 the “C source file” file group is selected and the “Compiler” and “MyPhase” phases are associated with it.

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.

4.4 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 4.3b).

i. Multiple Phase (e.g. compiler, assembler etc.)



ii. Single Phase (e.g. linker, batch file etc.)

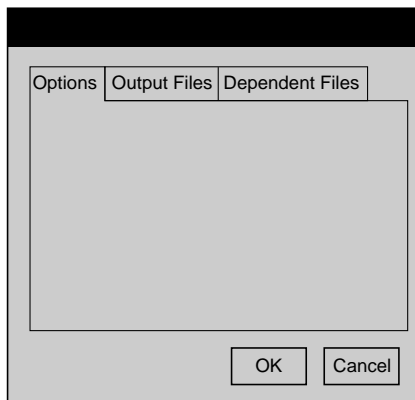


Figure 4.10 Custom Options Dialogs

If the phase is multiple then the dialog shown in figure 4.10.i is displayed. This allows you to set options for each file being input into a custom multiple phase. On the left-hand side is the file list, select the file or files that you want to set options for. On the right-hand side are the 3 options tabs, set the options that you want to apply to the selected file(s).

If the phase is single then the dialog shown in figure 4.10.ii is displayed. As the phase is not related to files in the project, it does not have the file selector on the left-hand side of the dialog. Select the options accordingly from the three tabs.

The tabs which appear on the single phase options dialog are identical to those in the multiple phase options dialog. They are detailed below.

4.4.1 Options Tab

The “Options” tab (figure 4.11) 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 4.3c). 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 B, “*Placeholders*”.

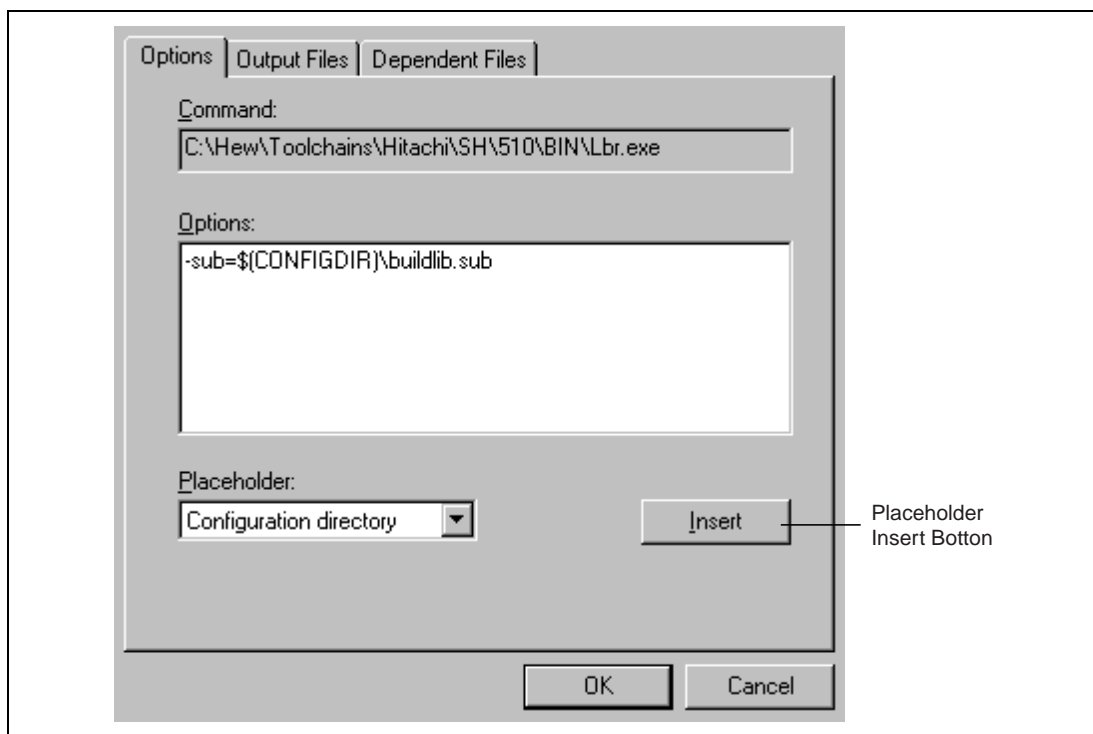


Figure 4.11 Custom Options, Options Tab

4.4.2 Output Files Tab

The “Output Files” tab (figure 4.12) 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 not, the phase is not executed for the files (i.e. the output file is up-to-date).

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

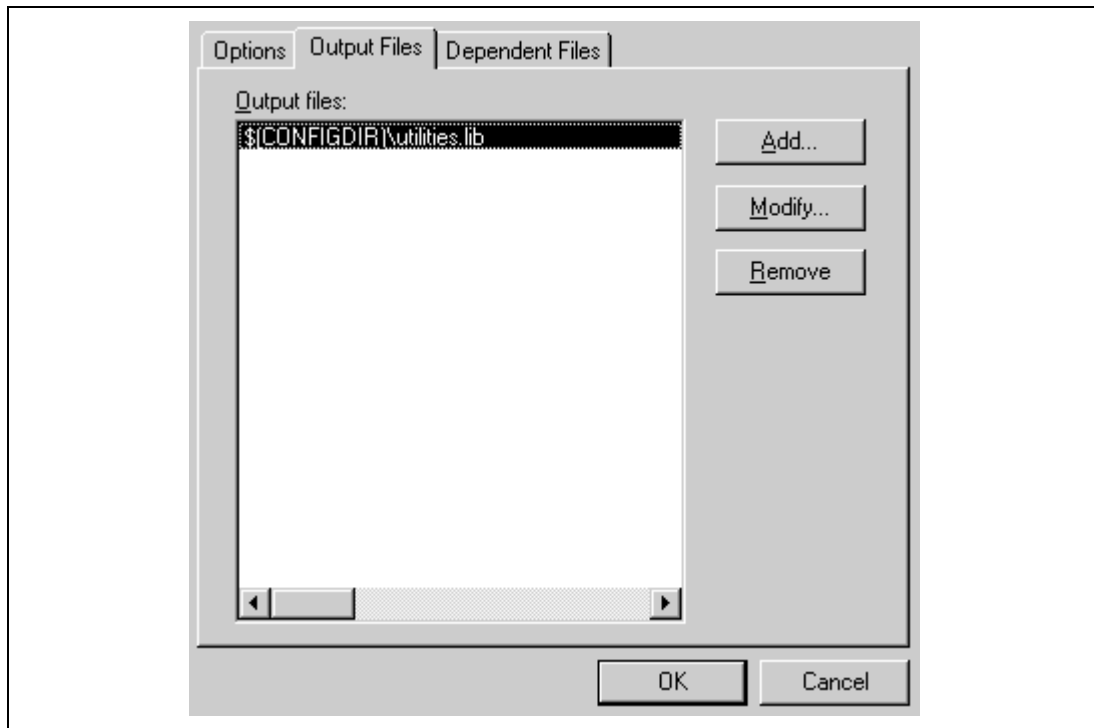


Figure 4.12 Custom Options, Output Files Tab

- To add an output file:
 1. Click “Add...”. The “Add Output File” dialog will be invoked (figure 4.13).
 2. Enter the file path or browse to it using the “Browse...” button.
 3. Click “OK” to add this output file to the list.

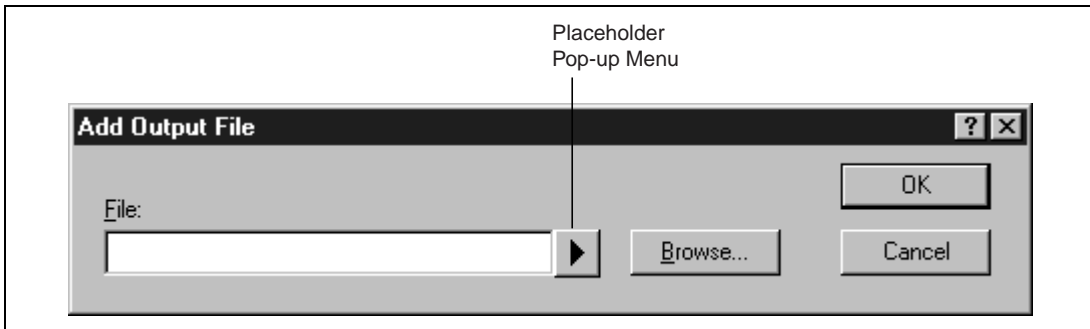


Figure 4.13 Add Output File Dialog

- 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 4.13 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.

4.4.3 Dependent Files Tab

The “Dependent Files” tab (figure 4.14) 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.

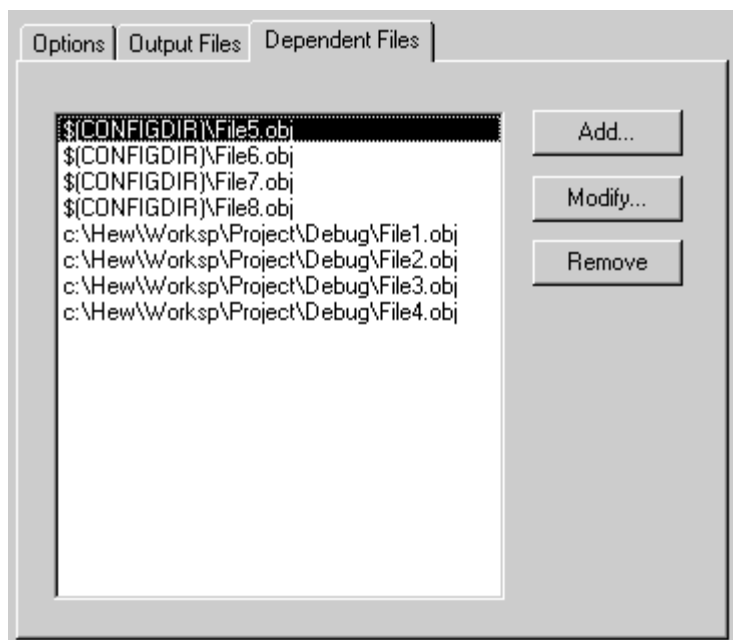


Figure 4.14 Custom Options, Dependent Files Tab

- To add a dependent file:
 1. Click “Add...”. The “Add Dependent File” dialog will be invoked (figure 4.15).
 2. Enter the file path or browse to it using the “Browse...” button.
 3. Click “OK” to add this output file to the list.

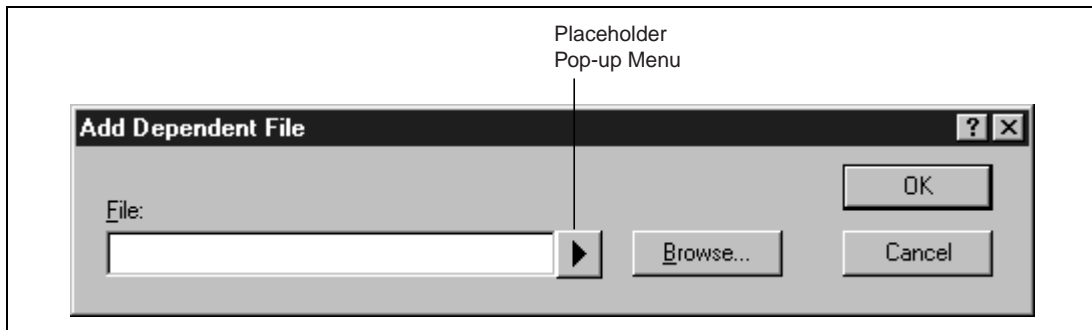


Figure 4.15 Add Dependent File Dialog

- 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 4.15 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.

4.5 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 4.3b). 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 4.16).

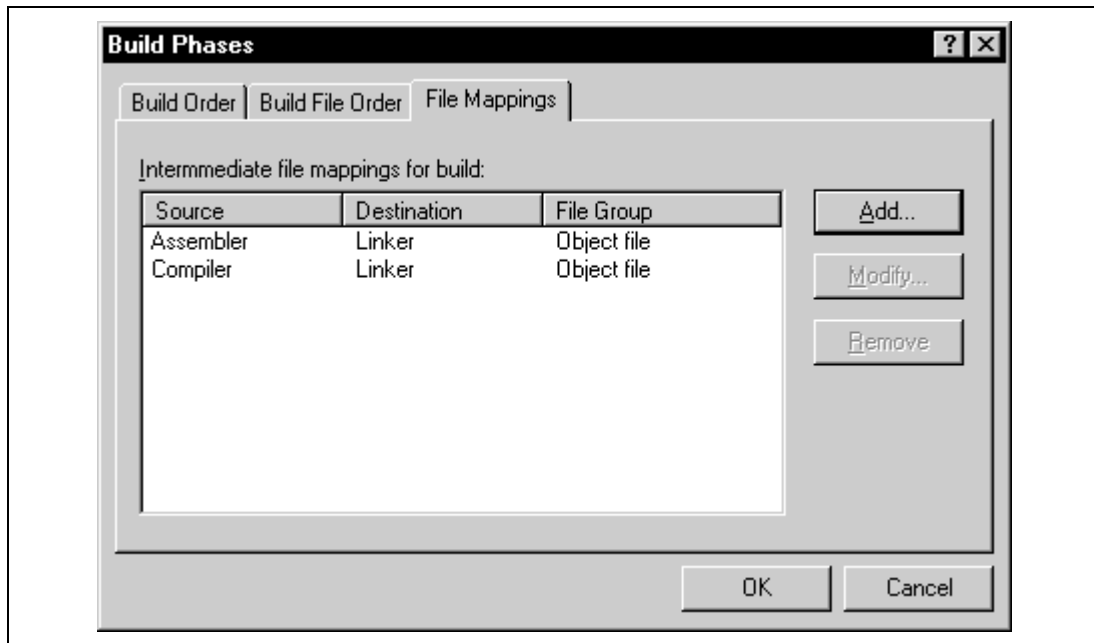


Figure 4.16 Build Phases Dialog, File Mappings Tab

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 4.17).
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.

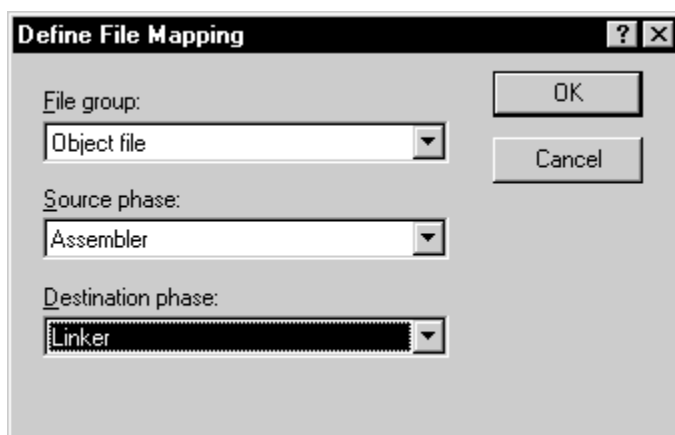


Figure 4.17 Define File Mapping Dialog

➡ 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 4.17).
3. Modify the options as necessary.
4. Click “OK” to commit the changes.

4.6 Controlling the Build

By default, the Hitachi Embedded Workshop will execute all of the phases in a build and only stop if a fatal error is encountered. You can change this behaviour by setting the controls on the “Build” tab of the “Tools Options” dialog (figure 4.18).

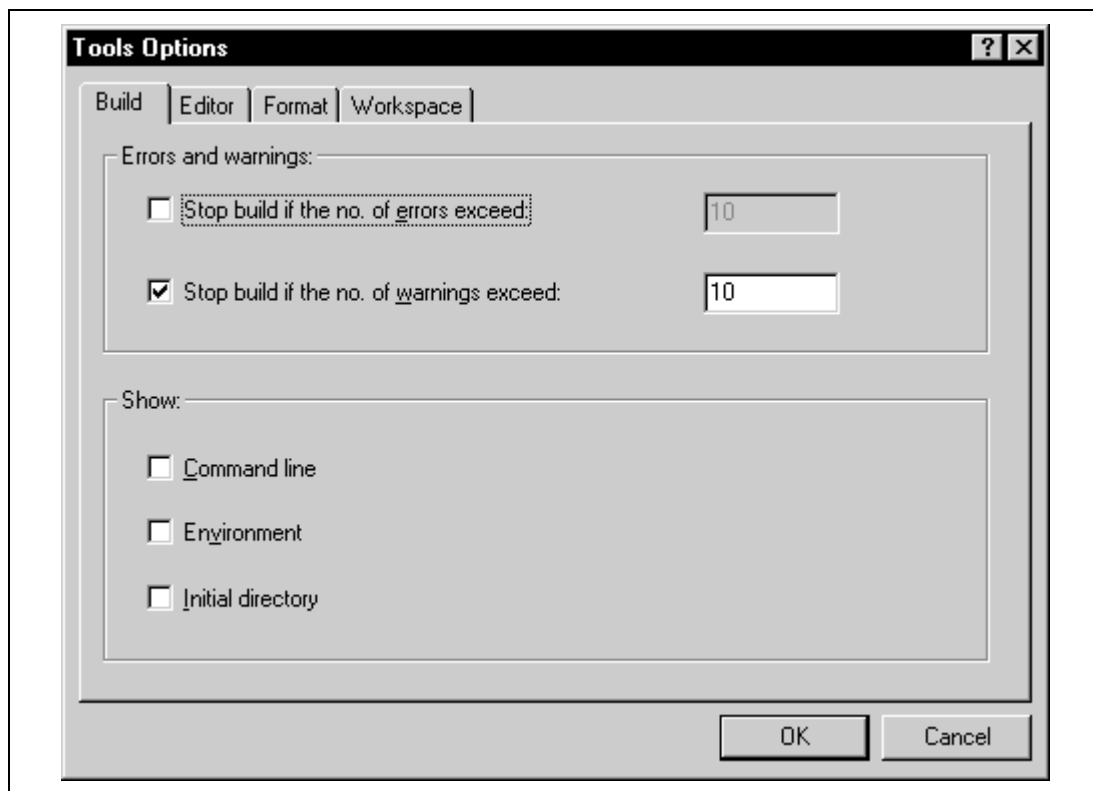


Figure 4.18 Tools Options Dialog, Build Tab

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.

4.7 Logging Build Output

If you would like to write the results of each build to file then invoke the “Tools Customize” dialog by selecting [**Tools->Customize...**] and select the “Log” tab (figure 4.19). 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.

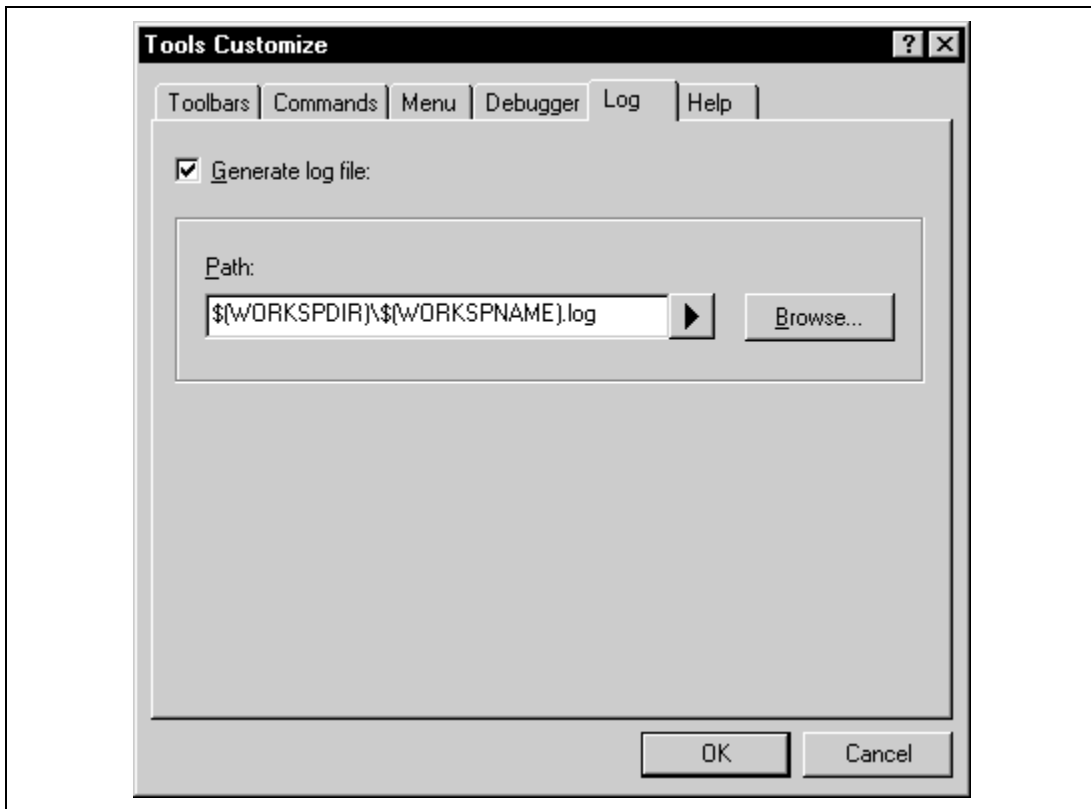


Figure 4.19 Tools Customize Dialog, Log Tab

4.8 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 “Toolchain Upgrade” dialog shown in figure 4.20. To invoke the dialog, select [**Tools->Upgrade...**]. Choose one of the versions from the “Toolchain version” drop-down list and click the “OK” button to enforce your choice.

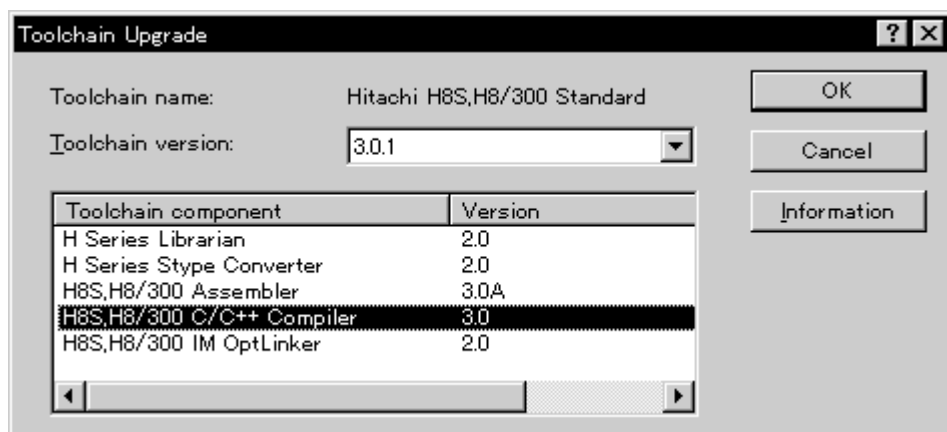


Figure 4.20 Toolchain Upgrade Dialog

To show information of a toolchain component, select a tool from the “Toolchain component” list on the “Toolchain Upgrade” dialog and click the “Information” button. Then a tool information dialog (figure 4.21) will show you the information of the tool. Click the “Close” button to close the dialog.

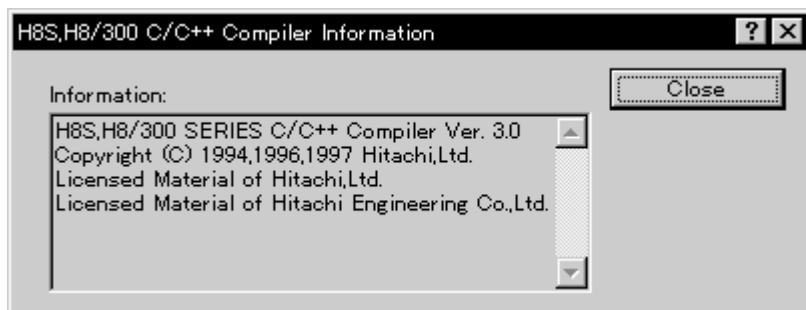


Figure 4.21 Toolchain Information Dialog

4.9 Using the Hitachi Debugging Interface (HDI)

The Hitachi Debugging Interface is designed to work with the Hitachi Debugging Interface (HDI) version 4.0 or greater. If you want to use another debugger then you must add it to the tools menu (as described in chapter 6, “*Customizing the Environment*”).

The “Debugger” tab of the “Tools Customize” dialog (figure 4.22) is where the HDI related information is configured. Invoke it by selecting [Tools->Customize...] and then selecting the “Debugger” tab.

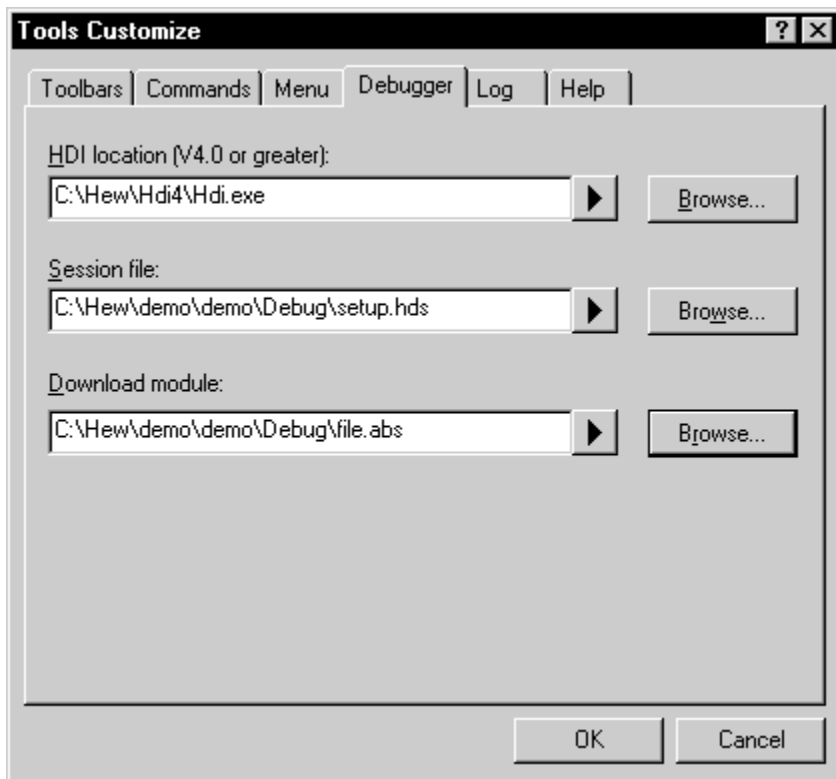


Figure 4.22 Tools Customize Dialog, Debugger Tab

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 behaviour 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 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.

4.10 Generating a Makefile

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

- ☛ To generate a makefile of the current project:
 1. Ensure that the project which you want to generate a makefile for is the current project.
 2. Ensure that the build configuration which you want to build the project with is the current configuration.
 3. Select **[Build->Generate Makefile]**.

The HEW will create a subdirectory “make” within the current project’s directory and then generate the makefile into it. It is named after the current configuration, with a .mak extension (e.g. 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 of 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.

Section 5 Tool Administration

You control the components which can be used by the Hitachi Embedded Workshop via the “Tools Administration” dialog (figure 5.1), which is invoked via **[Tools->Administration...]**. The “Tools Administration” dialog is only accessible when no workspace is open.

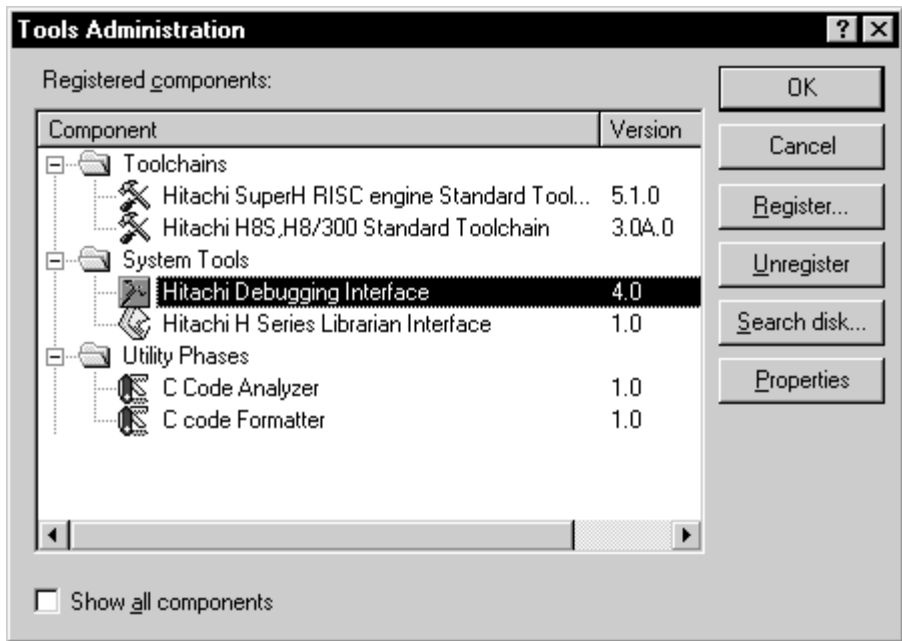


Figure 5.1 Tools Administration Dialog (Example)

There are three standard types of component:

- **Toolchain** - a set of build phases (e.g. compiler, assembler, linker and librarian). 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. 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.

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).

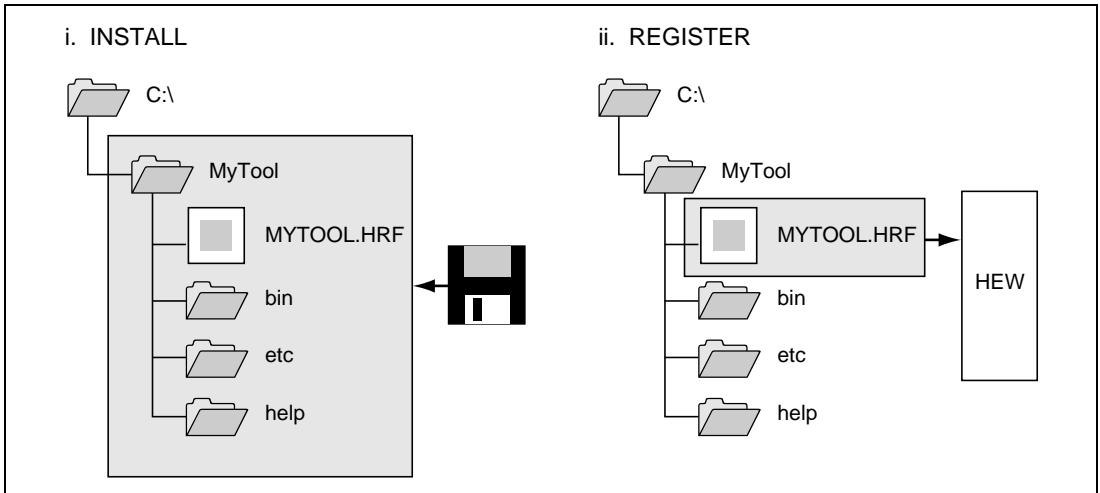


Figure 5.2 HRF File Location and Registration

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...].

Tool information is stored locally on your machine. The Hitachi Embedded Workshop does this so that workspaces and projects can be tool installation independent. In other words, a project can refer to a particular toolchain without including *where* it is stored – this information is extracted locally from the information about registered tools.

Figure 5.3 illustrates this principle graphically. It shows three machines that reference a shared project, which use a Hitachi compiler, that is stored in different locations on each machine.

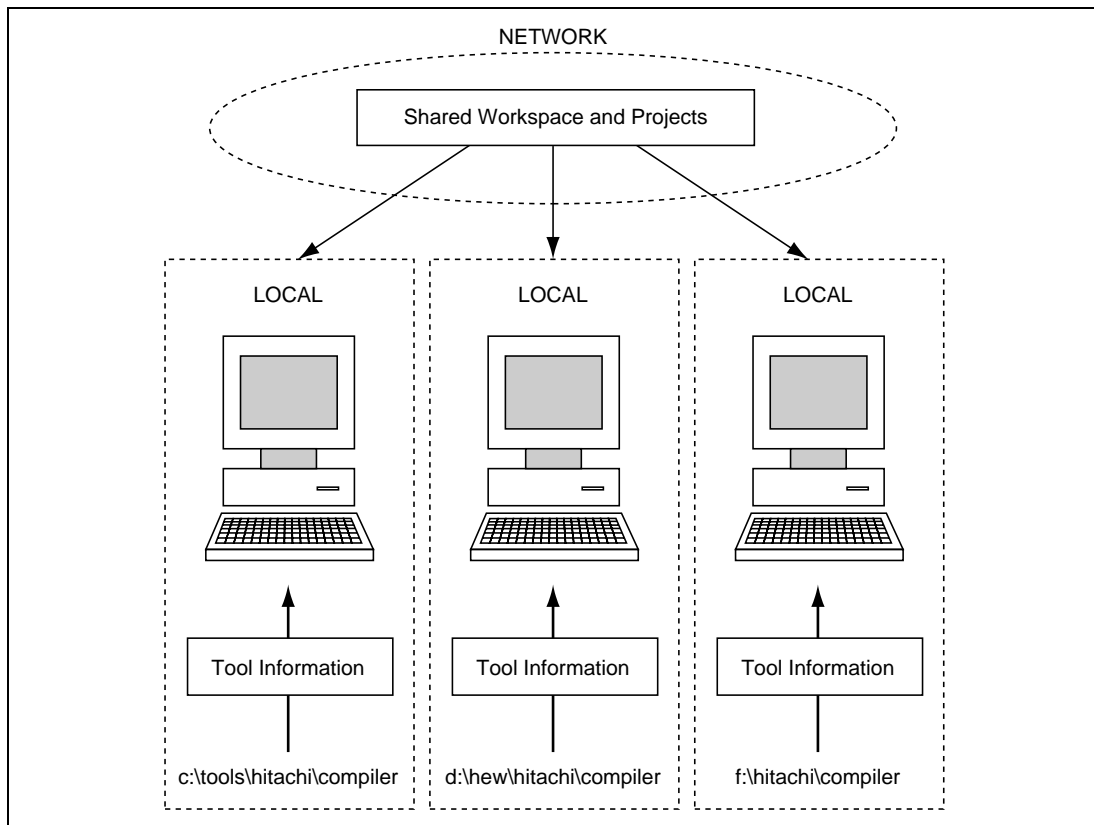


Figure 5.3 Local Storage of Tool Information

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 5.4).

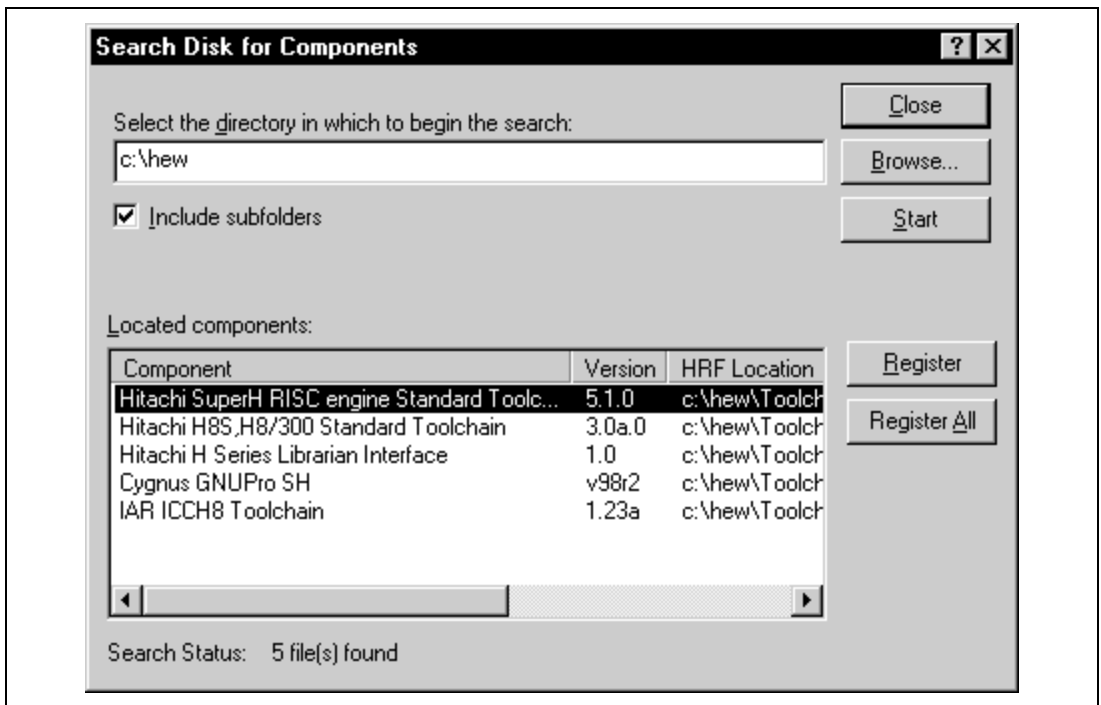


Figure 5.4 Search Disk for Components Dialog

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 (figure 5.1). 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 manually (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 5.5). This tab displays the name, version and location of the selected component. None of the information on this tab is editable.

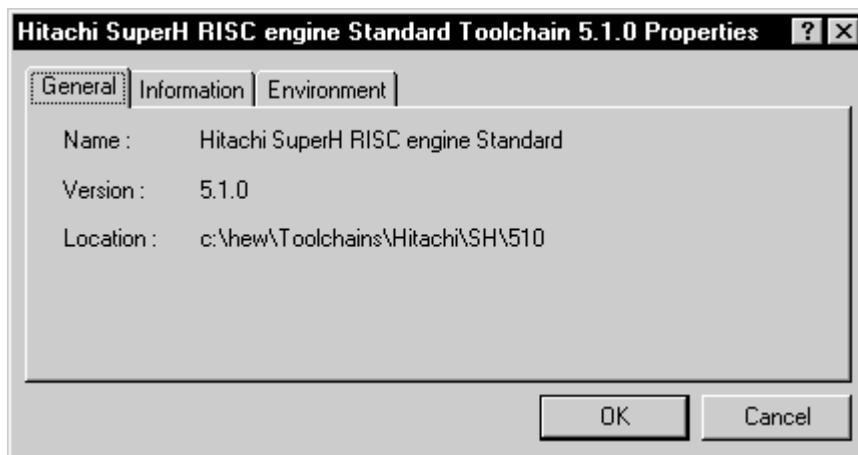


Figure 5.5 Properties Dialog, General Tab

Select the “Information” tab to view any information about the component (figure 5.6). This may include copyright information, enhancements, bug fixes, user notes and so on.

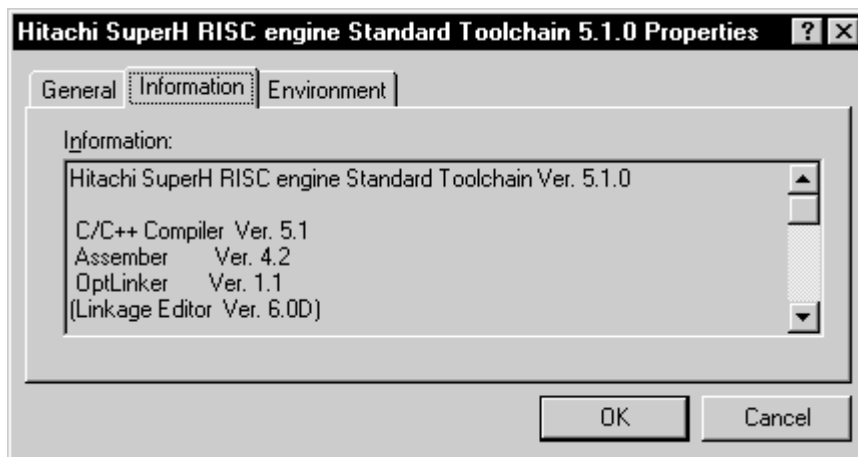


Figure 5.6 Properties Dialog, Information Tab

Select the “Environment” tab, if it exists, to view and edit a component’s environment settings (figure 5.7). This dialog is most commonly used to modify the environment of a toolchain.

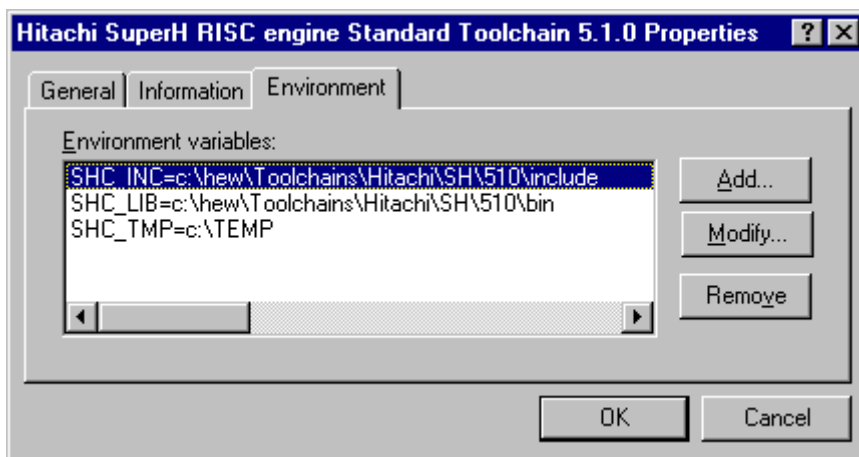


Figure 5.7 Properties Dialog, Environment Tab

To add a new environment variable, click the “Add...” button (the dialog shown in figure 5.8 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 B, “*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.

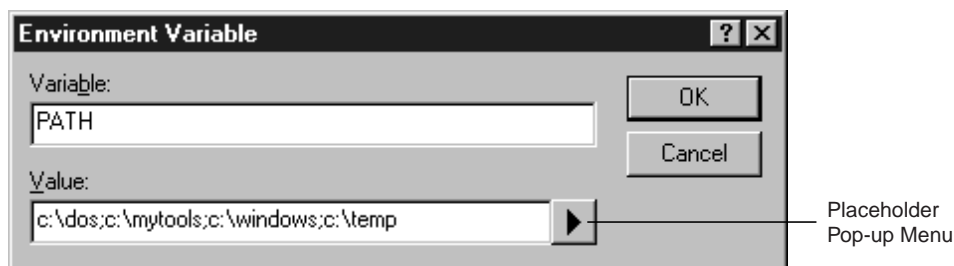


Figure 5.8 Environment Variable Dialog

5.6 Uninstalling Components

The HEW provides a separate application which removes component installations. To run the tool uninstaller, open the “Start” menu of Windows®, select “Programs”, select “Hitachi Embedded Workshop” and then select the shortcut of TUninst.EXE. Then the dialog in figure 5.9 will be invoked.

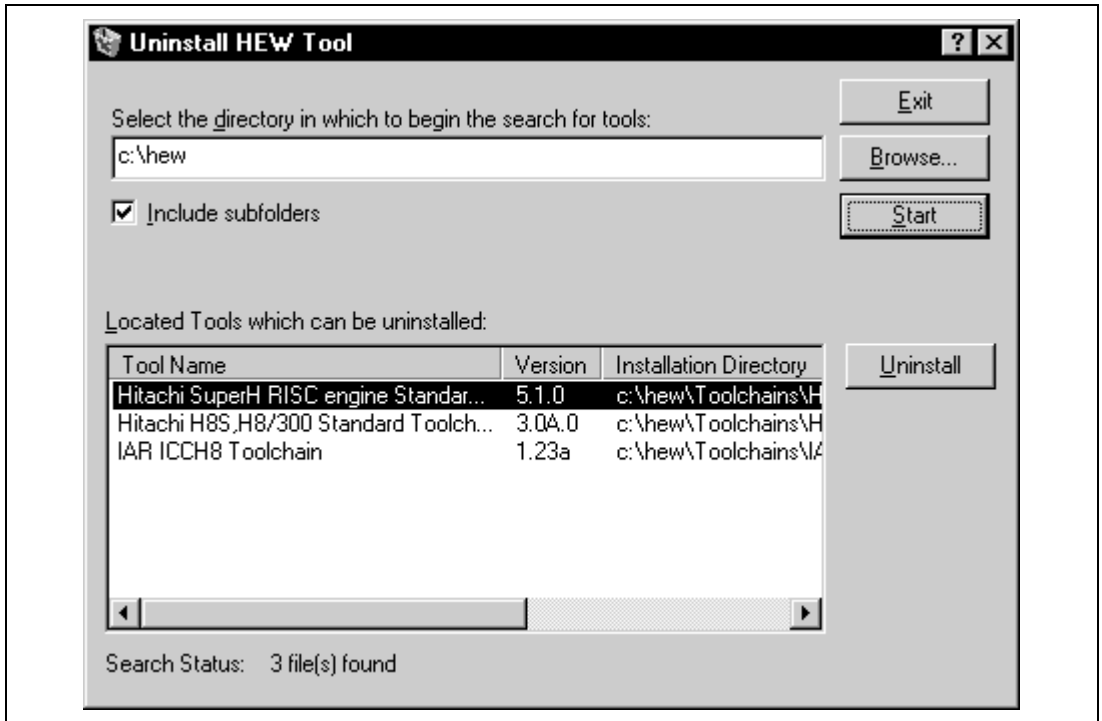


Figure 5.9 Uninstall HEW Tool

➡ To uninstall a component:

1. Enter the directory in which you would like to search into the top field or browse to it graphically by clicking the “Browse...” button.
2. Check the “Include subfolders” check box if you would like to search the directory specified and all directories below it.
3. 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.
4. 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.
5. 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 5.10 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.

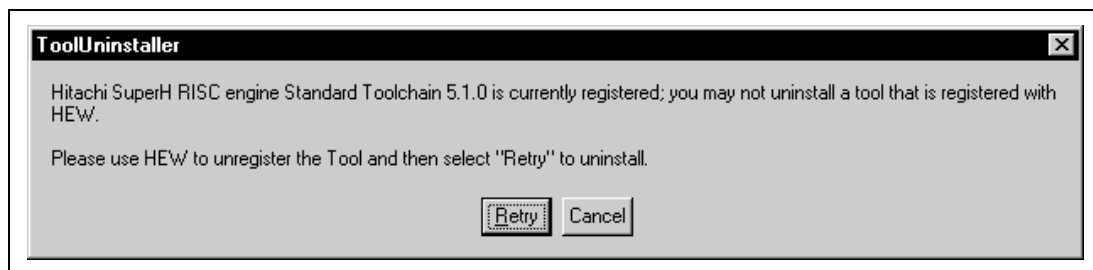


Figure 5.10 Unable to Uninstall Tool

If a tool is not registered with the HEW then the dialog shown in figure 5.11 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.

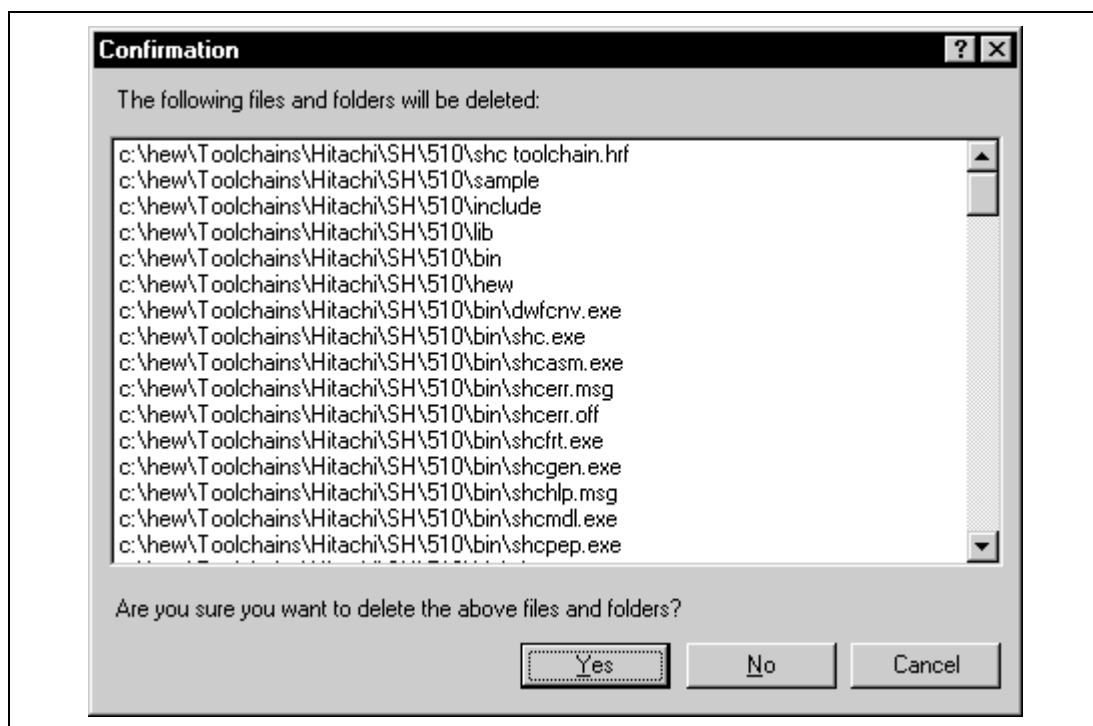


Figure 5.11 Confirmation Dialog

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 5.12).

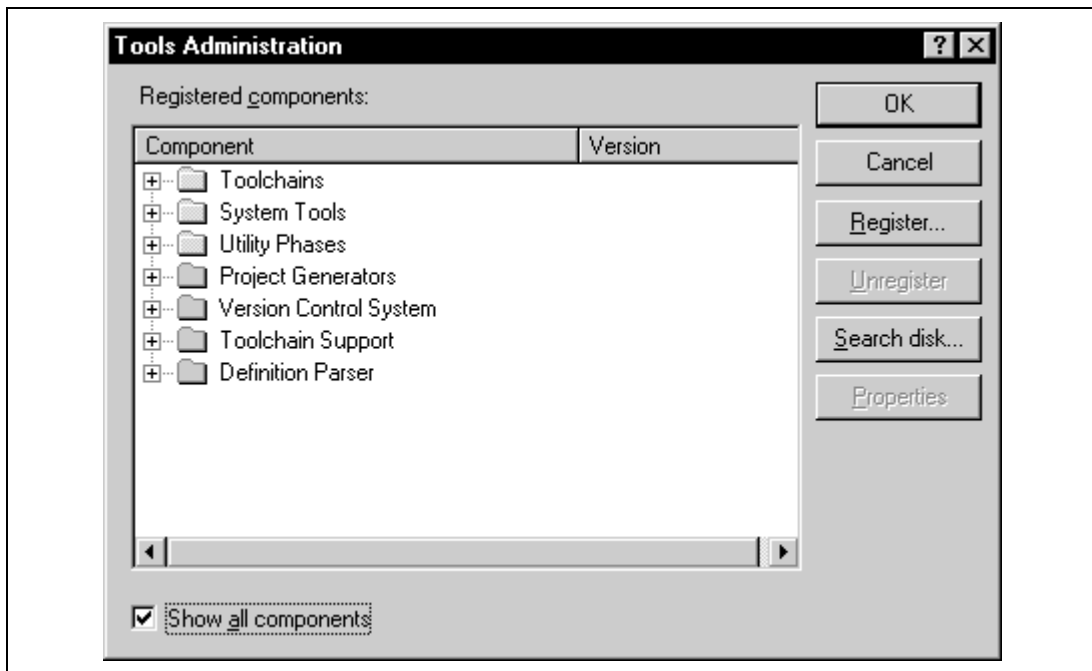


Figure 5.12 All Components Shown

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.

Section 6 Customizing the Environment

6.1 Customizing the Toolbar

The Hitachi 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 “Tools Customize” dialog (figure 6.1).

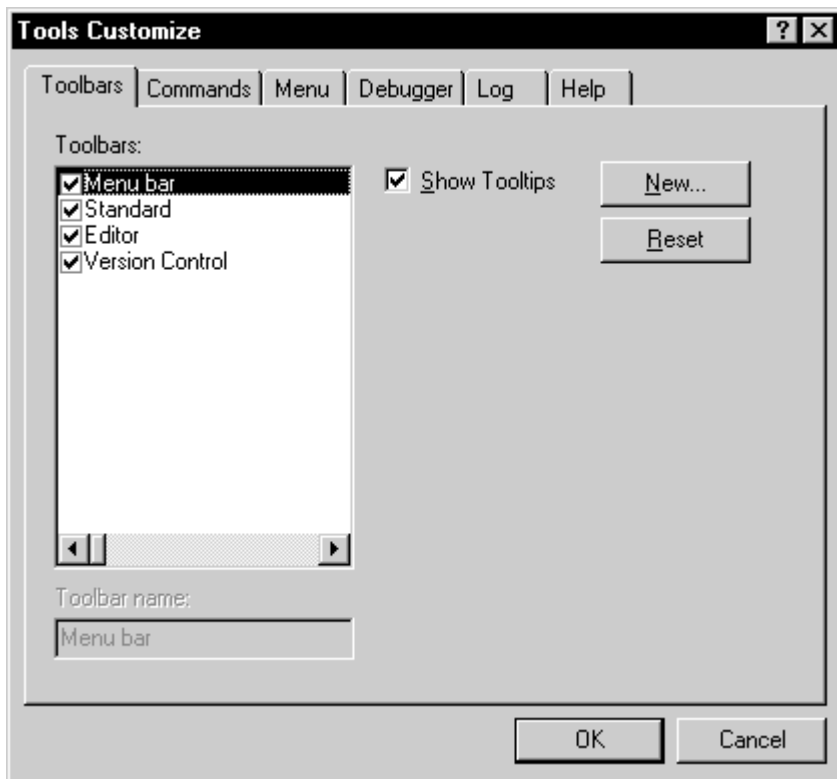


Figure 6.1 Tools Customize Dialog, Toolbars Tab

- 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.

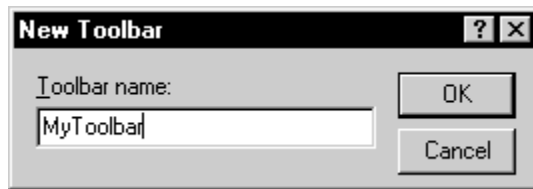


Figure 6.2 New Toolbar Dialog

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.

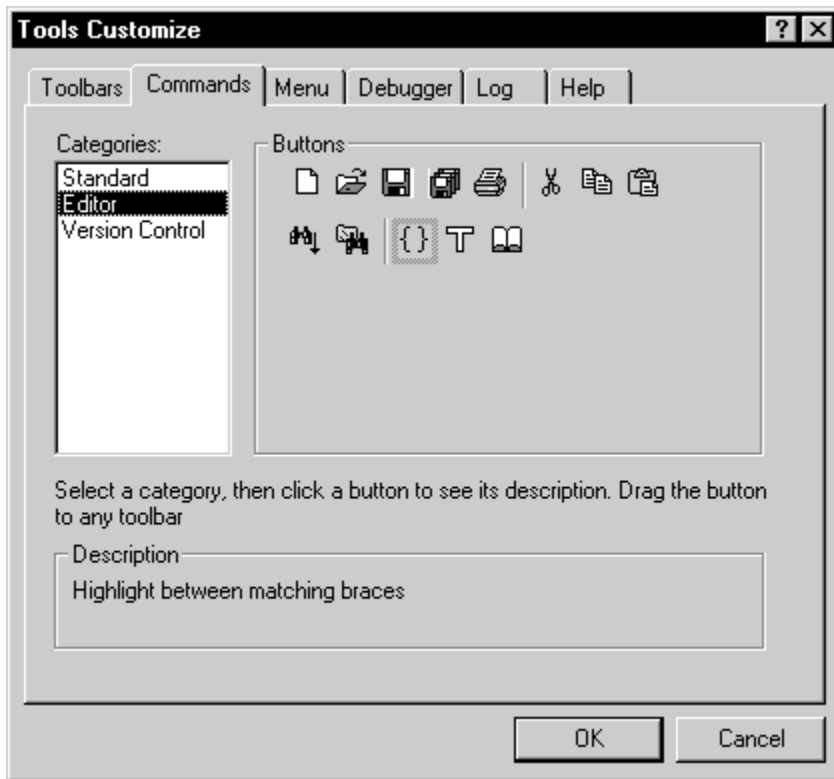


Figure 6.3 Tools Customize Dialog, Commands Tab

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

6.2 Customizing the Tools Menu

The [**Tools**] menu can be customized to include your own menu options.

- To add a new menu option:
 1. Select [**Tools->Customize...**]. The dialog shown in figure 6.1 will be displayed. Select the “Menu” tab (see figure 6.4).

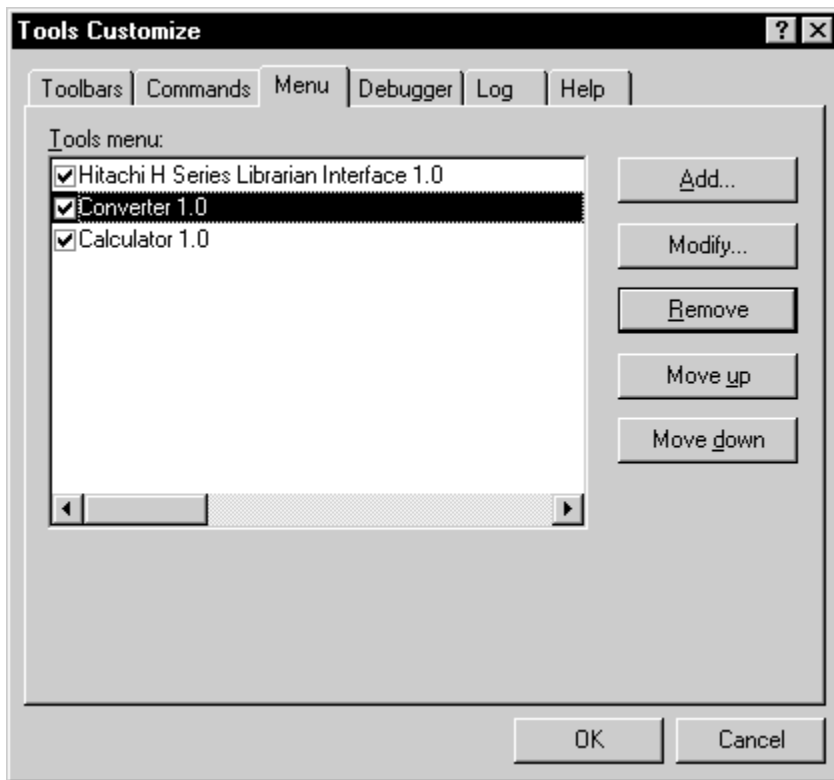


Figure 6.4 Tools Customize Dialog, Menu Tab

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 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.

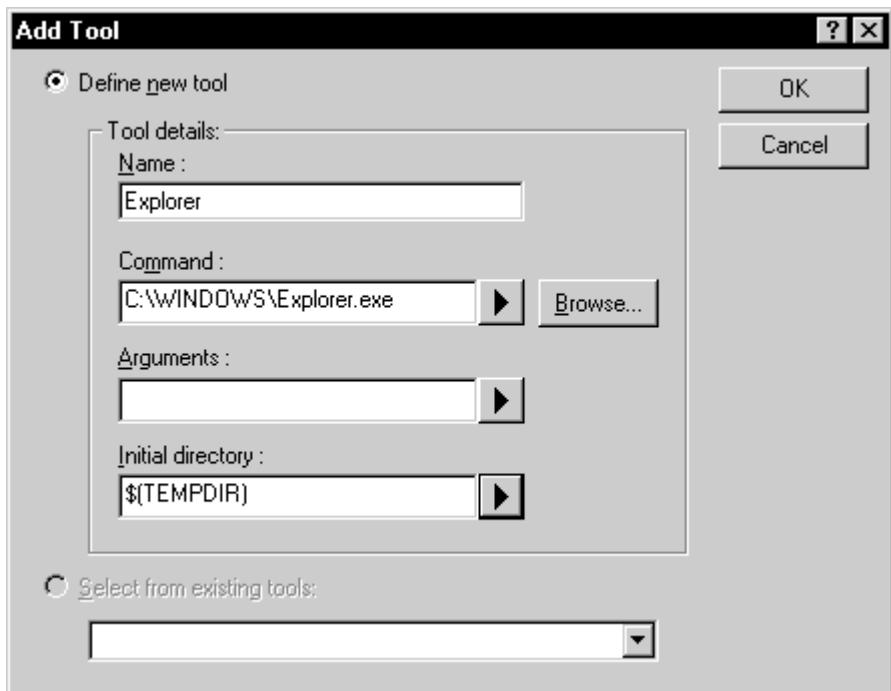


Figure 6.5 Add Tool Dialog

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 move a menu option up or down:
 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 move and then click the “Move up” and “Move down” buttons to move the menu option up or down.
 3. Click “OK” for the new position to take effect.
- 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”.

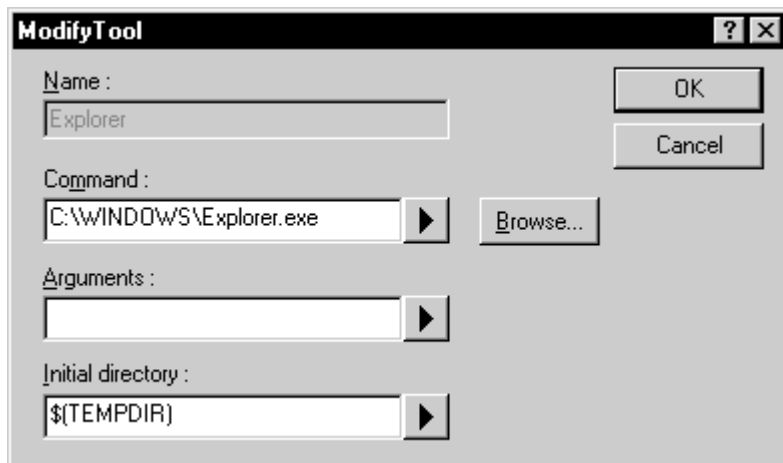


Figure 6.6 Modify Tool Dialog

➡ 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 Hitachi 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 Hitachi 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 “Tools 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).

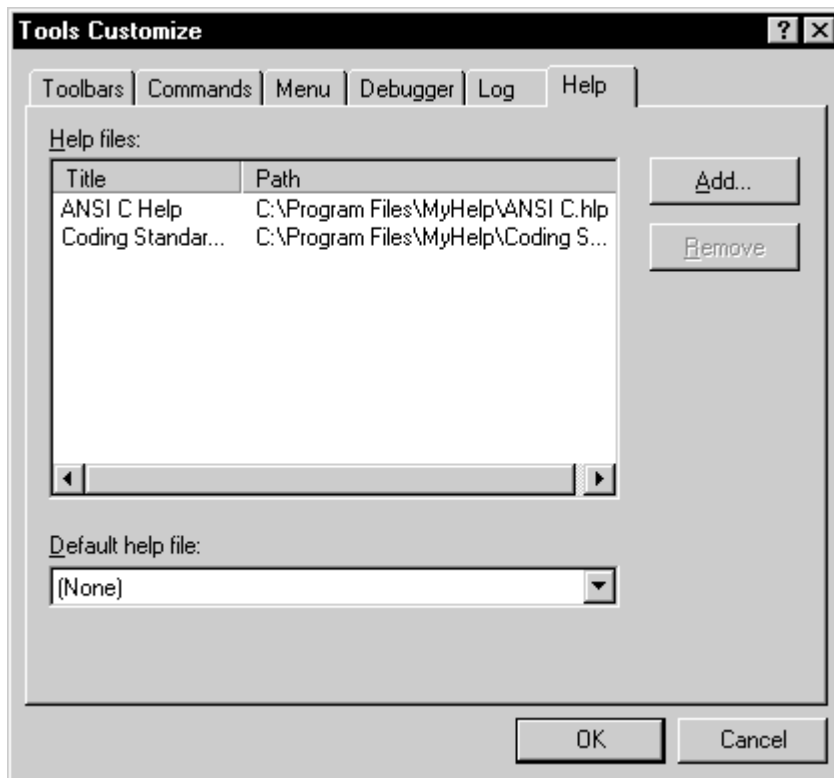


Figure 6.7 Tools Customize Dialog, Help Tab

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.

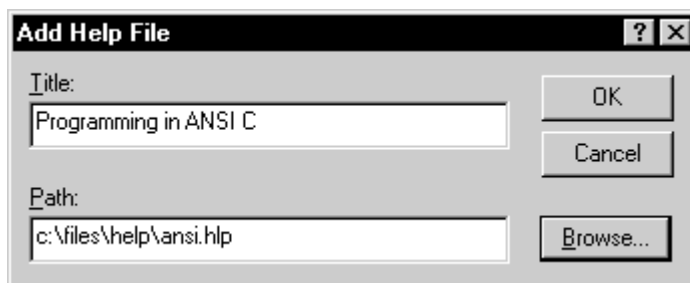


Figure 6.8 Add Help File Dialog

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

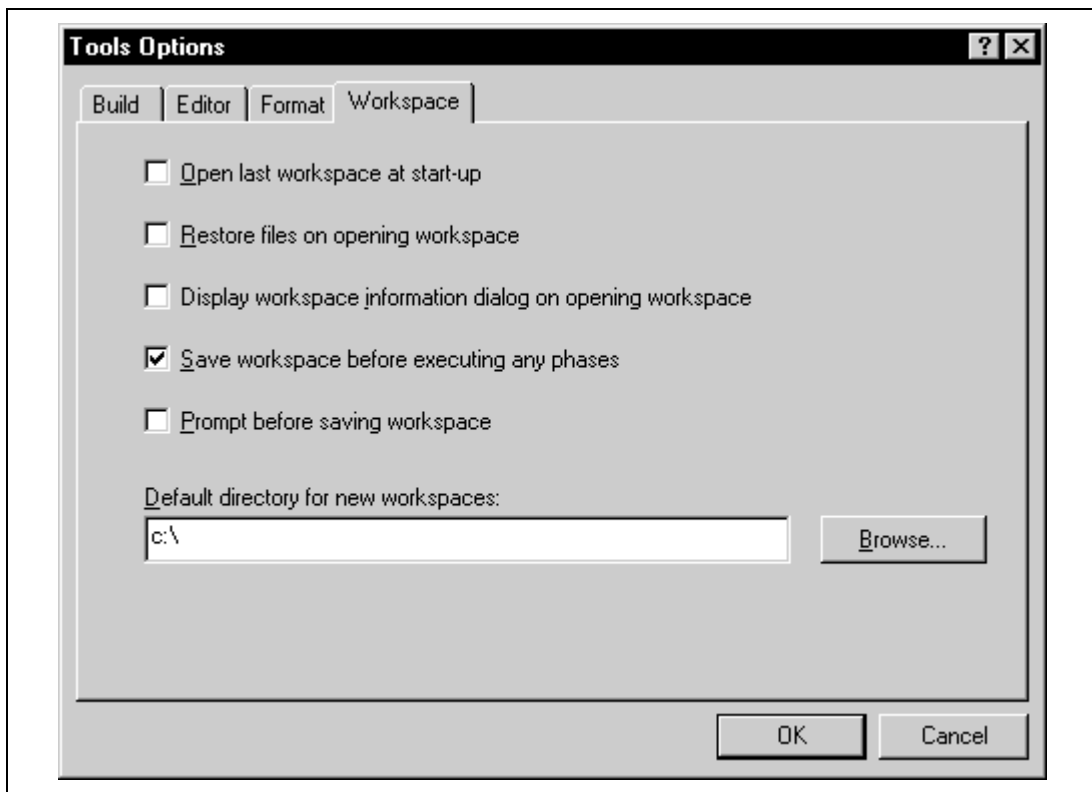


Figure 6.9 Tools Options Dialog, Workspace Tab

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 Hitachi 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 Hitachi 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 “Tools Options” dialog.
5. Click “OK” to save the description on the “Information” dialog. Click the “Cancel” button not to save the description.

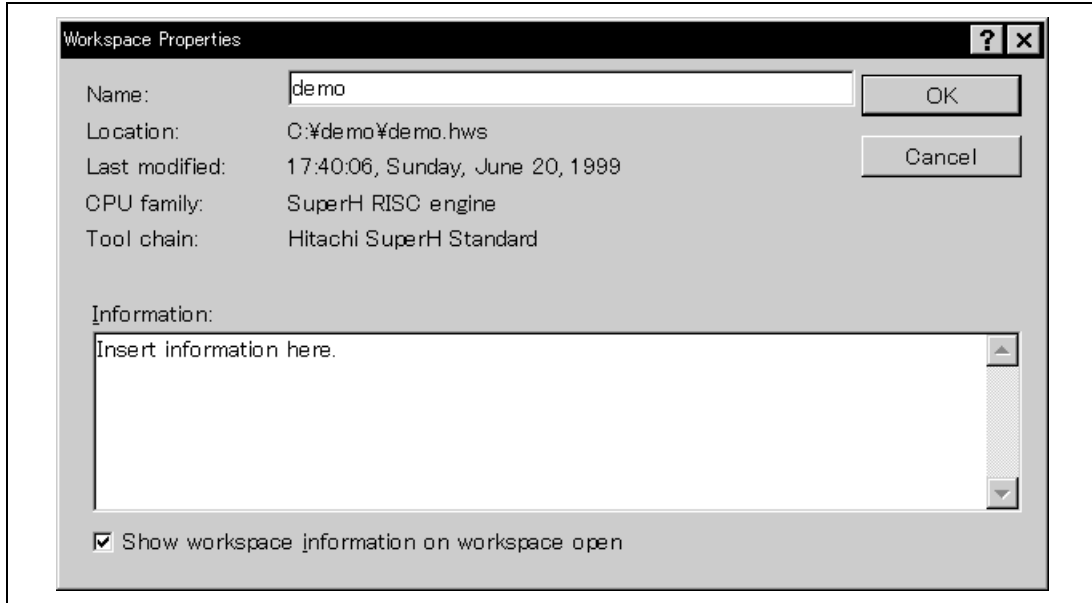


Figure 6.10 Workspace Properties Dialog

When a workspace is opened, the Hitachi 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 phases

To force the Hitachi 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 Hitachi Embedded Workshop invokes the “New Project 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.

6.5 Using an External Editor

The Hitachi 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 “Tools Options” dialog will be displayed. Select the “Editor” tab (figure 6.11).

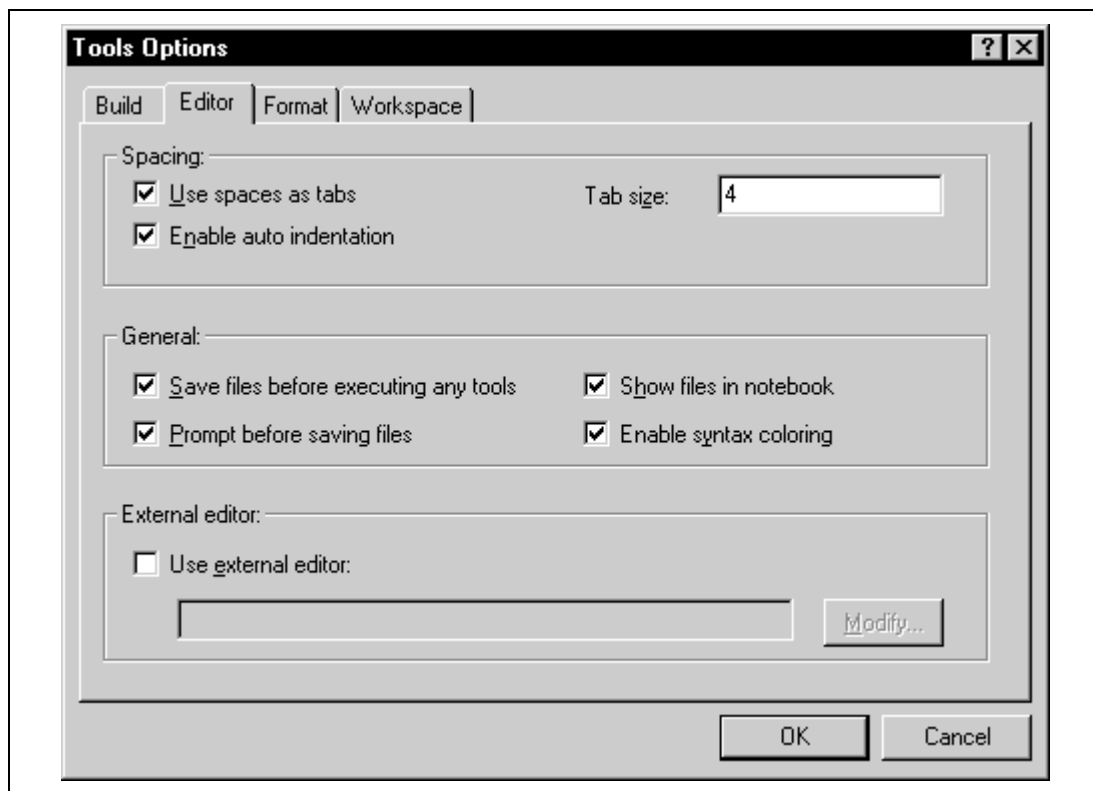


Figure 6.11 Tools Options Dialog (Editor Tab)

2. Check the “Use external editor” check box.
3. Click the “Modify...” button. The “External Editor” dialog will be displayed (figure 6.12).

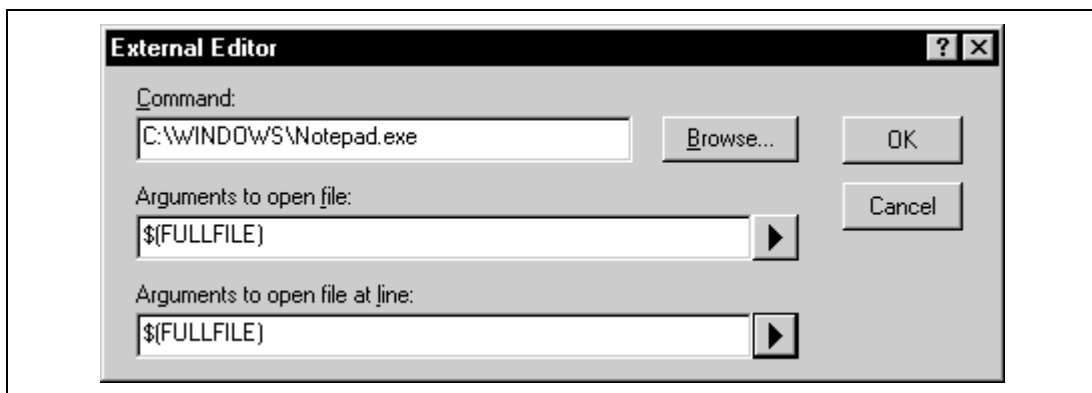


Figure 6.12 External Editor Dialog

4. Enter the path of the executable (without any arguments) into the “Command” field.
5. 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.
6. 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.
7. 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 Hitachi Embedded Workshop allows you to customize file save on the “Editor” tab of the “Tools 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 Hitachi 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.

Section 7 Version Control

The Hitachi Embedded Workshop provides facilities for connecting to a version control tool. Some of the reasons why version control is 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.

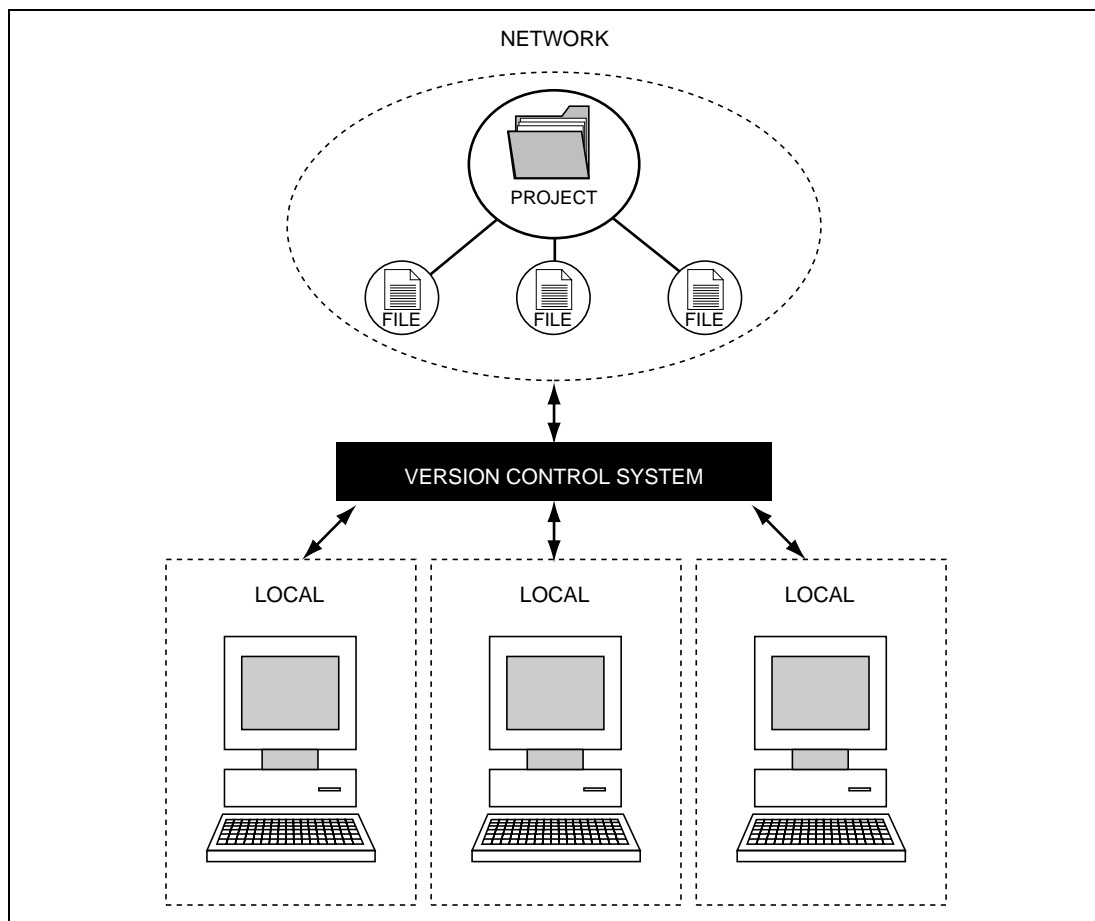


Figure 7.1 Version Control

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.

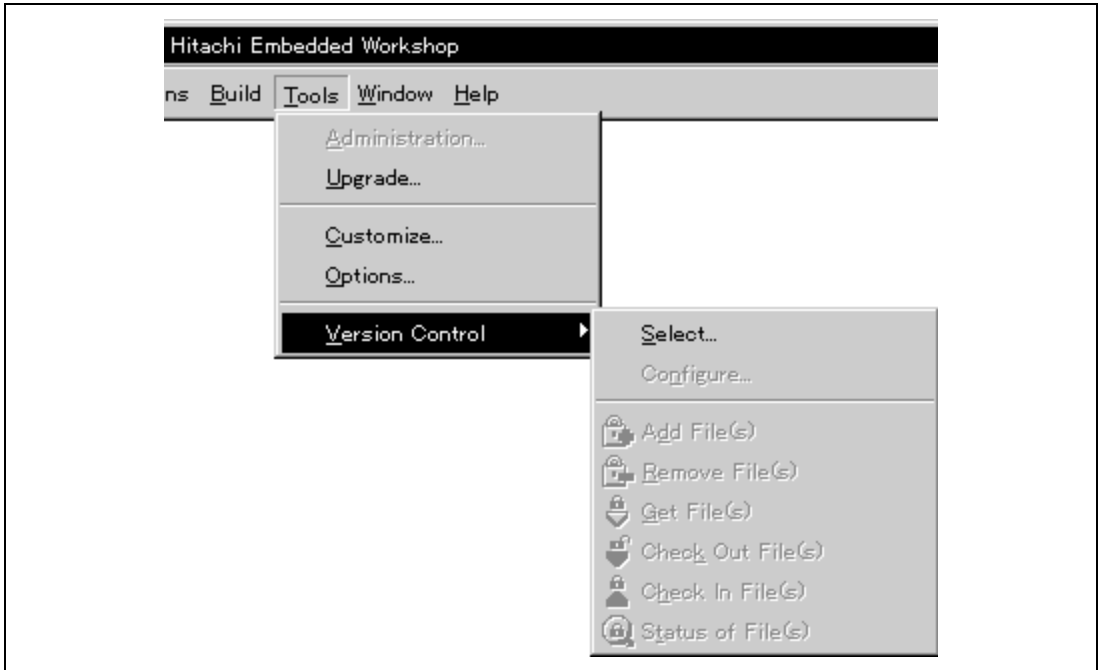


Figure 7.2 Version Control Sub-Menu

- ➡ 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.

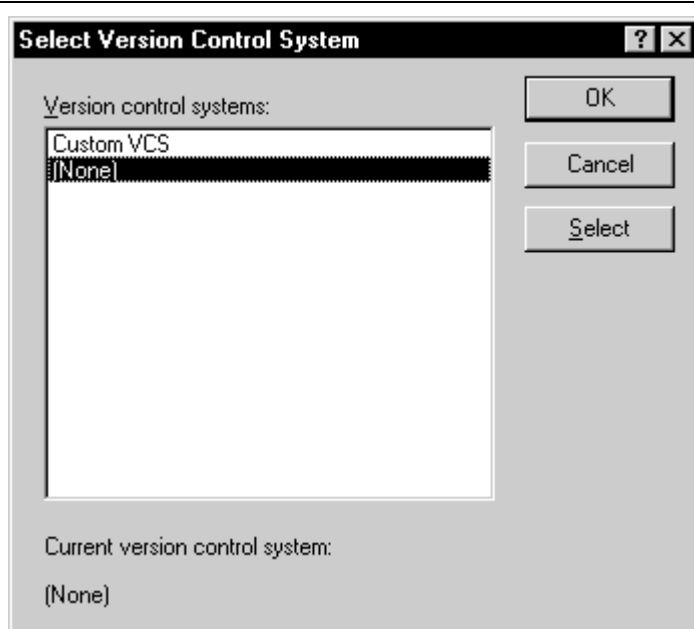


Figure 7.3 Select Version Control System Dialog

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.

Section 8 Using the Custom Version Control System

The custom version control system is a configurable addition to the Hitachi Embedded Workshop which allows you to connect to a version control system already installed on your machine. To clarify further, the Hitachi 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.
 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).

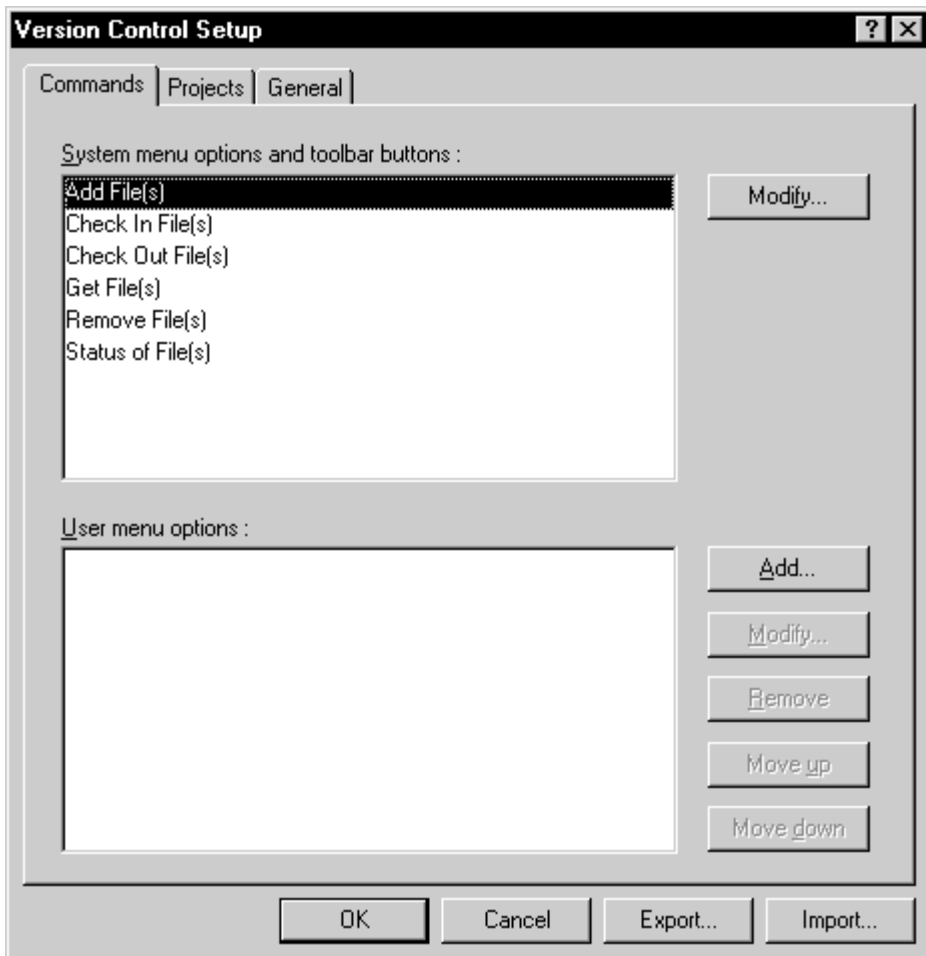


Figure 8.1 Version Control Setup Dialog, Commands Tab

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.

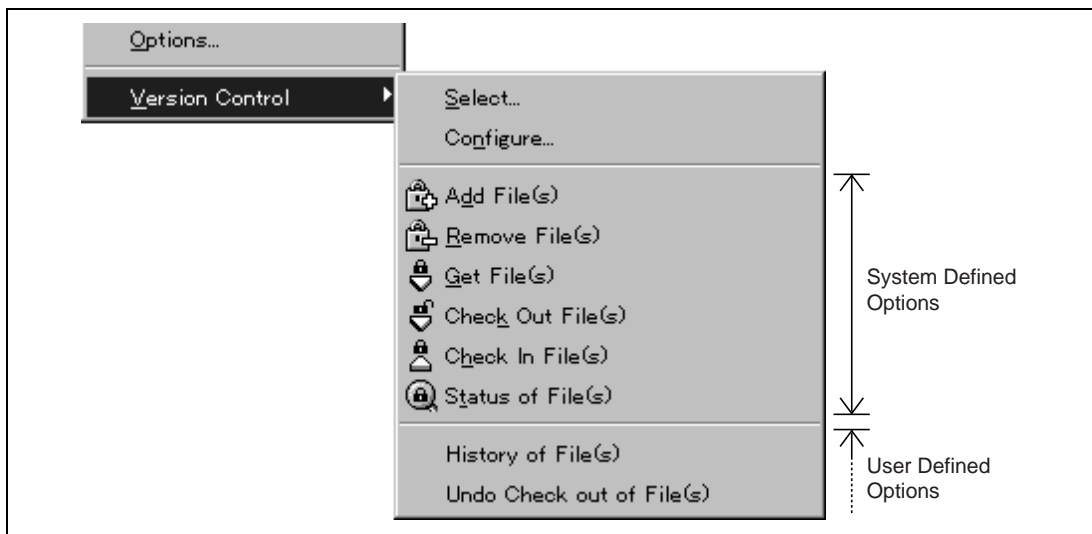


Figure 8.2 Version Control Sub-Menu

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 and Toolbar Button Actions

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

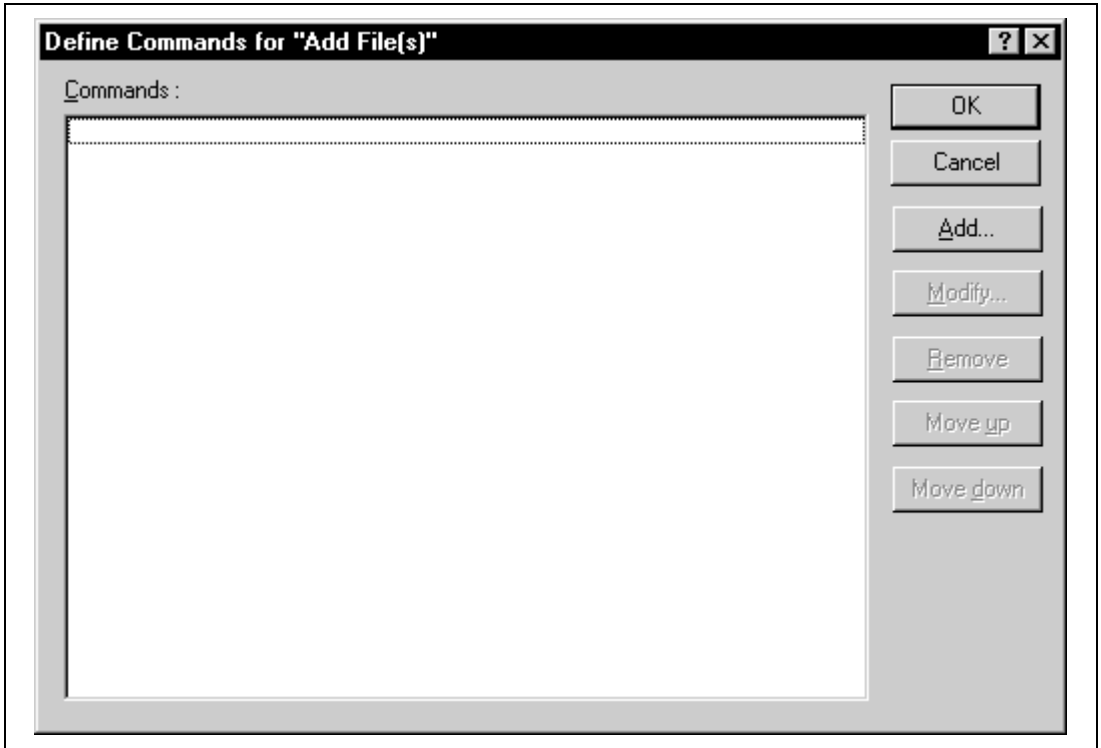


Figure 8.3 Modify System Menu Option (Example)

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”.

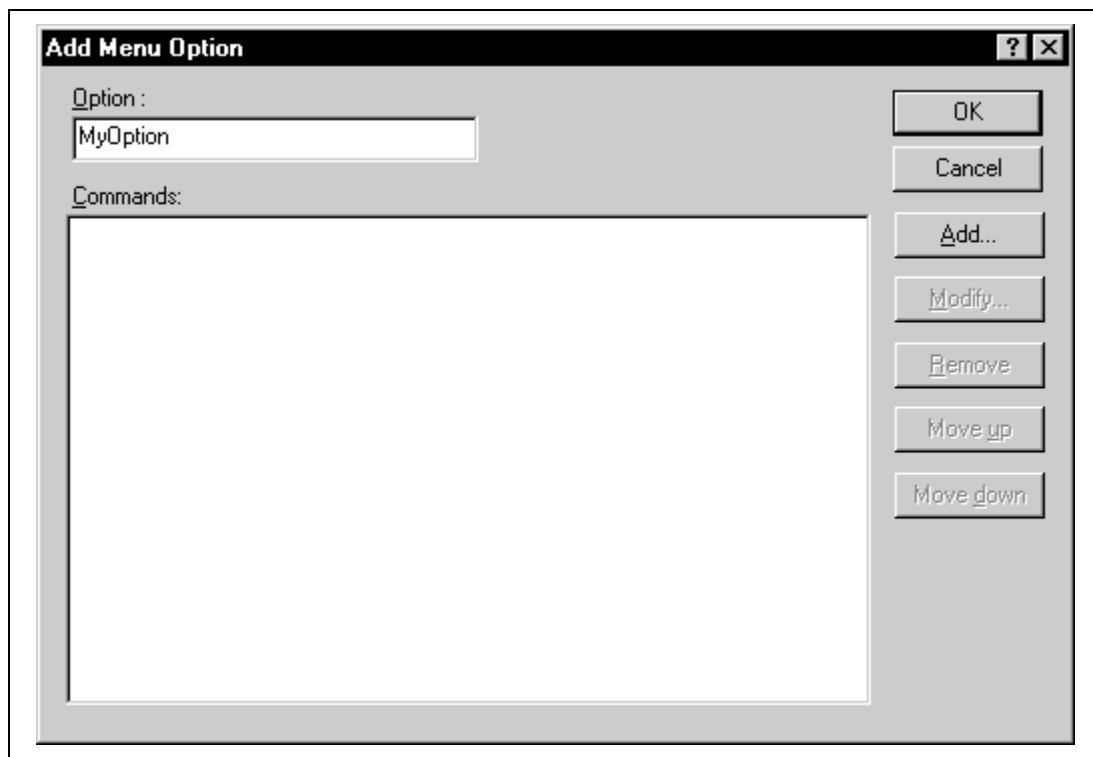


Figure 8.4 Add Menu Option Dialog

- ➡ 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 8.3 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 figures 8.3 and 8.4. In either case, the dialog shown in figure 8.5 is invoked.

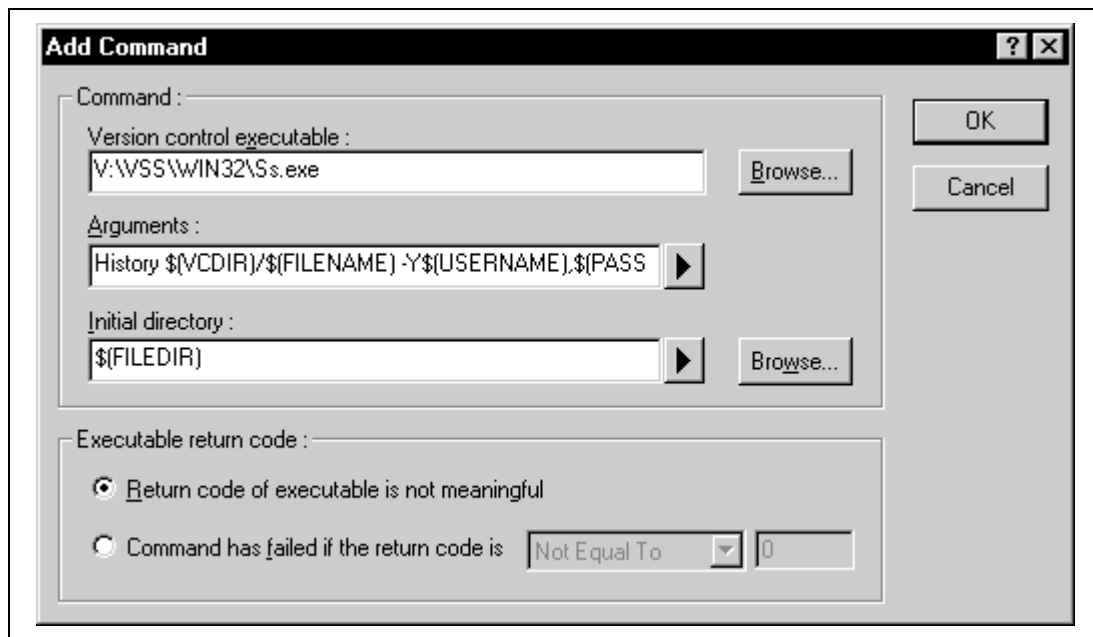


Figure 8.5 Add/Modify Command Dialog

- ➡ 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 executable 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 B, “*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.

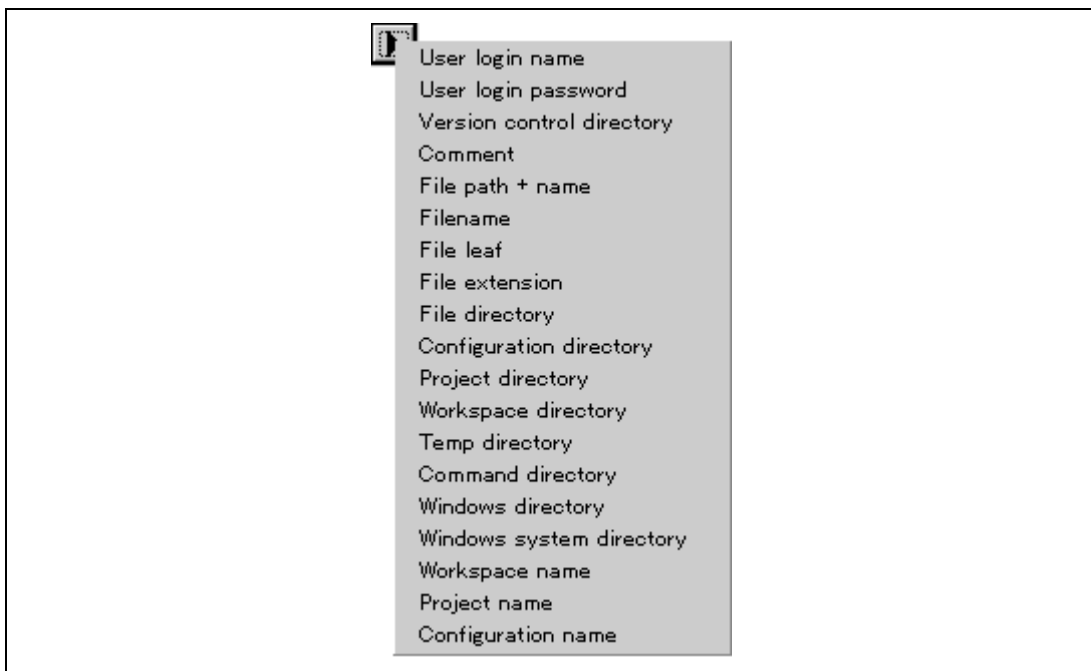


Figure 8.6 Arguments Field Placeholder Pop-up Menu

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).

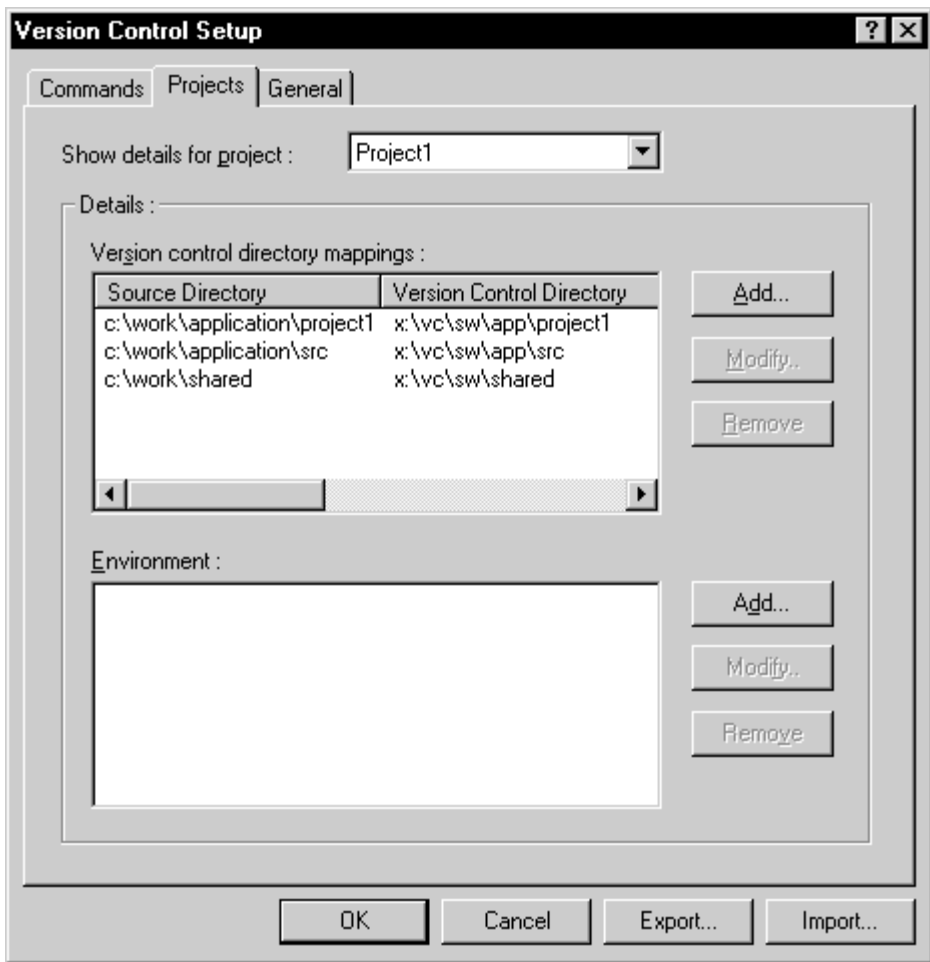


Figure 8.7 Version Control Setup Dialog, Projects Tab

- ➡ 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”.

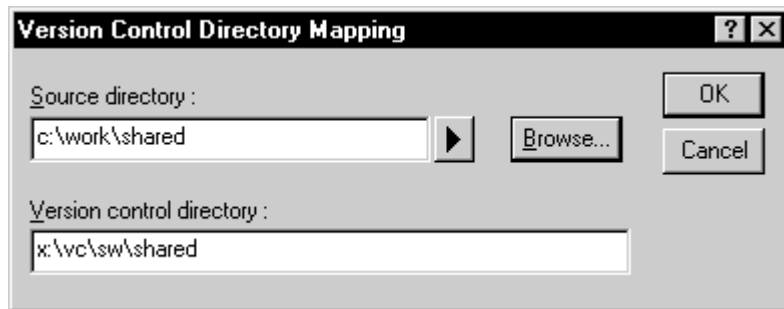


Figure 8.8 Version Control Directory Mapping Dialog

- 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:\).

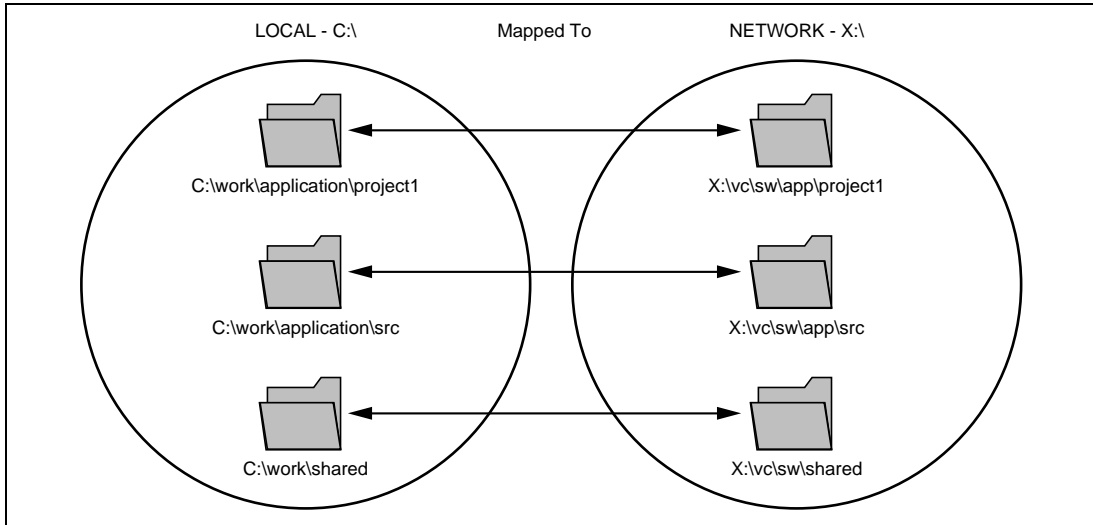


Figure 8.9 Example Mappings

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)

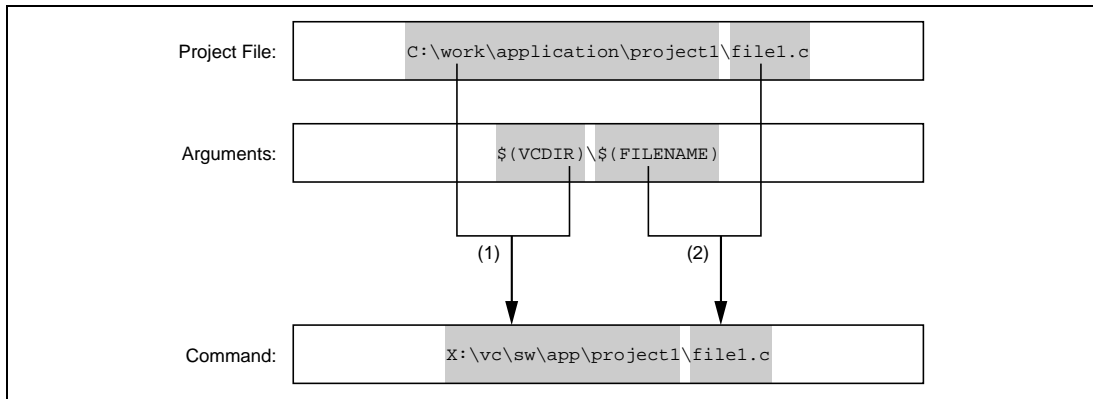


Figure 8.10 Version Control Directory Mapping

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 8.11 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.

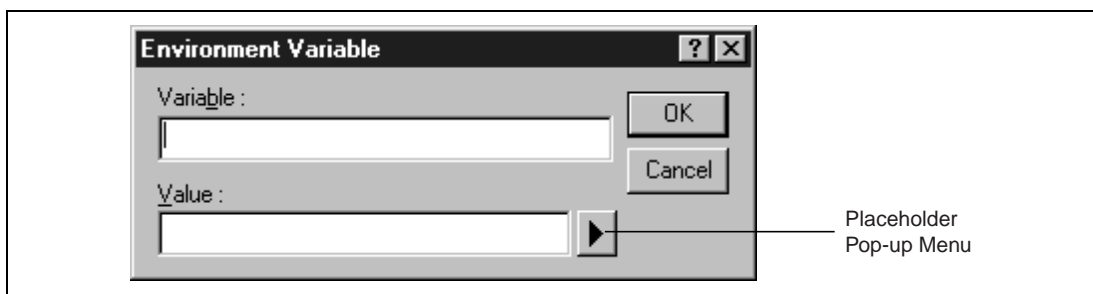


Figure 8.11 Environment Variable Dialog

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).

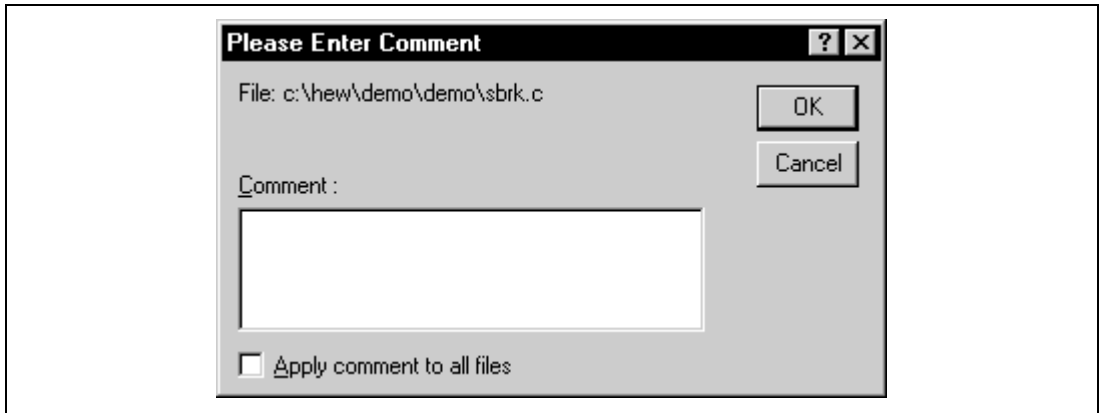


Figure 8.12 Please Enter Comment Dialog

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).

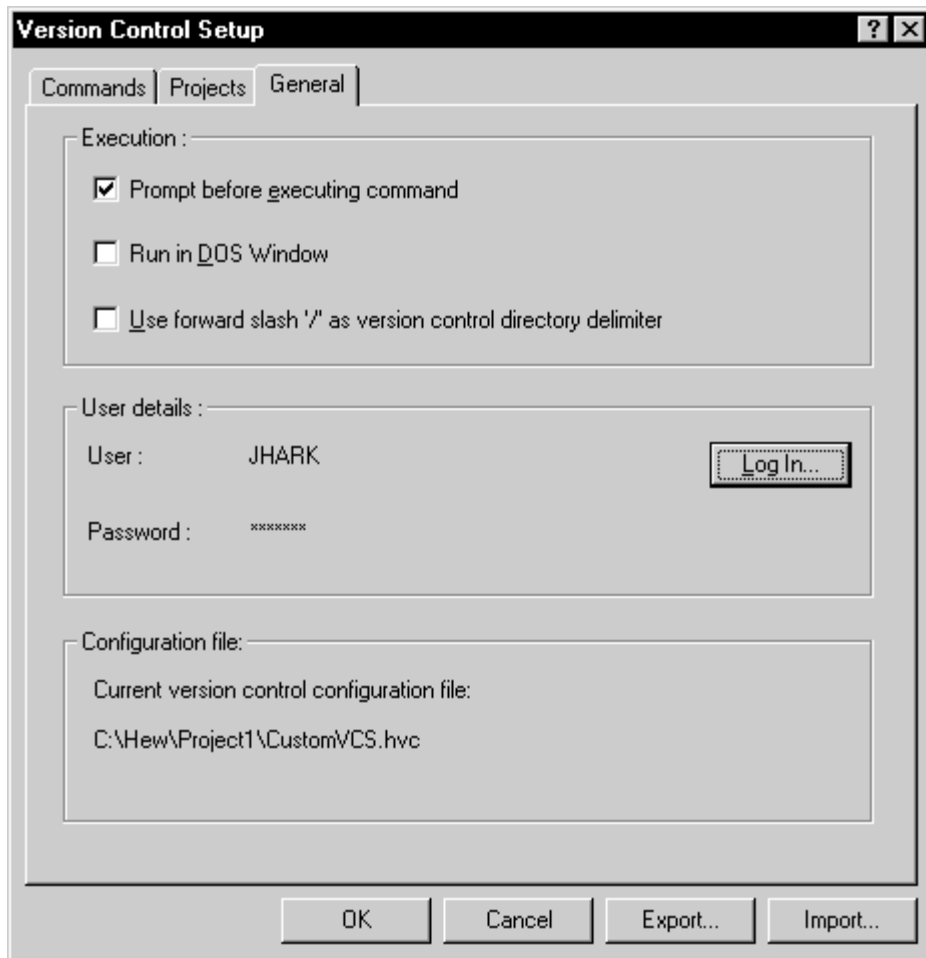


Figure 8.13 Version Control Setup Dialog, General Tab

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.

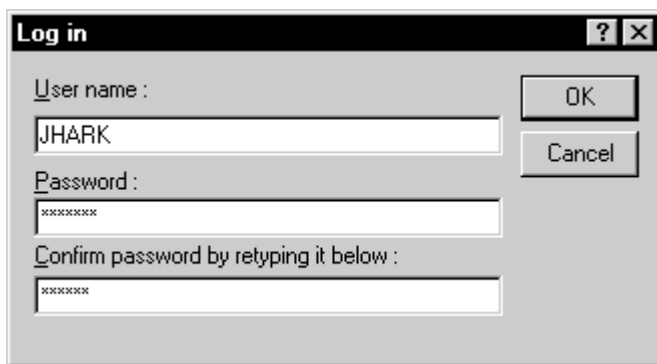


Figure 8.14 Log in Dialog

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.

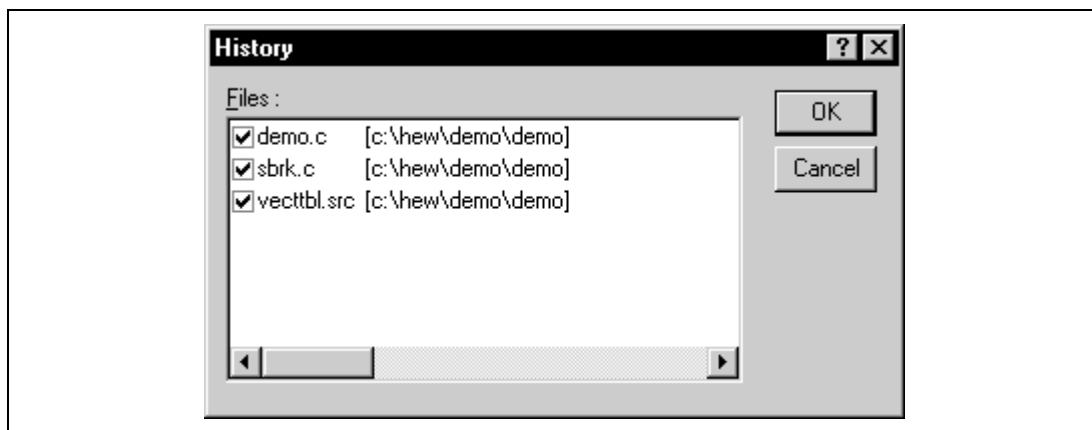


Figure 8.15 Command Prompt Dialog (Example)

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”.

Appendix A Regular Expressions

The Hitachi 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 A.1 and are detailed in the following pages.

Table A.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 “smar 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 B Placeholders

This appendix describes how to use the placeholders, a feature provided by several of the Hitachi Embedded Workshop components.

B.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.

B.2 Inserting a Placeholder

Placeholders can only be entered into three specific edit fields within the HEW (figures B.1, B.2 and B.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.

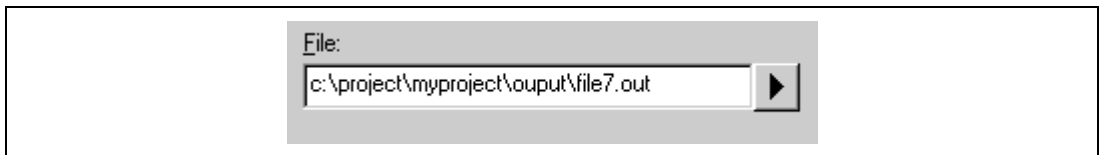


Figure B.1 Placeholder Pop-up Menu

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 B.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 B.3 Placeholder Drop-down List

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

B.3 Available Placeholders

Table B.1 lists the placeholders and their meanings.

Table B.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 B.2.

Table B.2 Placeholder Expansions (Example)

| Placeholder | Expanded placeholder (example) |
|--------------------|---|
| \$(FULLFILE) | c:\hew\workspace\project\file.src |
| \$(FILEDIR) | c:\hew\workspace\project |
| \$(FILENAME) | file.src |
| \$(FILELEAF) | file |
| \$(EXTENSION) | src |
| \$(WORKSPDIR) | c:\hew\workspace |
| \$(WORKSPNAME) | workspace |
| \$(PROJDIR) | c:\hew\workspace\project |
| \$(PROJECTNAME) | project |
| \$(CONFIGDIR) | c:\hew\workspace\project\debug |
| \$(CONFIGNAME) | debug |
| \$(HEWDIR) | c:\hew |
| \$(TCINSTALL) | c:\hew\toolchains\hitachi\sh\511 |
| \$(TOOLDIR) | c:\hew\toolchains\hitachi\sh\511 |
| \$(TEMPDIR) | c:\Temp |
| \$(WINDIR) | c:\Windows |
| \$(WINSYSDIR) | c:\Windows\System |
| \$(EXEDIR) | v:\vc\win32 |
| \$(USERNAME) | JHARK |
| \$(PASSWORD) | 214436 |
| \$(VCDIR) | "c:\project" is mapped to "x:\vc\project" |
| \$(COMMENT) | "Please Enter Comment" dialog is invoked |
| \$(LINE) | 12 |

In table B.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\hitachi\sh\511". This is referred to as \$(TCINSTALL) on the option setting dialogs of the **[Options]** menu and as \$(TOOLDIR) on the Tools Administration dialog.

- the Windows® 95 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.

B.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 C Using Visual SourceSafe

This appendix describes how to connect the custom version control system to Visual SourceSafe version 4.0 and above. This appendix shows one of examples in using the custom version control system.

C.1 Overview

This appendix details the creation of 7 Visual SourceSafe commands:

Table C.1 Commands

| Command | Description |
|------------------|---|
| Add | Add file to SourceSafe |
| Delete and Purge | Delete and purge file from SourceSafe |
| Get | Retrieve a read only copy of the file from SourceSafe |
| Check Out | Retrieve a editable copy of the file from SourceSafe |
| Check In | Commit changes of file to SourceSafe |
| Status | View the status of file in SourceSafe |
| History | Display information on revision history of file |

Table C.2 lists the paths that are used in this appendix.

Table C.2 Paths

| Item | Location |
|--|---------------------|
| Location of Visual SourceSafe .EXE | V:\VSS\DOS\SS.EXE |
| Location of Visual SourceSafe Project | \$/Sample |
| Location of project (i.e. working directory) | C:\Current\Appendix |

This tutorial will lead you through the process of setting up Visual SourceSafe in the following steps:

- Setting up the Visual SourceSafe project
- Identify the commands to be used
- Defining the commands within the HEW
- Executing the commands

C.2 Setting up the Visual SourceSafe Project

The Visual SourceSafe project must exist before any commands can be executed. To do this, invoke Visual SourceSafe and create the project as directed by the documentation of Visual SourceSafe. In this example, create a new project, “\$/Sample” in Visual SourceSafe.

C.3 Identifying Commands to be Used

The first step in using a version control system is to identify the commands to be used and the parameters which those commands need. Table C.3 shows the commands and the parameters which are required.

Table C.3 Commands and Parameters

| Visual SourceSafe Command | Parameters to SS.EXE |
|---------------------------|--|
| Add | CP <project> -Y<username>,<password> Add <local file> -C- -Y<username>,<password> |
| Delete and Purge | Delete <file> -Y<username>,<password> Purge <file> -Y<username>,<password> |
| Get | Get <file> -Y<username>,<password> |
| Check Out | Checkout <file> -Y<username>,<password> |
| Check In | Checkin <file> -Y<username>,<password> |
| Status | Status <file> -Y<username>,<password> |
| History | History <file> -Y<username>,<password> |

All of the commands require the same five parameters:

| | |
|--------------|---|
| <project> | The project in Visual SourceSafe |
| <local file> | The location of the file in local directory |
| <file> | The location of the file in Visual SourceSafe |
| <username> | The current user's login name |
| <password> | The current user's password |

C.4 Defining Commands

Before the menu options and commands can be defined, the custom version control system must be selected. Select [**Tools->Version Control->Select...**] to invoke the “Select Version Control System” dialog. Select “Custom VCS” from the “Version control systems” list, click the “Select” button and then click “OK” to close the dialog. Next, the version control system must be configured so that it can be used with the Visual SourceSafe commands listed in table C.3.

C.4.1 Defining the “History” command

Select [**Tools->Version Control->Configure...**] to invoke the “Version Control Setup” dialog (you’ll notice that the “User menu options” list is empty by default). Click the “Add...” button to create a new menu option (the dialog shown in figure C.1 will be invoked).

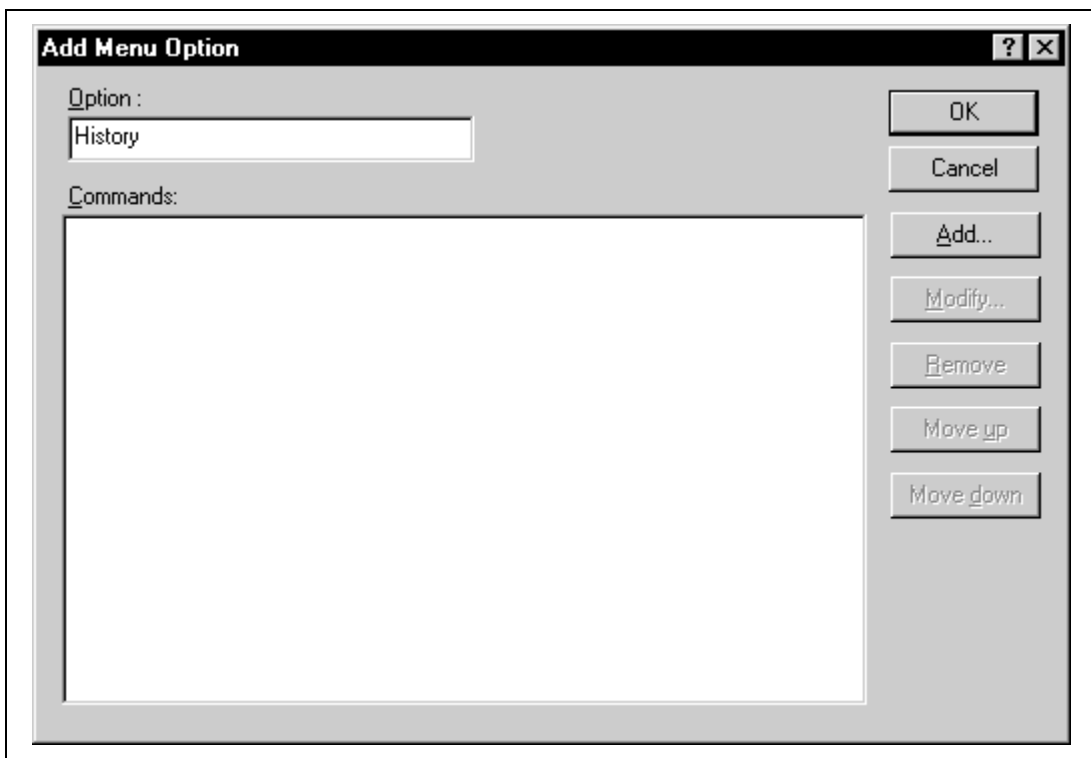


Figure C.1 Add Menu Option Dialog

Enter the menu option “History” into the “Option” field as shown in figure C.1. Click the “Add...” button to create a new command which will be associated with the “History” menu option (the dialog shown in figure C.2 will be invoked).

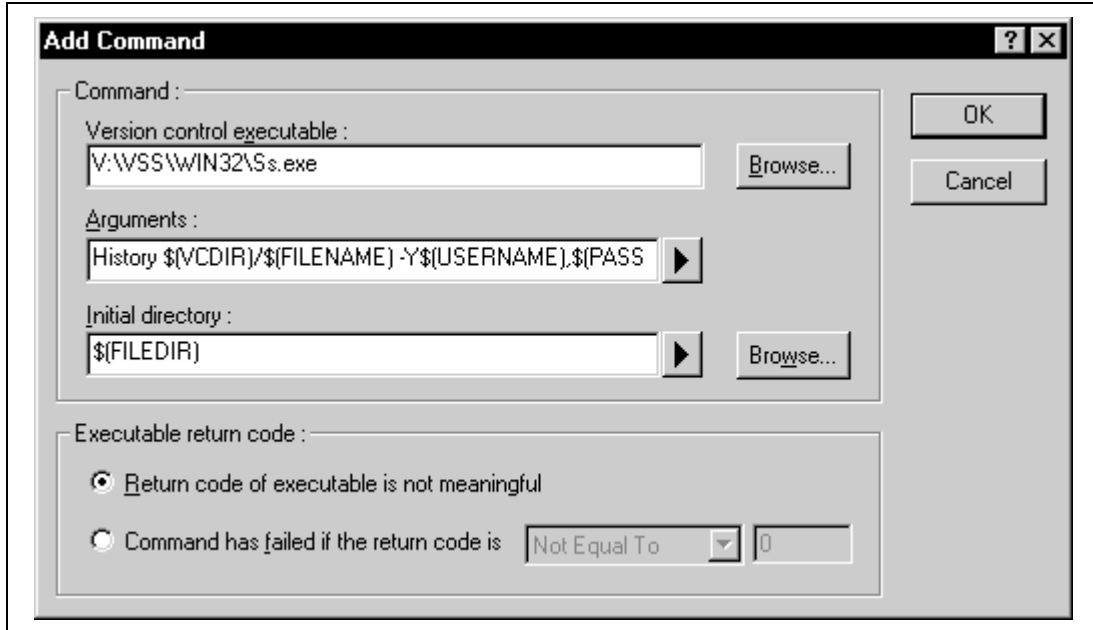


Figure C.2 Add Command Dialog

Fill in the “Add Command”, dialog as indicated below.

➤ To define the History command:

1. Enter the path of the Visual SourceSafe DOS executable (SS.EXE) into the “Version control executable” field. Alternatively, browse to it graphically by clicking the top “Browse...” button.
2. Enter the following string into the “Arguments” field (see below for full explanation):
`"History $(VCDIR)/$(FILENAME) -Y$(USERNAME),$(PASSWORD)"`
3. Set the “Initial directory” field to “\$(FILEDIR)”.
4. Set the “Return code of executable is not meaningful” option.

Click “OK” on the “Add Command” dialog and then “OK” on the “Add Menu Option” dialog to confirm creation of the new menu option.

At this point, the parameters previously entered into the “Arguments” field merit further explanation. Figure C.3 divides the arguments up into three sections:

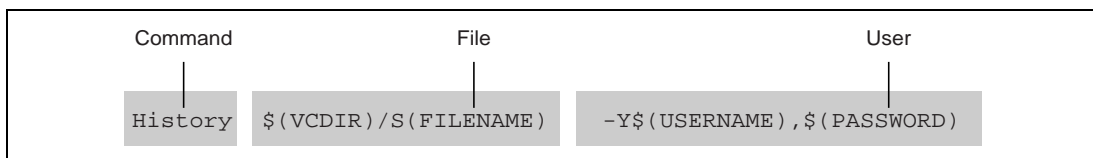


Figure C.3 History Command

C.4.2 Command

This is the Visual SourceSafe command “History”.

C.4.3 File

This is composed of two parts. The first, \$(VCDIR), is a placeholder which is later substituted with the path of the file in Visual SourceSafe (see the next section for further information). The second part, \$(FILENAME), is the filename with no path (e.g. MAIN.C). Notice that the separator between the two files is a forward slash character ('/'). This is because all paths within the Visual SourceSafe system use forward slashes as opposed to standard backslashes in DOS ('\\').

C.4.4 User

This section passes the user name and password which are required for security reasons.

This procedure can be repeated for the remaining commands (see table C.4 for the parameters required).

Table C.4 Menu Options and Parameters

| Menu Option | Parameters to SS.EXE |
|-------------------|---|
| Add File(s) | CP \$(VCDIR) -Y\$(USERNAME),\$(PASSWORD) Add \$(FULLFILE) -C- -Y\$(USERNAME),\$(PASSWORD) |
| Remove File(s) | Delete \$(VCDIR)/\$(FILENAME) -Y\$(USERNAME),\$(PASSWORD) Purge \$(VCDIR)/\$(FILENAME) -Y\$(USERNAME),\$(PASSWORD) |
| Get File(s) | Get \$(VCDIR)/\$(FILENAME) -Y\$(USERNAME),\$(PASSWORD) |
| Check Out File(s) | Checkout \$(VCDIR)/\$(FILENAME) -Y\$(USERNAME),\$(PASSWORD) |
| Check In File(s) | Checkin \$(VCDIR)/\$(FILENAME) -Y\$(USERNAME),\$(PASSWORD) |
| Status of File(s) | Status \$(VCDIR)/\$(FILENAME) -Y\$(USERNAME),\$(PASSWORD) |

C.4.5 Setting up directory mappings

As the placeholder “\$(VCDIR)” has been used in the commands, directory mappings must be specified between those in Visual SourceSafe and the working directory (i.e. local hard disk). If the “Version Control Setup” dialog is not already invoked then invoke it via [**Tools->Version Control->Configure...**]. Select the “Projects” tab (the dialog shown in figure C.4 will be displayed).

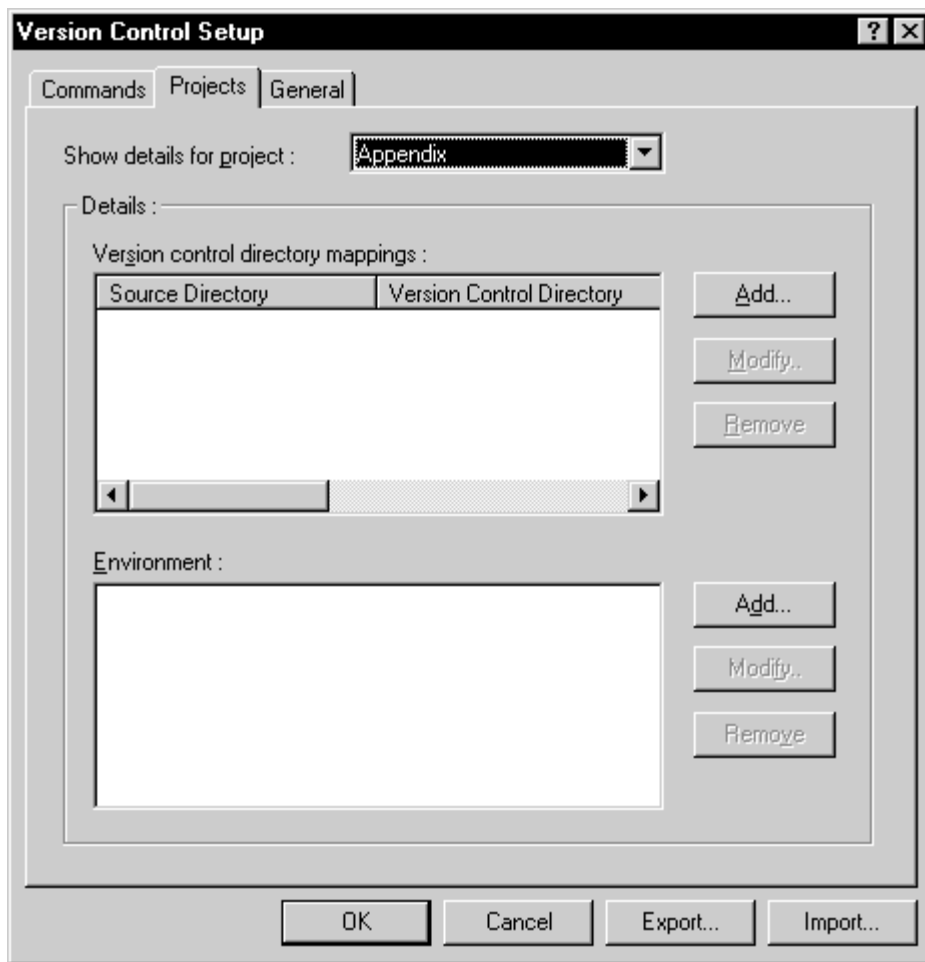


Figure C.4 Version Control Setup Dialog, Projects Tab

This tab allows the directory mappings and environment to be specified on a project by project basis. In this example, there is only one project in the workspace, named “Appendix”, and it is already selected by default in the “Show details for project” drop-down list box. Visual SourceSafe does not require any environment for its commands so the “Environment” list box will be left empty.

The “\$(VCDIR)” placeholder will be replaced by HEW when the version control command is executed. It examines the path of the source file in the local, working, directory (i.e. in this case C:\Current\Appendix) and compares it to those listed in the version control directory mappings. If it is not present then the HEW will report an error and request that a mapping be specified for that directory. If it is present, then the mapped path in the version control tool is replaced (i.e. in this case \$/Sample). Thus, the version control system allows you to map a file in the local, working directory to any directory in the version control system as long as the placeholder \$(VCDIR) is used and the appropriate mappings are specified.

In this example, there is only one mapping, as illustrated by figure C.5. The local files in C:\Current\Appendix are mapped to those within Visual SourceSafe in \$/Sample.

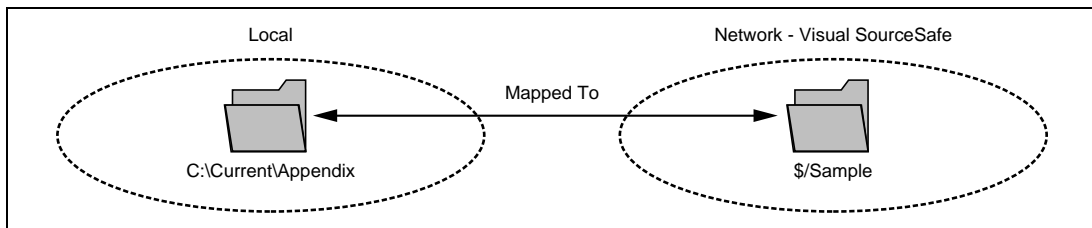


Figure C.5 Mappings

Click the top “Add...” button beside the “Version Control Setup Mapping” list box to create a new mapping (the dialog shown in figure C.6 will be invoked).

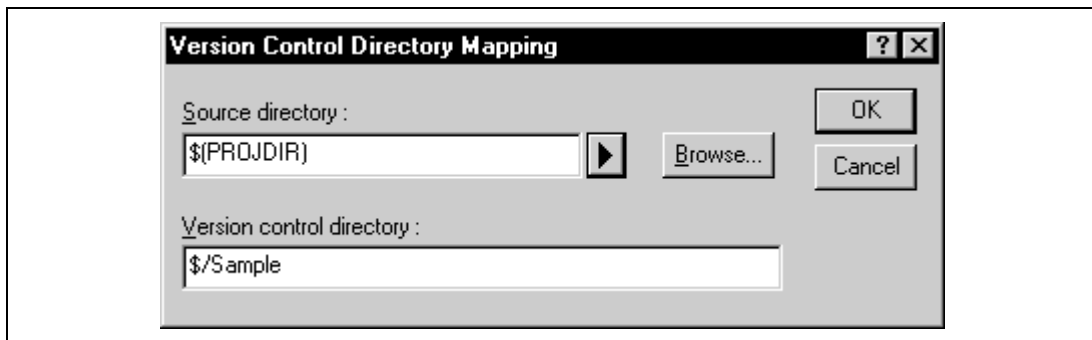


Figure C.6 Version Control Directory Mapping Dialog

This dialog is relatively simple, it requests the two directories involved in the mapping.

In the first field, “Source directory”, enter the full path of the source (i.e. working) directory. In this example, as the only mapping is from the project directory, the placeholder “\$(PROJDIR)” is entered to keep the file paths relative. This is just in case the project moves to a new location in the future, or if located elsewhere on another user’s machine. In the second field, “Version control directory”, enter the full path of the directory (i.e. project) within Visual SourceSafe. Click “OK” to define the mapping.

If you have a project file which is not in the project directory, it is one of the viable solutions to add another entry in the mapping. For example, suppose C:\Current\Header\define.h, which is located outside the project directory, is included by a project file in C:\Current\Appendix. To add this file to the \$/Sample project, add a mapping between C:\Current\Header and \$/Sample as shown in figure C.7.

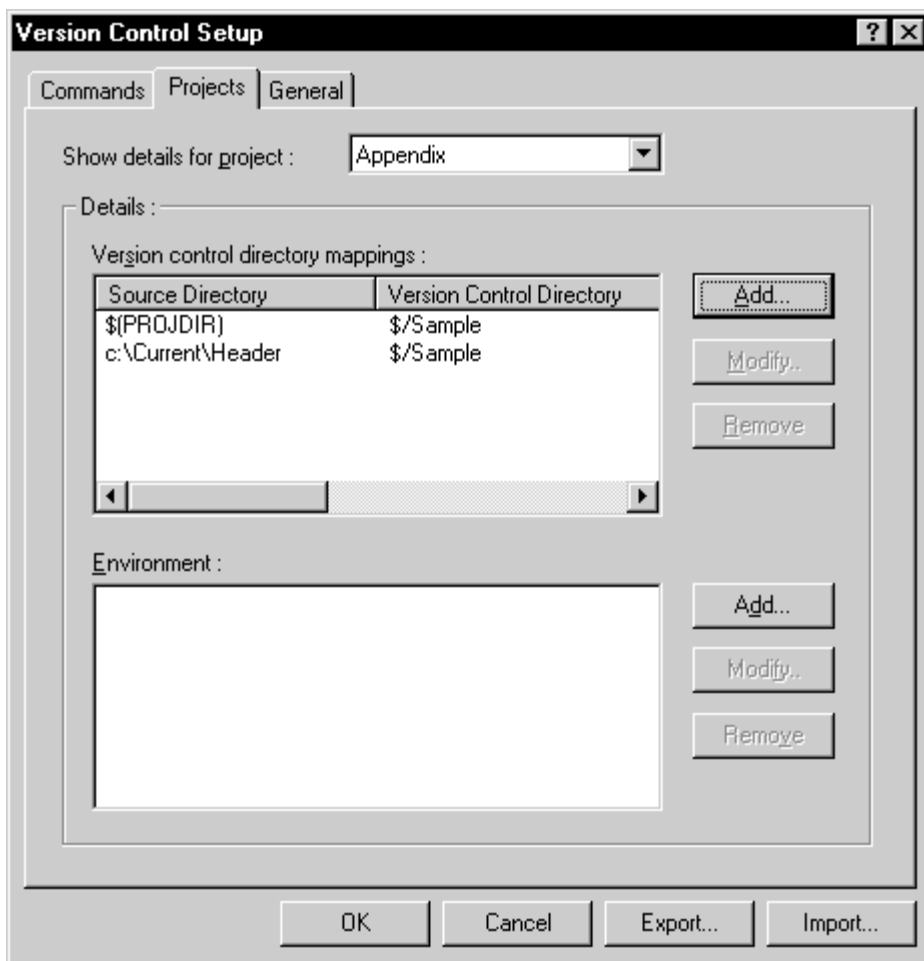


Figure C.7 Version Control Setup Dialog, Projects Tab

C.4.6 Setting environment variables

If you want to use the Visual SourceSafe database (i.e. SRCSAFE.INI) other than your default database, you need to assign the directory path of another database to the SSDIR environment variable. To do so, invoke the “Version Control Setup” dialog via [**Tools->Version Control->Configure...**]. Select the “Projects” tab. Click the top “Add...” button beside the “Environment” list box to define an environment variable (the dialog shown in figure C.8 will be invoked).

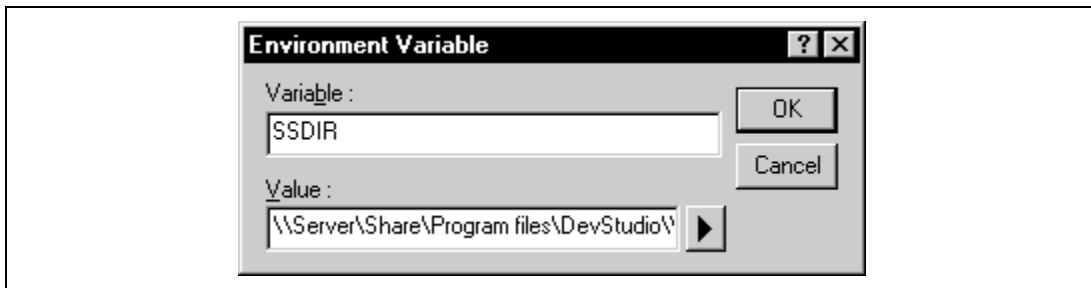


Figure C.8 Environment Variable Dialog

In the first field, “Variable”, enter “SSDIR”, the environment variable to point to the directory in which SRCSAFE.INI exists. In the second field, specify the full directory path of SRCSAFE.INI excluding the filename. Click “OK” to define the environment variable. Then the definition will be shown in the “Environment” list box of the “Projects” tab of the “Version Control Setup” dialog (figure C.9).

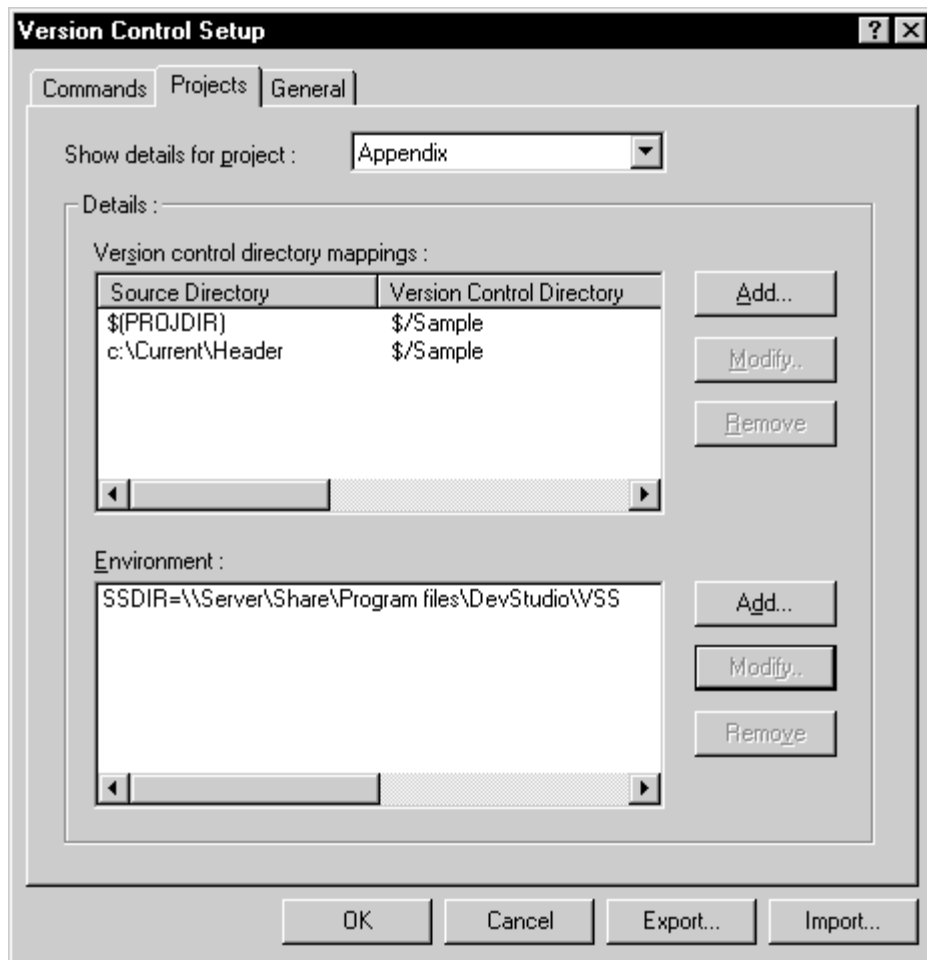


Figure C.9 Version Control Setup Dialog, Projects Tab

C.4.7 And finally...

The “General” tab of the “Version Control Setup” dialog (see figure C.10) contains several important options.

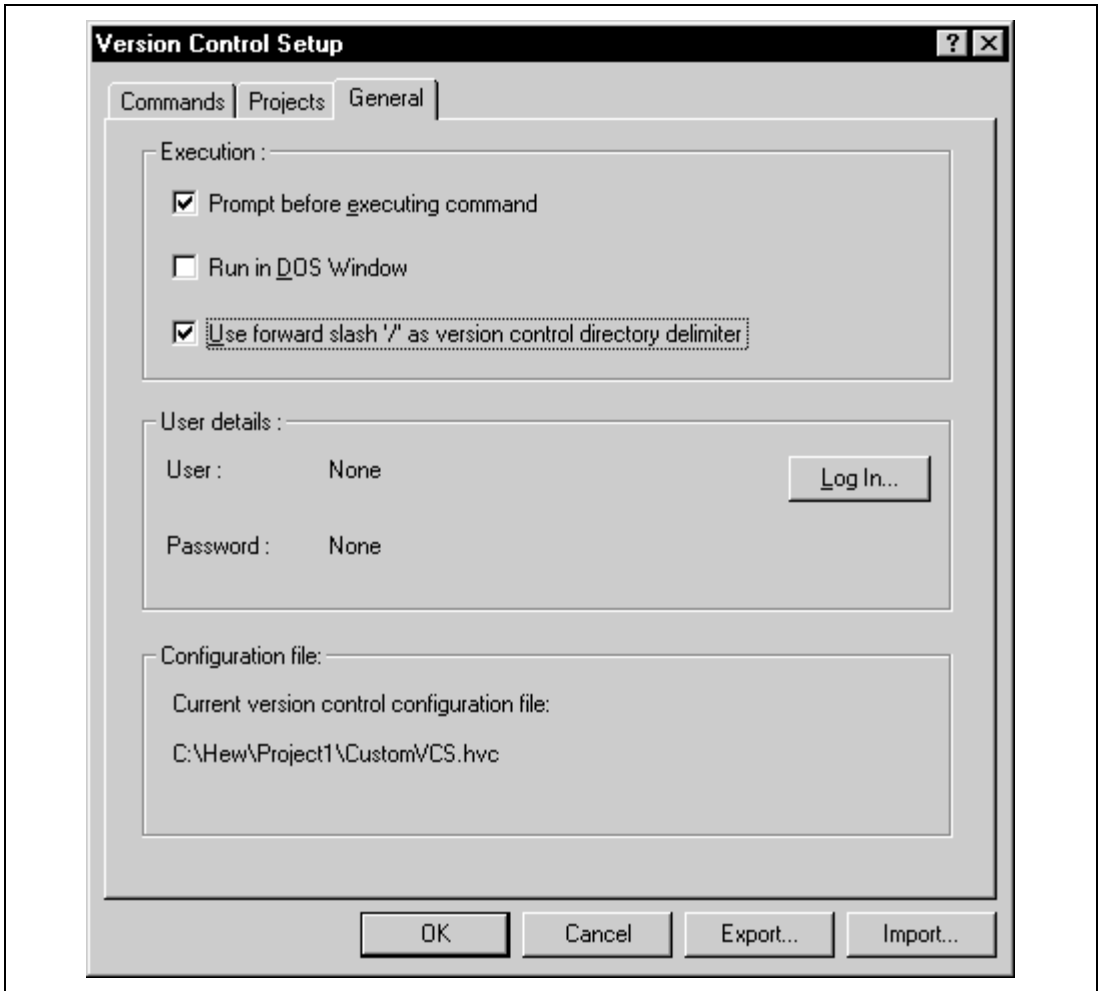


Figure C.10 Version Control Setup Dialog, General Tab

(1) Prompt before executing command

When creating commands for the first time it is worthwhile setting this check box so that you are aware of when version control commands are being executed and which files are involved in the operation.

(2) Use forward slash '/' as version control directory delimiter

When the HEW replaces the “\$(VCDIR)” placeholder, it inserts a slash between the directory and the filename. If the version control system which you are using uses the forward slash character to delimit directories then set this check box.

Note: This check box must be set when using Visual SourceSafe.

(3) User details

You will need to log into the HEW before a version control command containing either the “\$(USERNAME)” or “\$(PASSWORD)” placeholders can be executed. You can log in at this stage by clicking the “Log in...” button. Alternatively, you can attempt to execute the command, at which point the HEW will prompt you for the information.

C.5 Executing Commands

To execute the command, select the files that you want to apply the command to from the “Projects” tab of the “Workspace” window and then select the desired version control menu option. The output from the version control tool will be displayed in the “Version Control” tab of the “Output” window.

Some further advice:

- As with any aspect of version control, planning is essential. Be sure to clearly identify the directory structure of the project before attempting to use the HEW.
- Start with the most basic commands first. Those which require the least parameters will be the easiest to set-up and will allow you to become more familiar with the HEW version control system.
- If you fail to execute a command, try it without using any placeholders. Ensure that your commands run successfully when the commands, arguments and mappings are all specified without placeholders – they can always be added later.
- When specifying directory mappings, you need only specify the “highest” directory as all files contained in sub-directories of “mapped” directories will be mapped accordingly.
- The “Version Control” tab of the “Output” window is read only, that is, you can't type anything into it. Consequently, if the version control command executed prompts for input, you will not be able to enter a response and you must select **[Build->Terminate Current Tool]** to forcibly terminate the command. There are two solutions to this problem; first, investigate a way to pass the information which the tool is requesting via the command line. Second, set the “Run in DOS window” check box which is located on the “General” tab of the “Version Control Setup” dialog before executing the command. This will allow you to respond to version control requests.

Appendix D HMAKE User Guide

This appendix describes the specification of hmake. The following section describes the command line that should be used to execute the hmake program on a file using none or more of the available options.

D.1 Command line

D.1.1 Basic structure

The command line must be of the following syntax:

```
hmake <makefile> <parameter list>
```

If a file is specified without an extension then “.mak” will be appended to it. The parameter list may include none or more of the parameters listed in the following section. The parameters list may appear before the make file name if you wish. Each parameter must be separated by at least one white space character. Parameters are not case sensitive. If no parameters are given and no file is given then help information will be displayed.

D.1.2 Exit codes

If there are any syntax errors in the make file being executed or if any process executed whilst running the make file returns an invalid error code then hmake will exit with code 1. Otherwise hmake will exit with code 0 (See below for file syntax and how to specify exit code conditions).

D.1.3 Parameters

The following table shows the available parameters and their function:

Characters

| Parameter | Function |
|-----------|--|
| /A | Execute all commands regardless of input/output file status. Equivalent to a Build All. |
| /N | Use status of input/output files to calculate what commands need to be executed (as normal) and then display the commands but do not execute them. |
| /? | Display help info. |

D.2 File syntax

There are four basic types of statement used in a hmake file, the variable declaration the description block, the comment and the message command. These can be combined in any order to produce a hmake file but a variable must be declared in a variable declaration before it is used in a description block or other variable declaration. The first “all” statement used in nmake files is not required in a hmake file. Commands are executed in order, as they appear in the make file. Note: the “→” character is used to show where a tab character must be used in order to keep the make file syntactically correct.

D.2.1 Variable declarations

A variable declaration declares a variable which can then be used in any statement throughout the rest of the hmake file. A declaration has the following syntax:

```
<variable name> = <value>
```

Any number of white space characters are allowed between the variable name and the ‘=’ sign and the value and the ‘=’ sign. The value may be split over several lines using a ‘\’ character. If the value contains ‘\’ characters within the main text then these are taken literally. Only ‘\’ characters followed by a new line are considered to indicate a value wrapping over more than one line.

There follows some examples of valid variable declarations:

```
EXECUTABLE = c:\dir\prog.exe
OUTPUT = c:\dir2\file1.out
INPUT = c:\dir2\file1.c
DEPEND = c:\dir2\file2.h \
        c:\dir2\file3.h \
        c:\dir2\file4.h
```

In order to use a variable later in the hmake file write the variable name with “\$(” added to the front and “)” added to the back. The variable name (along with the “\$()” characters) will be substituted with the variable’s value. For examples of this see later under description blocks. Only alphanumeric characters and underscore characters are allowed in variable names. It is possible to use a variable inside the declaration of a different variable but all variables must be declared before they are used.

D.3 Description blocks

D.3.1 Basic outline

A description block specifies one or more targets, zero or more dependants and a list of commands which should be executed if the newest dependent is newer than the newest target. If none of the targets exist and/or none of the dependants exist then the commands will always be executed. It is not necessary to specify any dependants if you wish the commands to always be executed. A description block has the following syntax:

```
<target1> <target2> ... : <dependant1> <dependant2> ...  
→ <command1>  
→ <command2>  
→ ...  
→ <commandn>
```

Any number of white space characters are allowed between the last target and the ‘:’ character and the first dependant and the ‘:’ character. No white space is allowed before the first target. Each target and each dependant must be separated by at least one white space character. A tab character must be present at the start of a line containing a command. Variables may be used in a description block using the syntax specified above under variable declarations.

There follows some examples of valid description blocks (one of which uses the variable specified above under variable declarations):

```
c:\dir1\file1.obj : c:\dir1\file1.c c:\dir1\file1.h  
→ gcc c:\dir1\file1.c  
$(OUTPUT) : $(INPUT) $(DEPEND)  
→ $(EXECUTABLE) $(INPUT)
```

It does not matter that CHANGEDIR and SETENV are not file names. They will be treated as files that do not exist and so the commands will always be executed.

D.3.2 Sub command files

If you wish hmake to generate a sub command file for you then the command part of the description block should be specified as follows (this replaces <commandn> above):

```
→ <command start> <<
→ <sub command1>,
→ <sub command2>,
→ ...
→ <sub commandn>
<<<command end>
```

This will generate a sub command file, in the windows temporary directory, which will contain the lines <sub command1>, <sub command2> etc. This command file will be deleted once the make process has completed. The name of the command file will be substituted for all the text between the two "<<"s. You do not have to worry about the name of the sub command file. This is generated by hmake.

For example:

```
c:\dir1\file1.obj : c:\dir1\file1.c c:\dir1\file1.h
→ gcc @<<
→ -c -o c:\dir1\file1.obj c:\dir1\file1.c
<<
```

If the sub command file generated has the name "c:\temp\hmk111.cmd" then the following would be executed by hmake (assuming c:\dir1\file1.obj is out of date):

```
gcc @c:\temp\hmk111.cmd
```

The command file (c:\temp\hmk111.cmd) would contain:

```
-c -o c:\dir1\file1.obj c:\dir1\file1.c
```

It is possible to include more than one command in the description block and to use combinations of the standard, and sub command file commands.

D.4 Comments

A '#' character signifies a comment. When this character appears as the first character on a line the rest of the line (up until the next new line character) is ignored. There follows examples of valid comments:

```
# My hmake file

# Variable declaration
OUTPUT= c:\dir1\file1.obj
# Descriptor
$(OUTPUT) : c:\dir1\file1.c c:\dir1\file1.h
→ set VAR1=value1
→ gcc c:\dir1\file1.c
```

A comment must occupy its own line in the hmake file. It is not possible to put comments on the end of other statements.

D.5 Message commands

The message command is used to output a line of text to standard out whilst a make file is executing. These text lines will be output in the order they appear in the make file, in amongst output from any executables being executed as appropriate. No buffering of output text will take place. A message command has the following syntax:

```
!MESSAGE <text to output>
```

A new line character is assumed to come after the last character in <text to output>. Any white space between !MESSAGE and <text to output> will be ignored. There follows an example of a valid message command:

```
!MESSAGE Executing C Compiler
```

Index

| | | | |
|--|---------------|-----------------------------------|----------|
| Administration | 97 | external editor | 117 |
| Arrange Icons | 46 | file group..... | 23 |
| bookmark..... | 57 | File Mappings | 89 |
| Bookmarks Toolbar | 4 | Find | 53 |
| brace | 69 | Find in Files | 9, 54 |
| Build | 9, 32, 33, 91 | float..... | 5, 8 |
| Build All | 32 | Font..... | 63 |
| Build File | 32 | Format..... | 63 |
| Build File Order..... | 82 | General..... | 102, 139 |
| Build Order..... | 79 | Generate log file | 92 |
| Build Phases | 72, 74 | Generate Makefile | 95 |
| Cascade..... | 46 | HDI | 94 |
| Close..... | 51 | help | 11 |
| Close All..... | 46, 51 | Help | 112 |
| Command line | 33 | Hitachi Debugging Interface | 94 |
| Commands..... | 108, 127 | hmake..... | 162 |
| configuration | 29 | HRF | 98 |
| Configure..... | 123 | Import | 142 |
| Configure View | 35 | indentation | 61 |
| Copy | 52 | Information | 102 |
| custom phase | 74, 83 | Initial directory | 33 |
| custom version control system | 125 | Insert Project..... | 38 |
| Cut | 52 | Insert Template | 68 |
| Debugger | 94 | keyword | 64 |
| Default directory for new workspaces.... | 116 | Load Project..... | 42 |
| Define Templates..... | 67 | Log..... | 92 |
| Delete..... | 52 | make..... | 163 |
| Dependent Files..... | 87 | makefile | 95 |
| Dependent Projects | 40 | mapping | 90, 134 |
| Display workspace information dialog on opening workspace | 116 | Match Braces | 69 |
| dock | 5, 7 | Menu | 109 |
| Editor | 60, 118 | modify a custom phase | 80 |
| Editor Toolbar | 3 | move a phase..... | 79 |
| editor window | 45 | multiple phase | 75, 83 |
| Enable auto indentation | 61 | Navigation | 6 |
| Environment | 33, 103 | New..... | 49 |
| environment variable | 137 | new workspace..... | 13 |
| Export | 142 | Next Bookmark..... | 58 |
| extension..... | 22 | Open..... | 50 |
| | | open a workspace..... | 14 |

| | | | |
|---|------------------|--|------------------|
| Options | 84 | Show files in notebook | 46 |
| Output..... | 9 | Show standard library includes | 36 |
| Output Files | 85 | Show workspace information on workspace open | 115 |
| Page Setup | 59 | single phase..... | 75, 83 |
| password..... | 139 | Size | 63 |
| Paste..... | 52 | split a window | 62 |
| phase..... | 17 | Standard Toolbar | 3 |
| placeholder | 133, 145, 147 | status bar | 10 |
| Previous Bookmark | 58 | Stop Build | 33 |
| Print | 58 | Stop build if the no. of errors exceed | 91 |
| project..... | 1 | Stop build if the no. of warnings exceed... | 91 |
| Projects | 5, 134 | syntax coloring..... | 64 |
| Prompt before executing command | 141 | system phase | 74 |
| Prompt before saving files..... | 119 | System Tool..... | 97 |
| Prompt before saving workspace..... | 116 | Tab size..... | 60 |
| Properties..... | 43, 44, 102, 115 | template | 67 |
| properties of a system phase | 79 | Templates Toolbar | 4 |
| Recent Files | 50 | Terminate Current Tool | 33 |
| Redo | 52 | Tile..... | 46 |
| register a component..... | 101 | Toggle Bookmark | 57 |
| Regular Expression..... | 143 | Toolbar..... | 107 |
| remove a phase | 79 | Toolchain | 97 |
| Remove Project | 42 | Toolchain version | 93 |
| rename a project | 44 | Tools Administration | 97, 106 |
| rename a workspace | 43 | Tools Customize | 92, 94, 107 |
| Replace | 55 | Tools Options..... | 60, 91, 114, 118 |
| Run in DOS Window..... | 141 | Undo | 52 |
| Save | 50 | uninstall a component | 104 |
| Save All | 50 | Unload Project | 42 |
| Save As..... | 50 | Unregistering a component..... | 101 |
| Save files before executing any tools | 119 | Update All Dependencies | 34 |
| Save workspace before executing any phases | 116 | Upgrade | 93 |
| search for components | 100 | Use external editor | 118 |
| Search Toolbar | 3 | Use forward slash '/' as version control directory delimiter..... | 141 |
| Select | 122 | Use spaces as tabs | 60 |
| Select All | 52 | user name | 139 |
| Set as Current Project | 38 | Utility Phase..... | 97 |
| Set Current Project | 38 | version control | 121 |
| Show all components..... | 106 | Version Control | 9 |
| Show dependencies under each file..... | 35 | Version Control Toolbar | 4 |
| Show file groups in separate folders..... | 37 | Visual SourceSafe..... | 150 |
| Show file paths | 37 | | |

| | |
|-----------------|----|
| Window | 46 |
| workspace | 1 |

| | |
|-----------------------------|--------|
| Workspace | 5, 114 |
| workspace information | 115 |

High-performance Embedded Workshop User's Manual

Publication Date: 1st Edition, June 1999

2nd Edition, February 2000

Published by: Electronic Devices Sales & Marketing Group
Semiconductor & Integrated Circuits
Hitachi, Ltd.

Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.

Copyright © Hitachi, Ltd., 1999. All rights reserved. Printed in Japan.