

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 2

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

HEW Builder

User's Manual



ADE-702-279A

Rev. 2.0

06/03/02

Hitachi, Ltd.

Cautions

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

Trademarks

Microsoft, MS-DOS, Windows, Windows NT are registered trademarks of Microsoft Corporation.

Visual SourceSafe is a trademark of Microsoft Corporation.

IBM is a registered trademark of International Business Machines Corporation.

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

Document Information

Product Code: S32HEWM

Version: 2.1

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

Copyright © Hitachi, Ltd. 2001. All rights reserved.

About This Manual

This manual describes how to use the Hitachi Embedded Workshop and the builder functionality. It covers all aspects of the build process from creating your project, adding and removing files and building the files. For information on the “look and feel” of the Hitachi Embedded Workshop or customizing the HEW environment please refer to the main HEW user manual.

Document Conventions

This manual uses the following typographic conventions:

Table 1 **Typographic Conventions**

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

Contents

Cautions.....	iii
1. Overview	1
1.1 Workspaces, Projects and Files	1
1.1.1 The Toolbars	2
1.1.2 The Workspace Window	3
1.1.3 The Output Window	4
1.2 Launching the HEW	5
1.3 Creating a New Workspace	6
1.4 Opening a Workspace	7
1.5 Saving a Workspace	7
1.6 Closing a Workspace	8
1.7 Using Old Workspaces	8
1.8 Exiting the HEW	8
2. Build Basics.....	9
2.1 The Build Process	9
2.2 Project Files.....	10
2.2.1 Adding Files to a Project	11
2.2.2 Removing Files from a Project.....	13
2.2.3 Excluding a Project File from Build	15
2.2.4 Including a Project File in Build	15
2.3 File Extensions and File Groups.....	15
2.4 Specifying How to Build a File	22
2.5 Build Configurations.....	23
2.5.1 Selecting a Configuration	24
2.5.2 Adding and Deleting Configurations	24
2.6 Building a Project.....	26
2.6.1 Building a Project	26
2.6.2 Building Individual Files	26
2.6.3 Stopping a Build	27
2.6.4 Building Multiple Projects	27
2.6.5 The Output Window	28
2.6.6 Controlling the Content of the Output Window	28
2.7 File Dependencies	29
2.8 Configuring the Workspace Window	29
2.8.1 Show Dependencies under Each File	29
2.8.2 Show Standard Library Includes	30
2.8.3 Show File Paths	31
2.9 Setting the Current Project.....	31
2.10 Inserting a Project into a Workspace.....	31
2.11 Specifying Dependencies between Projects	33
2.12 Removing a Project from a Workspace.....	34
2.13 Relative projects paths in the workspace.....	35
3. Advanced Build Features	36
3.1 The Build Process Revisited.....	36
3.1.1 What is a Build?.....	36
3.2 Creating a Custom Build Phase	38
3.3 Ordering Build Phases	42

3.3.1	Build Phase Order.....	43
3.3.2	Build File Phase Order	45
3.4	Setting Custom Build Phase Options	46
3.4.1	Options Tab	46
3.4.2	Output Files Tab	47
3.4.3	Dependent Files Tab.....	49
3.5	File Mappings	51
3.6	Controlling the Build	53
3.7	Logging Build Output.....	54
3.8	Changing Toolchain Version.....	55
3.9	Using an External Debugger.....	56
3.10	Generating a Makefile	58

1. Overview

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

1.1 Workspaces, Projects and Files

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

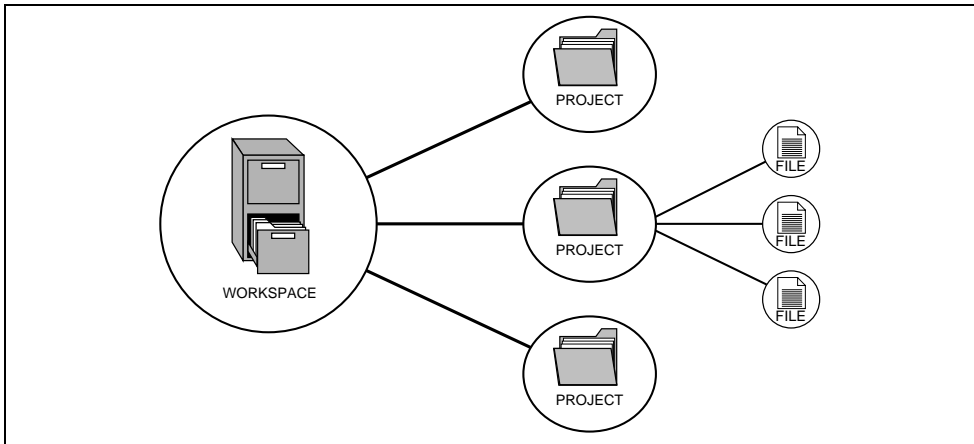


Figure 1.1: Workspaces, Projects and Files

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

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

1.1.1 The Toolbars

The toolbars provide a shortcut to the options, which you will use the most often. There are eight default toolbars: Bookmark, Debug, Debug Run, Editor, Search, Standard, Templates and Version Control. Toolbars can be created, modified and removed via the [Tools->Customize...] menu option (see chapter 5, “Customizing the Environment”, in the Hitachi Embedded Workshop 2.1 User’s Manual, for further information). The builder toolbar is detailed below.

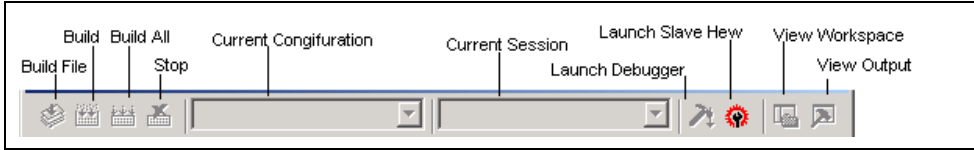


Figure 1.2: Standard Toolbar

1.1.2 The Workspace Window

The “Workspace” window when the HEW is launched only has a single pane. This is the “Projects” tab. If a workspace is opened then the workspace window displays two default tabs. The “Projects” tab shows the current workspace, projects and files (figure 1.3). You can quickly open any project file or dependent file by double clicking on its corresponding icon.

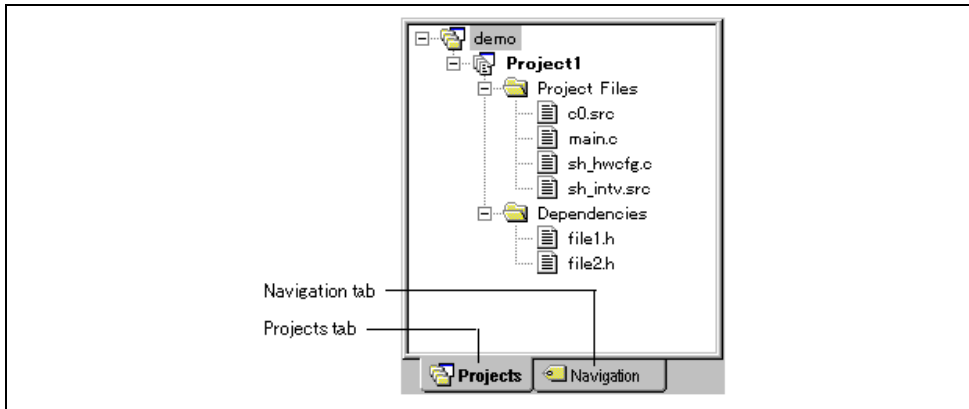


Figure 1.3: Workspace Window Projects Tab

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

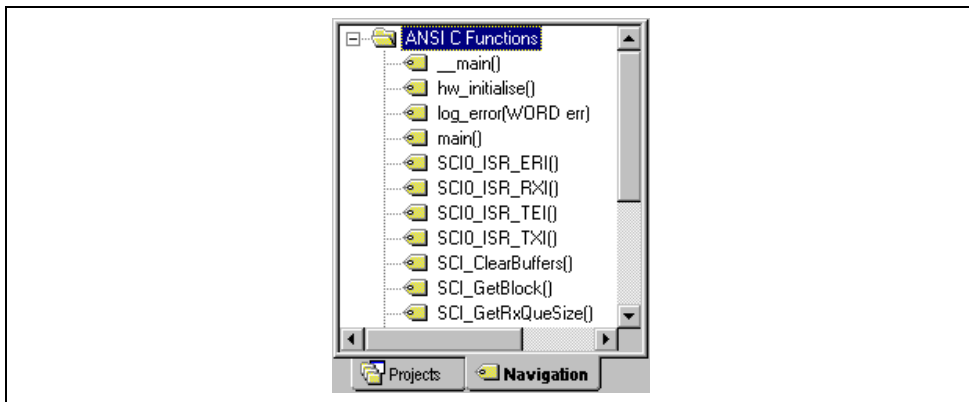


Figure 1.4: Workspace Window Navigation Tab

1.1.3 The Output Window

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

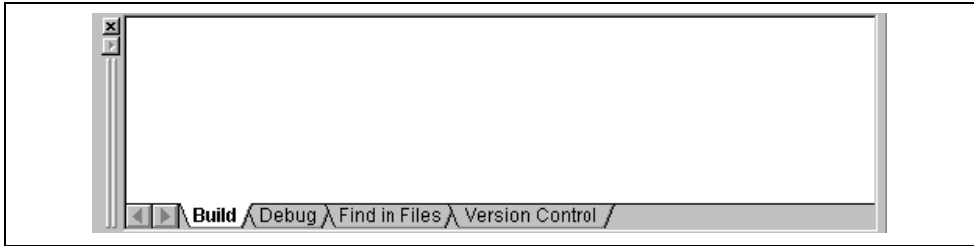


Figure 1.5: Output Window

1.2 Launching the HEW

To run the HEW, open the “Start” menu of Windows®, select “Programs”, select “Hitachi Embedded Workshop 2” and then select the shortcut of the Hitachi Embedded Workshop. By default, the “Welcome!” dialog shown in figure 1.6 will be displayed.

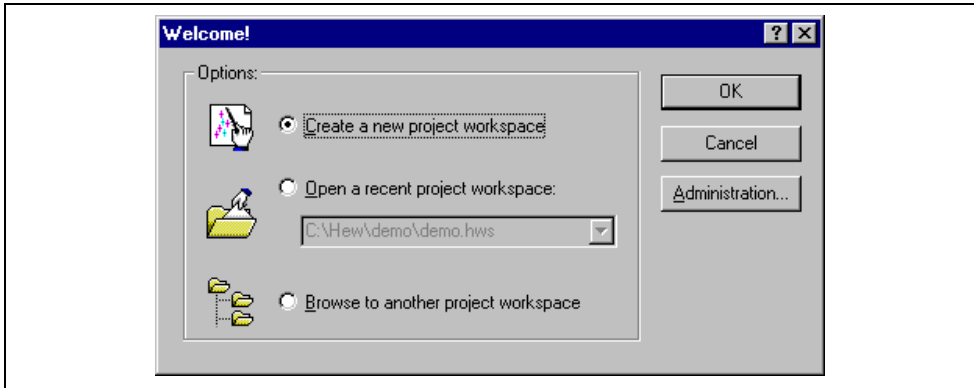


Figure 1.6: Welcome! Dialog

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. Click the “Cancel” button to use the HEW without opening a workspace.

1.3 Creating a New Workspace

➔ To create a new workspace:

1. Select the “Create a new project workspace” option from the “Welcome!” dialog (figure 1.6) and press “OK” or select [**File->New Workspace...**]. The “New Project Workspace” dialog will be displayed (figure 1.7).

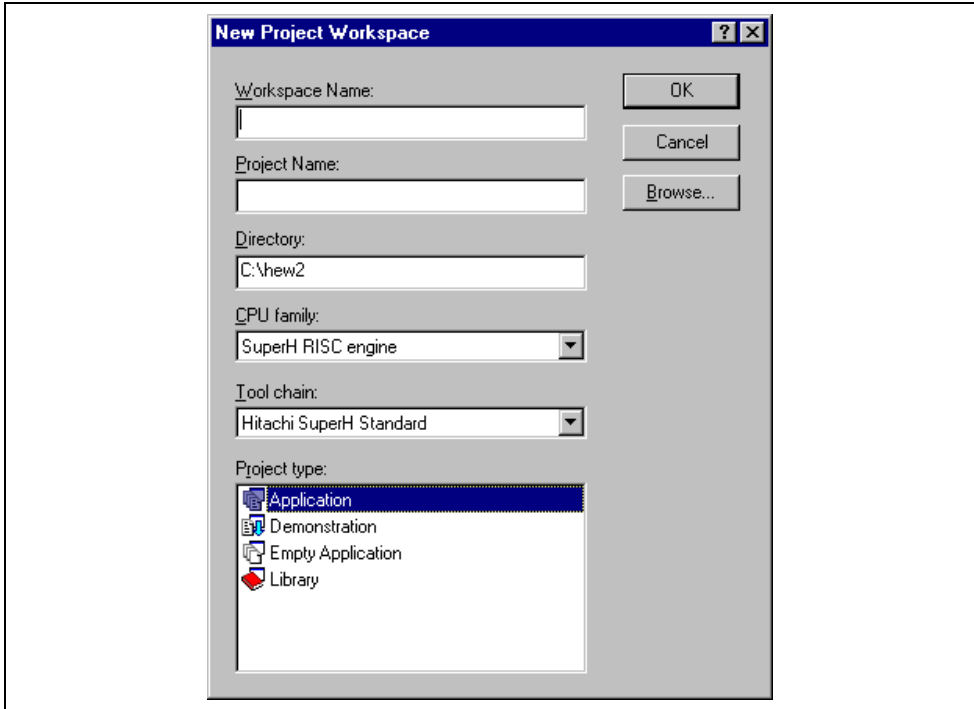


Figure 1.7: New Project Workspace Dialog

2. Enter the name of the new workspace into the “Workspace Name” field. This can be up to 32 characters in length and contain letters, numbers and the underscore character. As you enter the workspace name the HEW will add a subdirectory and project name for you automatically. This can be changed if desired. This allows the workspace and project name to be different. 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 name specified in the project name field and place it inside the new workspace automatically. 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. The project types displayed will be all valid types for the current CPU family and tool chain pair.
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.

1.4 Opening a Workspace

☞ To open a workspace:

1. Select “Browse to another project workspace” option from the “Welcome!” dialog (figure 1.6) 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 1.8). Otherwise, the workspace will be opened.

Note that whether the workspace properties dialog is shown depends on the setting of either the “Show workspace information on workspace open” check box on the workspace properties dialog or the “Display workspace information dialog on opening workspace” check box on the “Workspace” tab of the “Tools Options” dialog. This dialog is described in the “Specifying Workspace Options” section in chapter 5, “*Customizing the Environment*”, in the Hitachi Embedded Workshop 2.1 User’s Manual. Click “OK” to open the workspace. Click “Cancel” to stop opening the workspace.

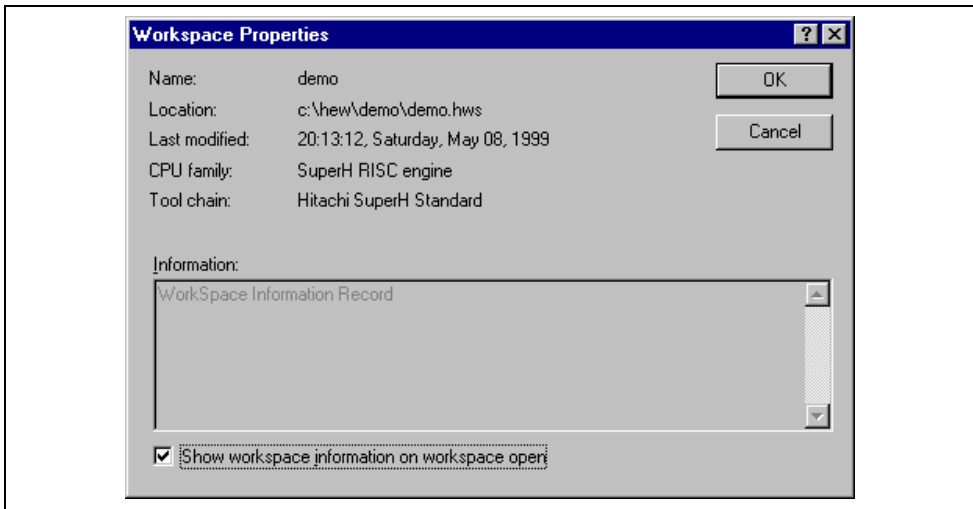


Figure 1.8: Workspace Properties Dialog

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

☞ To open a recently used workspace:

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

1.5 Saving a Workspace

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

1.6 Closing a Workspace

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

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

1.7 Using Old Workspaces

The HEW can open any workspace that was created on a previous version of the HEW. This should not cause any problems and any differences in the workspace file details should be upgraded on the open.

Note: The upgraded project cannot be used in the previous environment. A back-up version of the initial workspace or project file should have been saved if the project is used in the previous environment.

1.8 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.

2. Build Basics

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

2.1 The Build Process

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

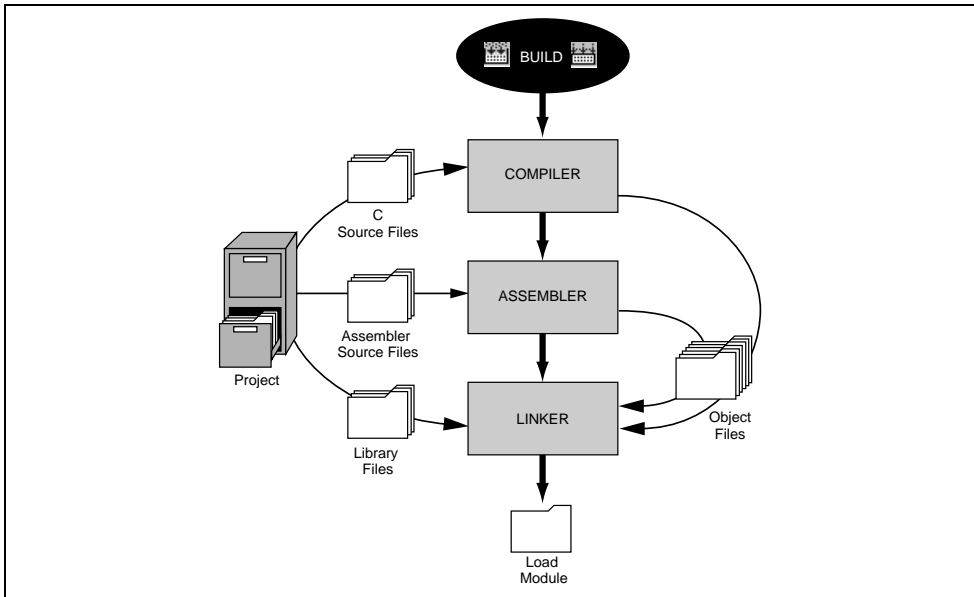


Figure 2.1: Typical Build Process

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

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

2.2 Project Files

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

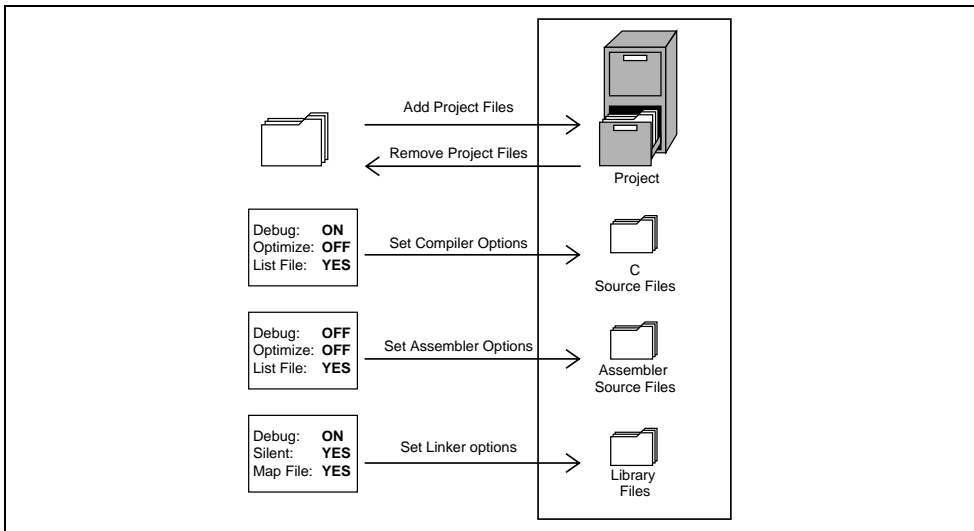


Figure 2.2: Editing a Project

2.2.1 Adding Files to a Project

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

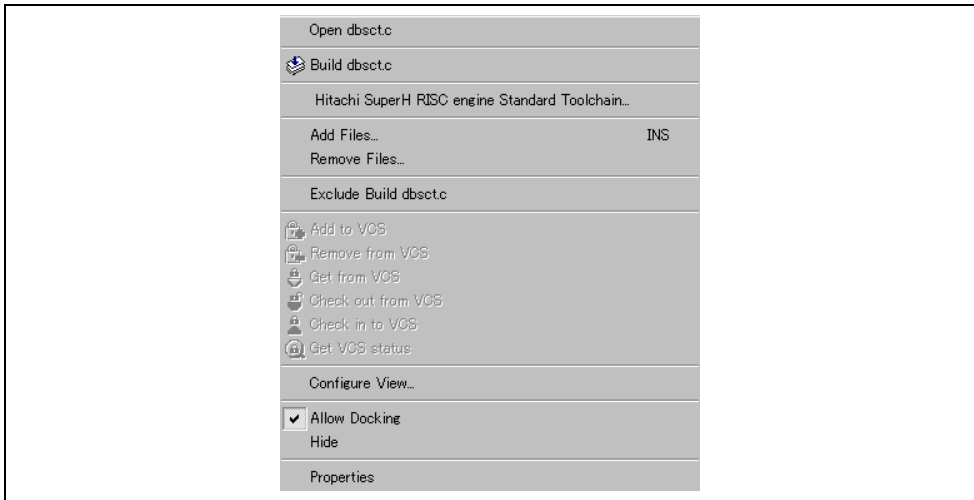


Figure 2.3: Project Pop-up Menu

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

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

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

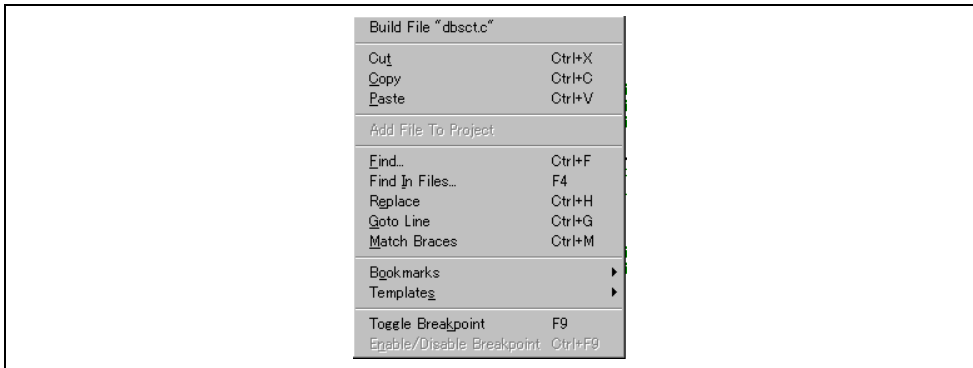


Figure 2.4: Editor Window Pop-up Menu

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

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

2.2.2 Removing Files from a Project

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

☞ To remove files from a project:

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

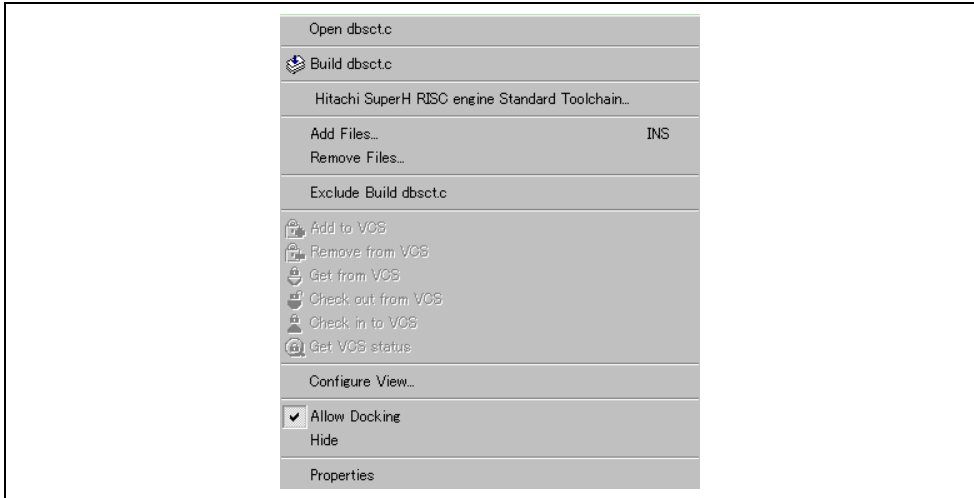


Figure 2.5: Projects Tab Pop-up Menu

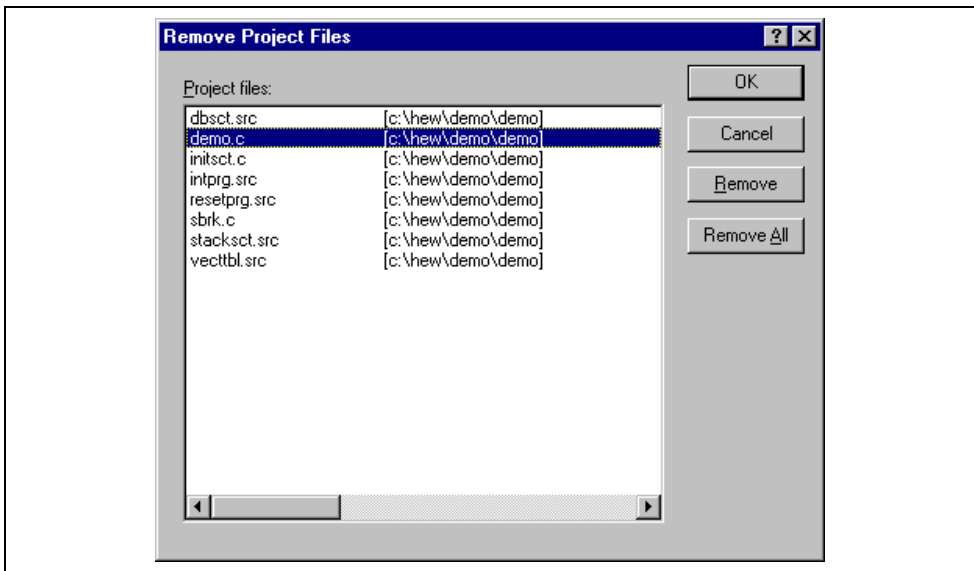


Figure 2.6: Remove Project Files Dialog

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

2.2.3 Excluding a Project File from Build

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

☞ To exclude a file in a project from build:

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

2.2.4 Including a Project File in Build

An excluded file can be included in the project again.

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

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

2.3 File Extensions and File Groups

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

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

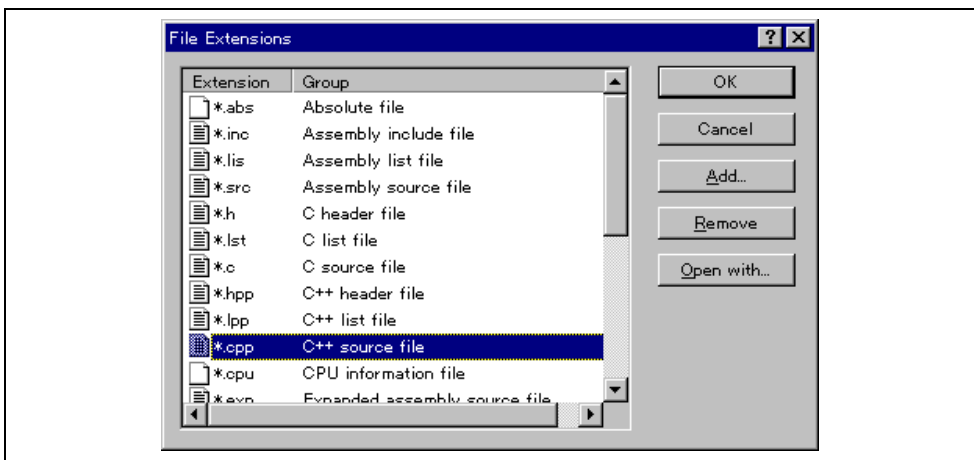


Figure 2.7: File Extensions Dialog

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

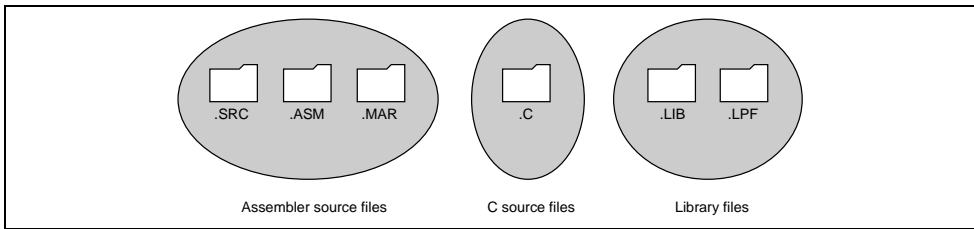


Figure 2.8: File Extensions and Groups

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

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

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

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

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

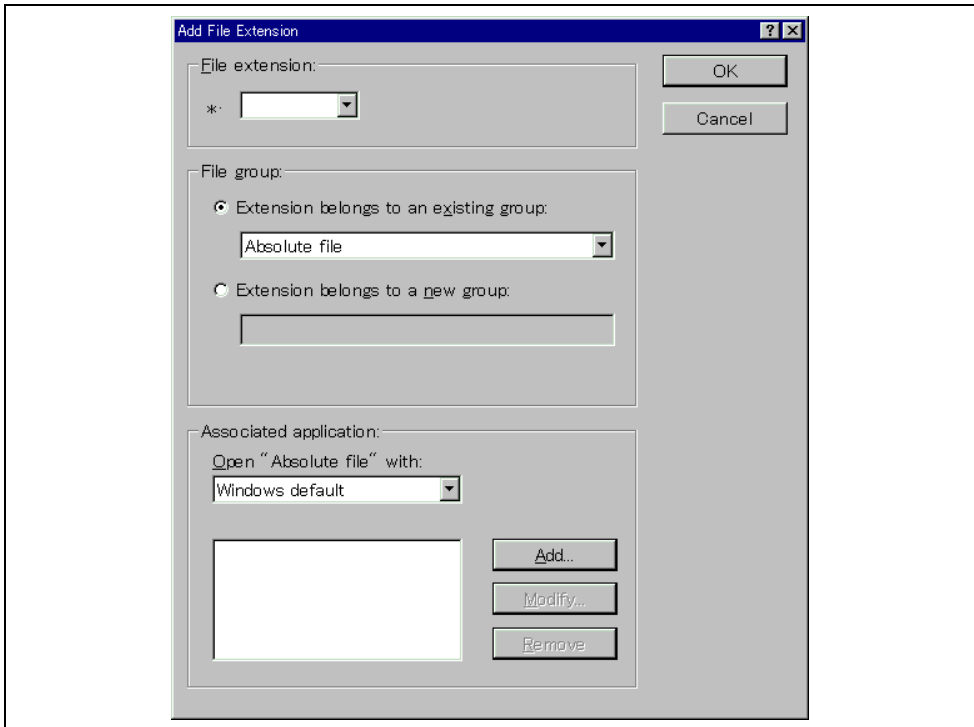


Figure 2.9: Add File Extension Dialog (New Group)

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

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

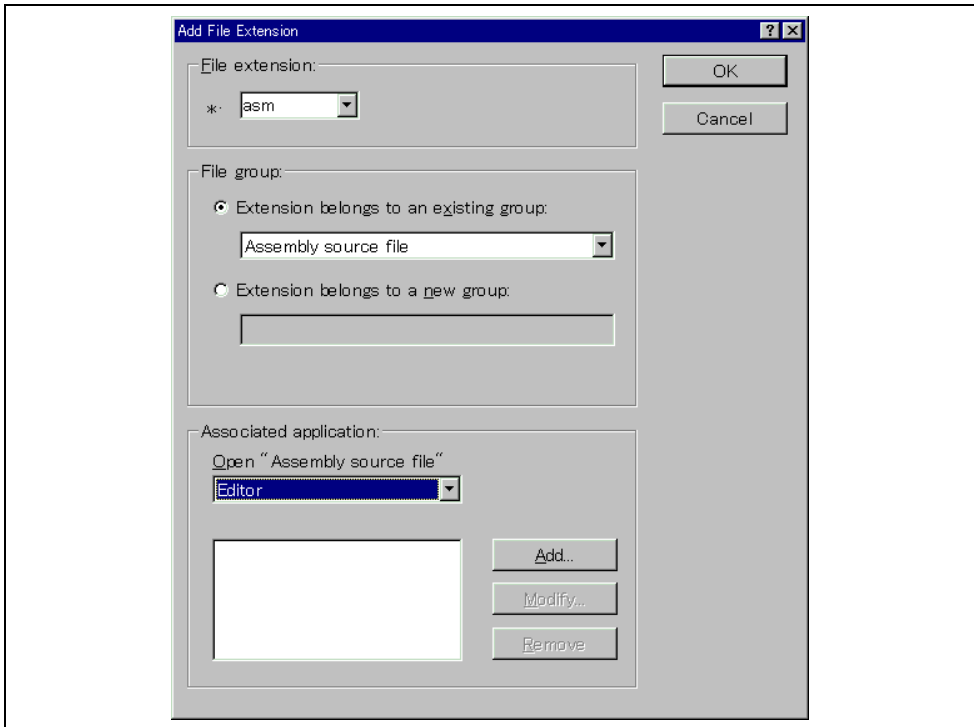


Figure 2.10: Add File Extension Dialog (Existing Group)

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

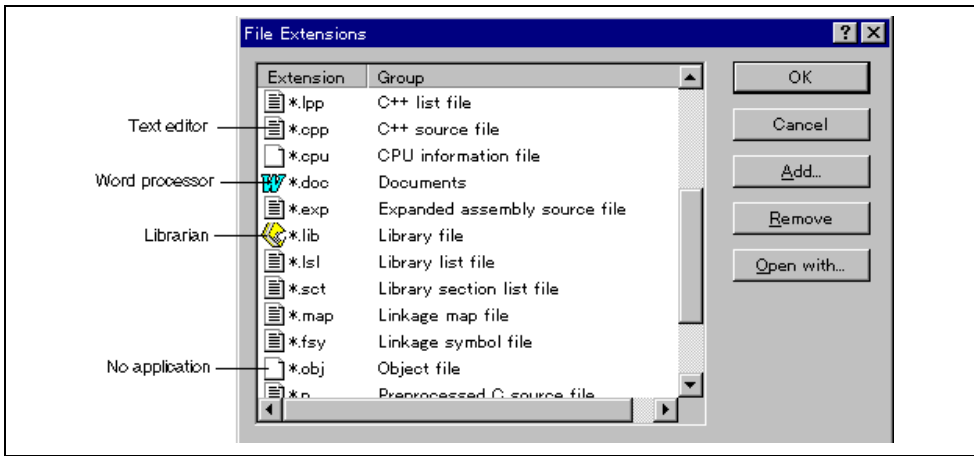


Figure 2.11: File Groups and Applications

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

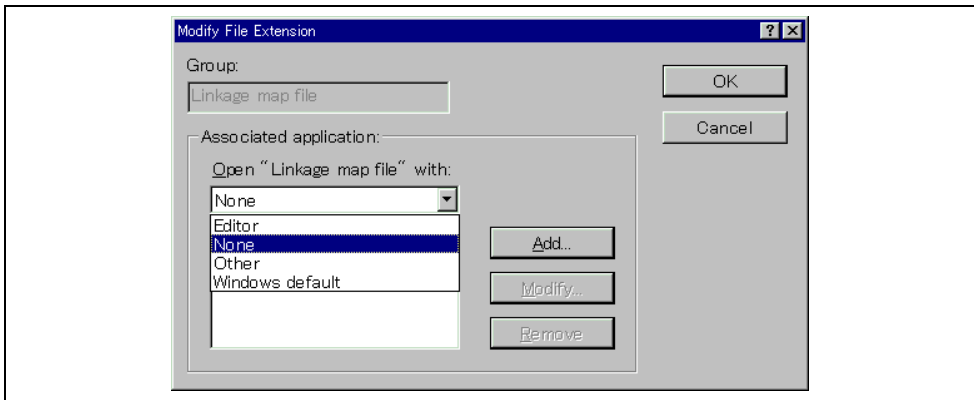


Figure 2.12: Modify File Extension Dialog

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

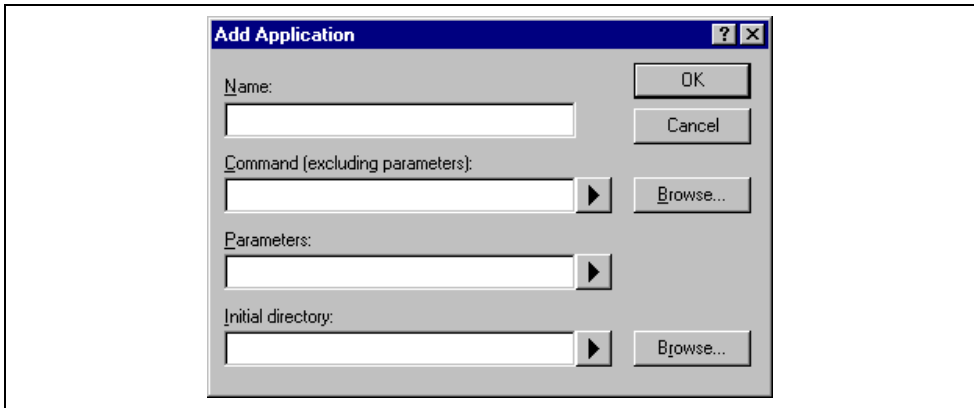


Figure 2.13: Add Application Dialog

5. Enter the name of the tool into the “Name” field. Enter the full path to the tool in the “Command” field (do not include any parameters). Enter the parameters that are required to open a file in the “Parameters” field. Be sure to use the \$(FULLFILE) placeholder to specify the location file (see appendix C, “Placeholders”, in the Hitachi Embedded Workshop 2.1 User’s Manual, for more information on placeholders and their uses). Enter the initial directory, in which you would like the application to run, into the “Initial directory” field. Click “OK” to create the application.
6. Click “Modify...” to modify an application. The “Modify Application” dialog will be displayed. This dialog is the same as the “Add Application” dialog described above except that the “Name” field is read only. Modify the settings as desired and then click “OK”.
7. Click “OK” to set the application for the selected file group.

2.4 Specifying How to Build a File

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

➤ To set options for a build phase:

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

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

2.5 Build Configurations

The HEW allows you to store all of your build options into a build configuration (figure 2.14). This means that you can “freeze” all of the options and give them a name. Later on, you can select that configuration and all of the options for all of the build phases will be restored. These build configurations also allow the user to specify debugger settings for a build configuration. This means that each configuration can be targeted at a different end platform. (See chapter 7, “*Debugging your HEW project*”, in the Hitachi Embedded Workshop 2.1 HEW Debugger User’s Manual, for further information).

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

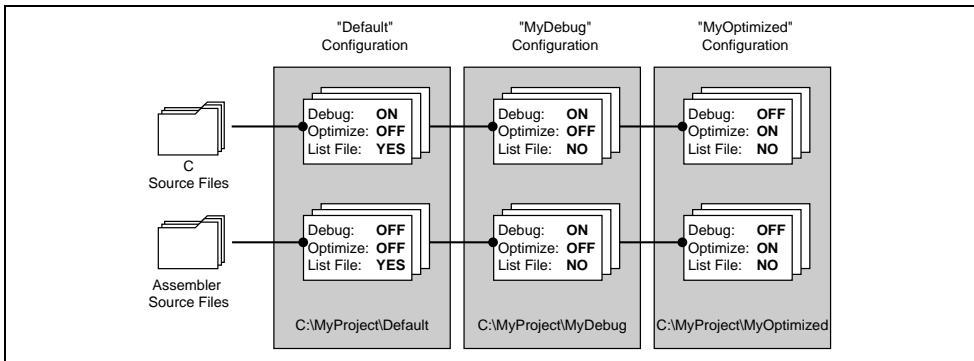


Figure 2.14: Configurations and File Options

2.5.1 Selecting a Configuration

The current configuration can be set in two ways:

Either:

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

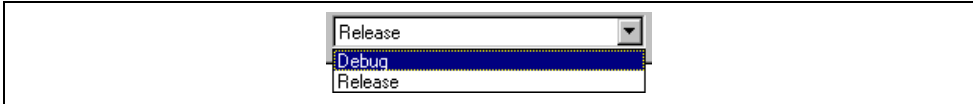


Figure 2.15: Toolbar Selection

or:

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

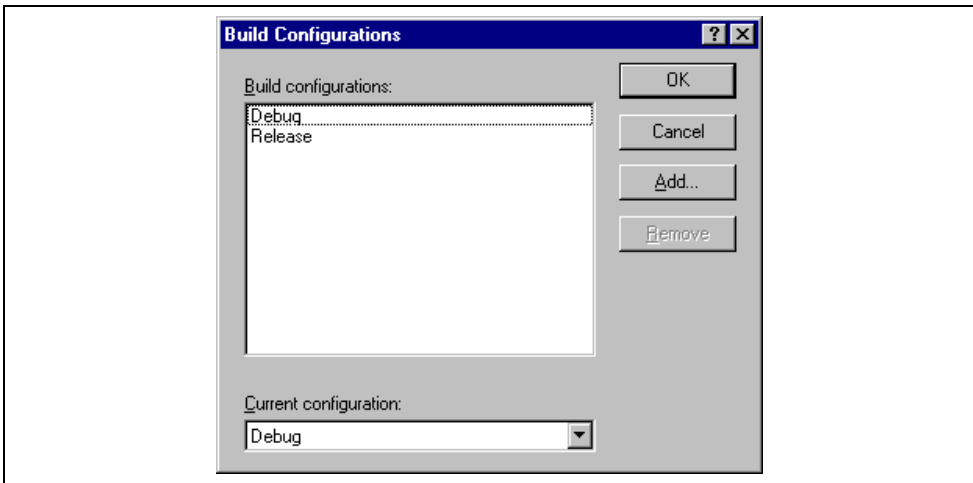


Figure 2.16: Build Configurations Dialog

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

2.5.2 Adding and Deleting Configurations

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

☞ To add a new configuration:

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

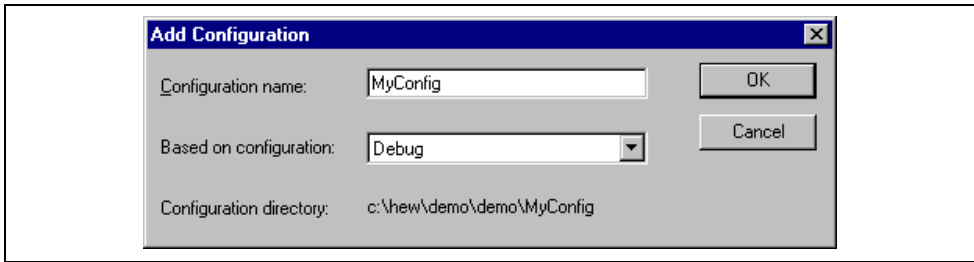


Figure 2.17: Add Configuration Dialog

3. Enter the new configuration name into the “Configuration name” field. As you enter the new configuration name, the directory underneath changes to reflect the configuration directory that will be used. Select one of existing configurations, from which you want to copy a configuration, out of the drop-down list of the “Based on configuration” field. Click “OK” on both dialogs to create the new configuration.
- To remove a configuration:
1. Select [**Options->Build Configurations...**] to display the “Build Configurations” dialog (figure 2.16).
 2. Select the configuration that you want to remove and then click the “Remove” button.
 3. Click “OK” to close the “Build Configurations” dialog.


2.6 Building a Project

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

2.6.1 Building a Project


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

☞ To perform a build:

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

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

☞ To perform a build all:


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

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

2.6.2 Building Individual Files


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

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

2.6.4 Building Multiple Projects

The Hitachi Embedded Workshop lets you build multiple projects and configurations at once.

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

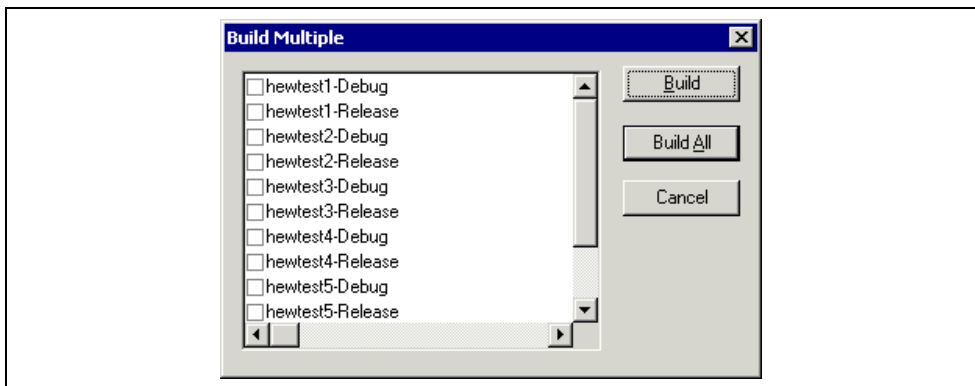


Figure 2.18: Build Multiple Dialog

2.6.5 The Output Window

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

2.6.6 Controlling the Content of the Output Window

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

☛ To view or hide extra information during a build:

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

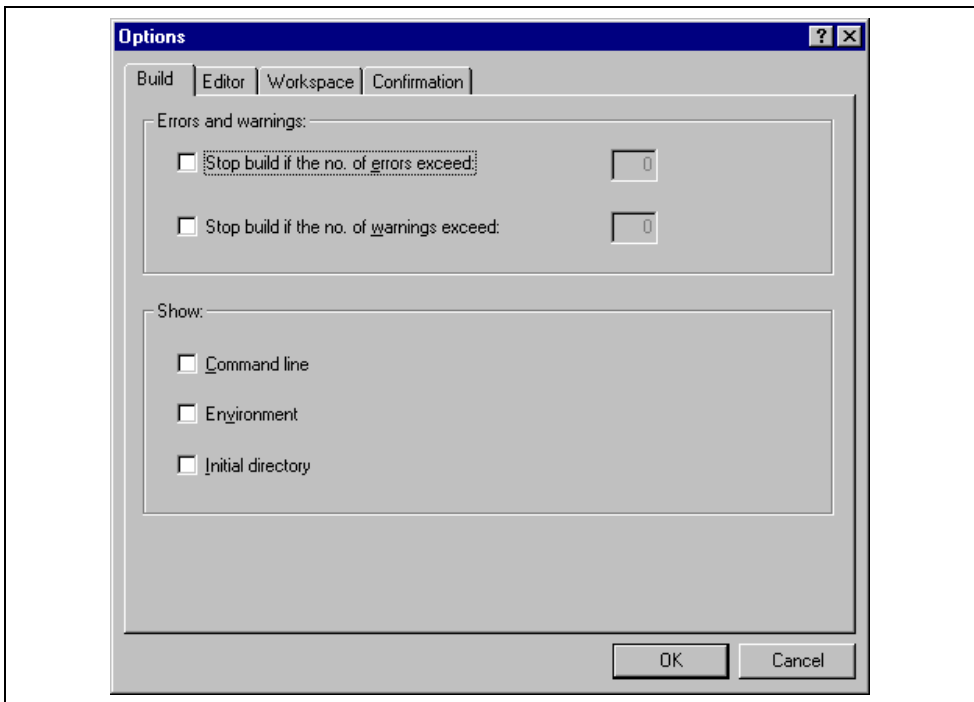


Figure 2.19: Options Dialog Build Tab

2.7 File Dependencies

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

☞ To update a project’s dependencies:

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

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

2.8 Configuring the Workspace Window

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

2.8.1 Show Dependencies under Each File

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

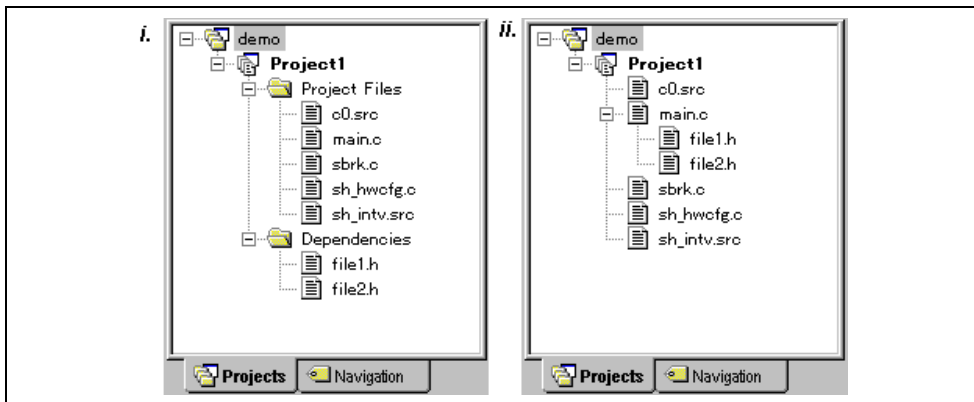


Figure 2.20: Dependencies under Each File

2.8.2 Show Standard Library Includes

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

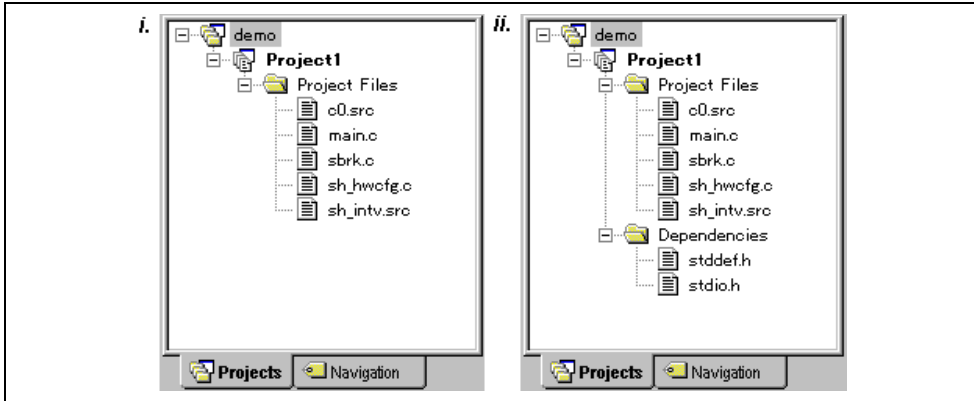


Figure 2.21: Standard Library Includes

2.8.3 Show File Paths

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

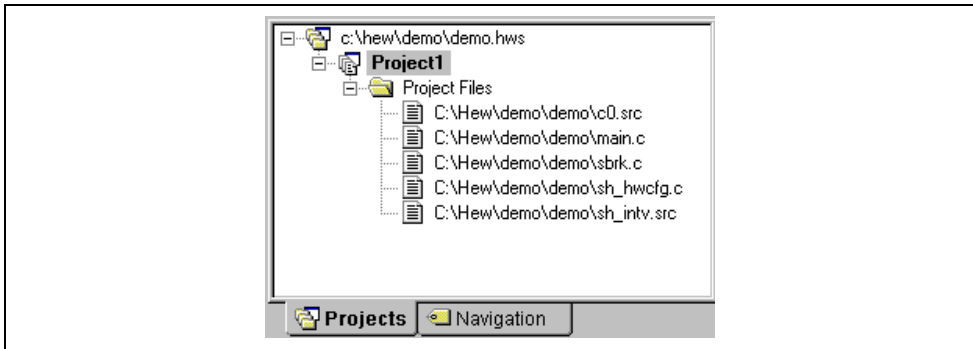


Figure 2.22: File Paths Shown

2.9 Setting the Current Project

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

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

2.10 Inserting a Project into a Workspace

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

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

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

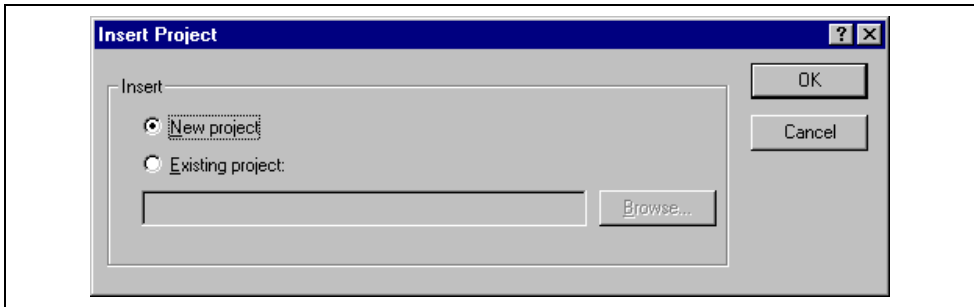


Figure 2.23: [Insert Project] Dialog

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

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

2.11 Specifying Dependencies between Projects

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

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

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

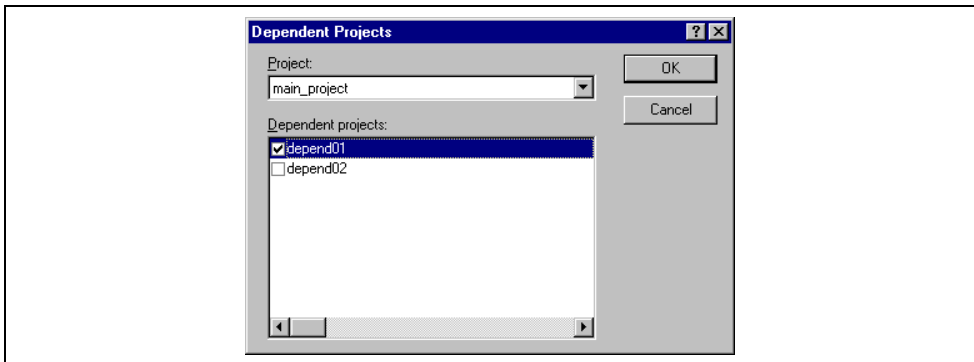


Figure 2.24 Dependent Projects dialog

2.12 Removing a Project from a Workspace

➡ To remove a project from a workspace:

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

or:

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

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

2.13 Relative projects paths in the workspace

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

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

☞ To change a projects relative path flag:

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

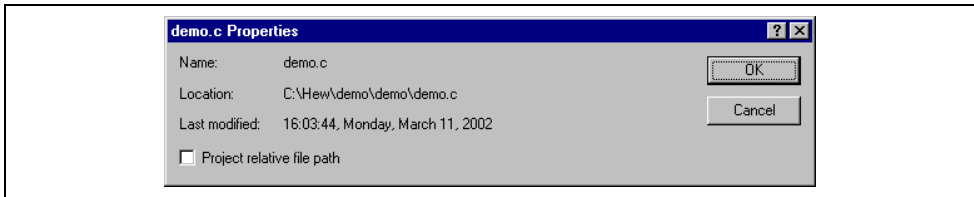


Figure 2.25: Properties Dialog

3. Advanced Build Features

This chapter explains the more advanced build concepts.

3.1 The Build Process Revisited

Chapter 2, “*Build Basics*” began by describing the build process in terms of a compiler, an assembler and a linker (figure 2.1). This will be the case for most installations of the 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.

3.1.1 What is a Build?

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

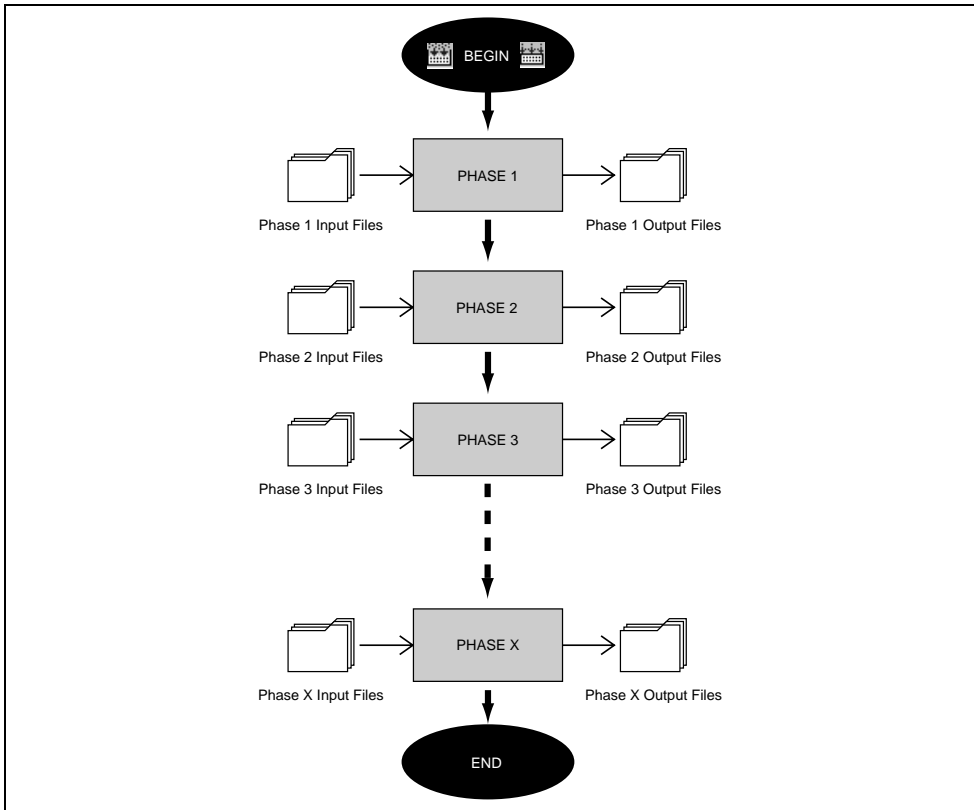


Figure 3.1: Build Process

The 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 3.2). On the left-hand side are the phases that are defined in the current project (Figure 3.2 shows a standard set of build phases). The remainder of this chapter details the various functions that the “Build Phases” dialog provides.

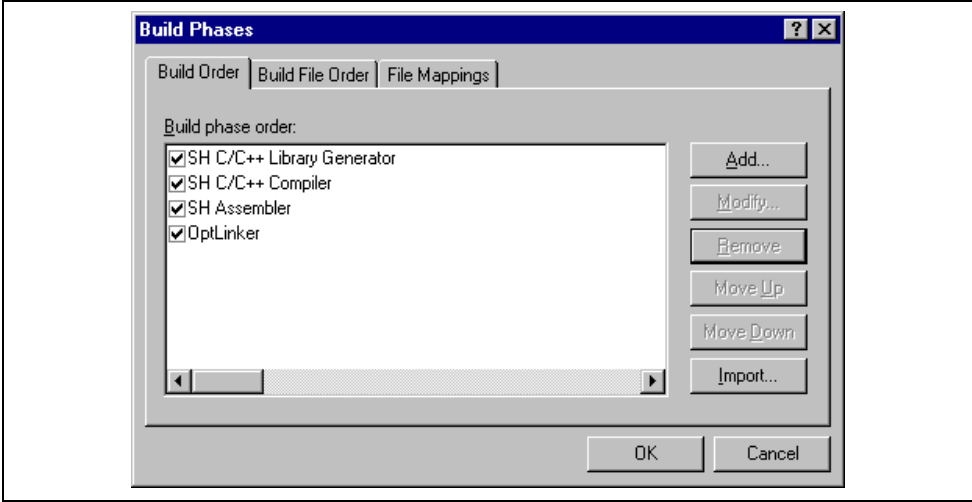


Figure 3.2: Build Phases Dialog

3.2 Creating a Custom Build Phase

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

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

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

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

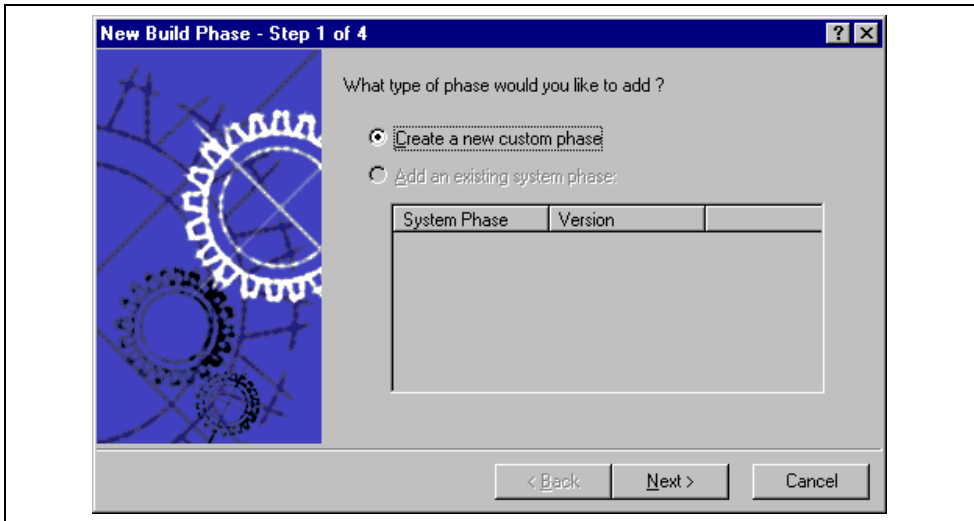


Figure 3.3a: New Build Phase Dialog (Step 1)

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

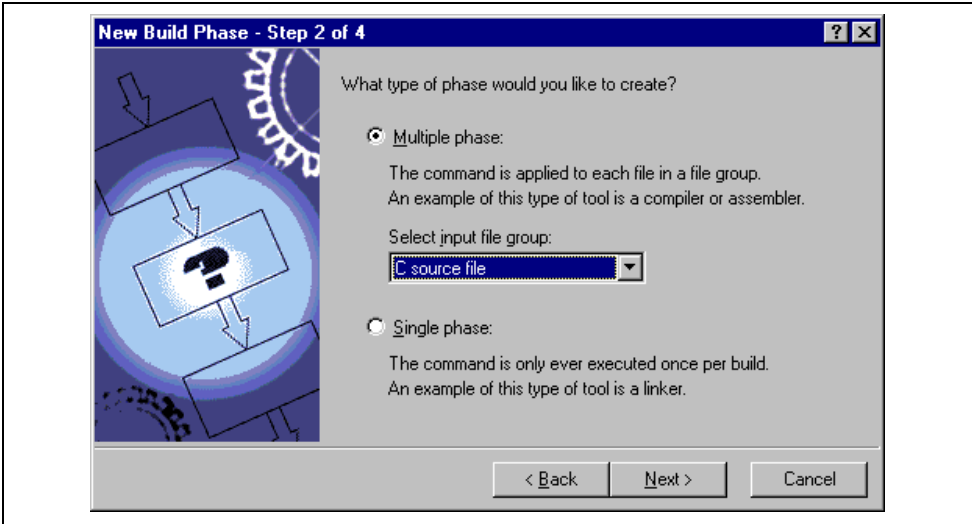


Figure 3.3b: New Build Phase Dialog (Step 2)

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

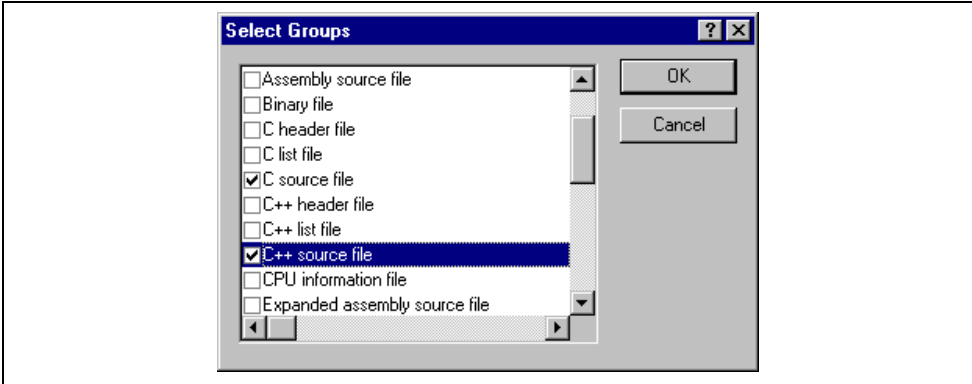


Figure 3.3c: Modify multiple input file groups

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

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

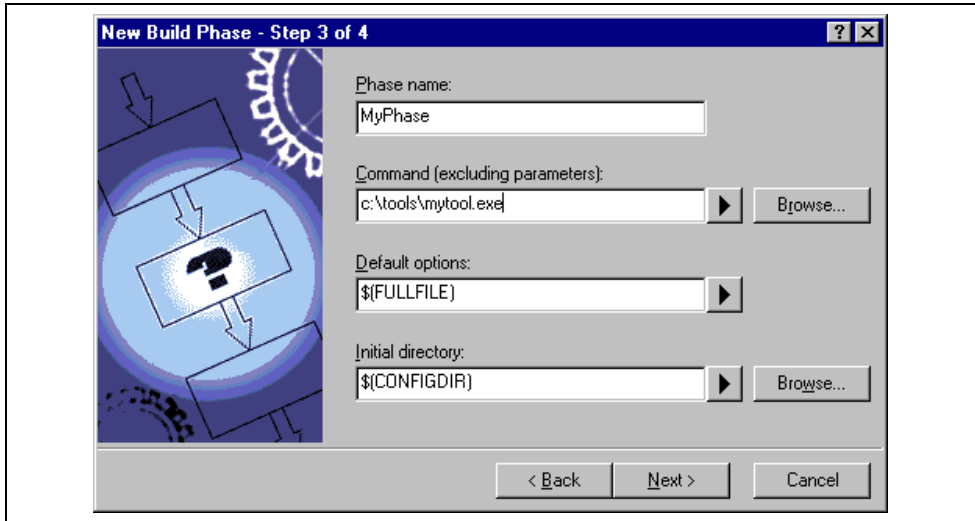


Figure 3.3d: New Build Phase Dialog (Step 3)

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

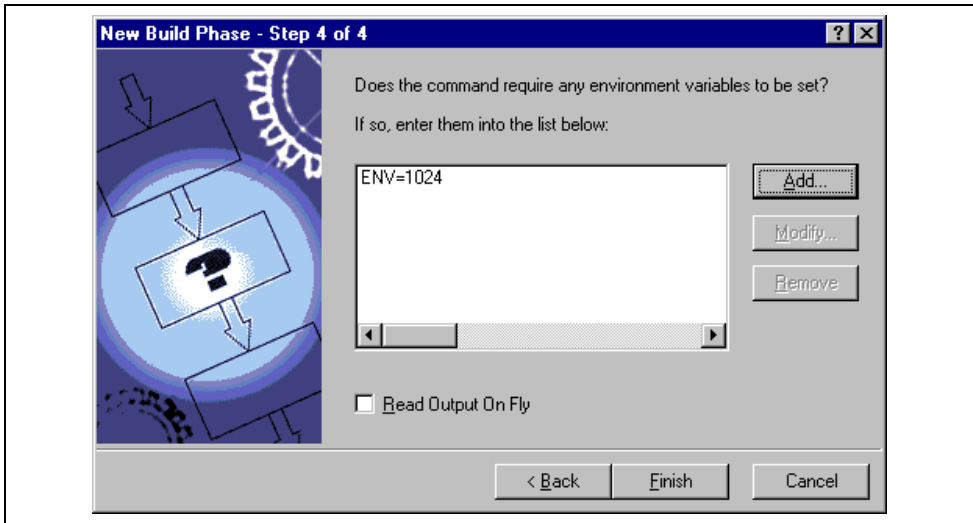


Figure 3.3e: New Build Phase Dialog (Step 4)

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

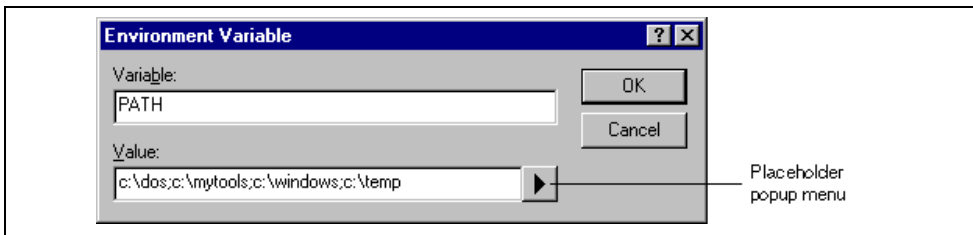


Figure 3.4: Environment Variable Dialog

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

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

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

3.3 Ordering Build Phases

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

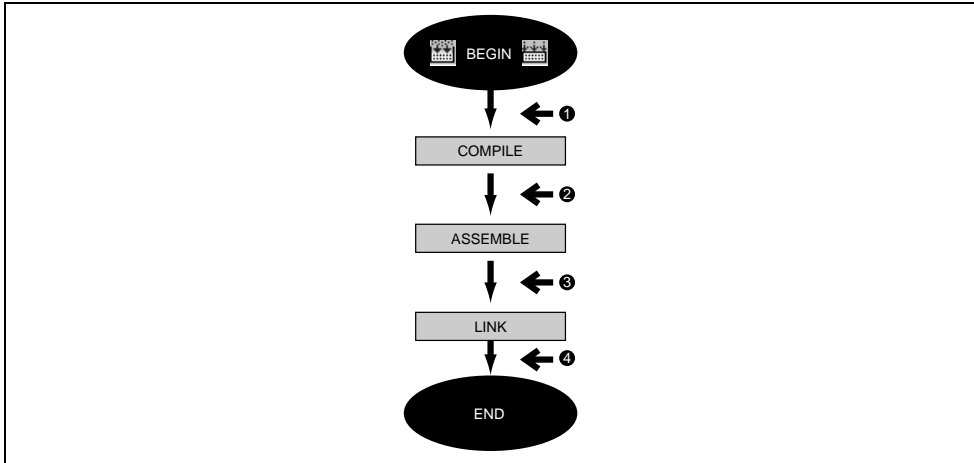


Figure 3.5: Typical Build Process

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

3.3.1 Build Phase Order

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

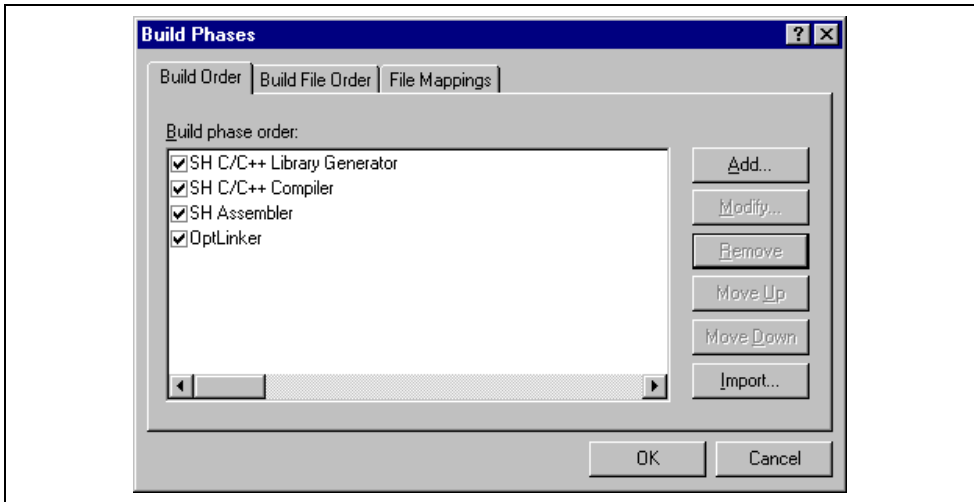


Figure 3.6: Build Phases Dialog Build Order Tab

In addition the following operations can be performed:

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

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

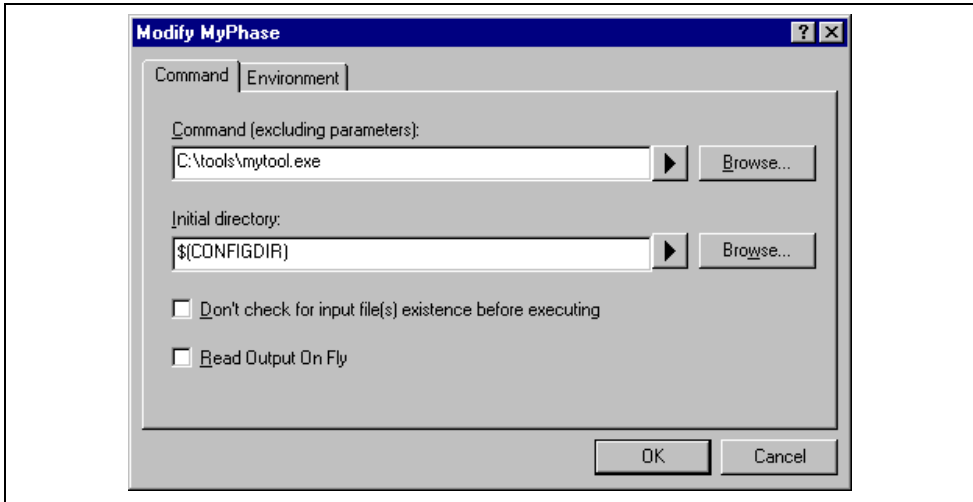


Figure 3.7: Modify Phase Dialog Command Tab

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

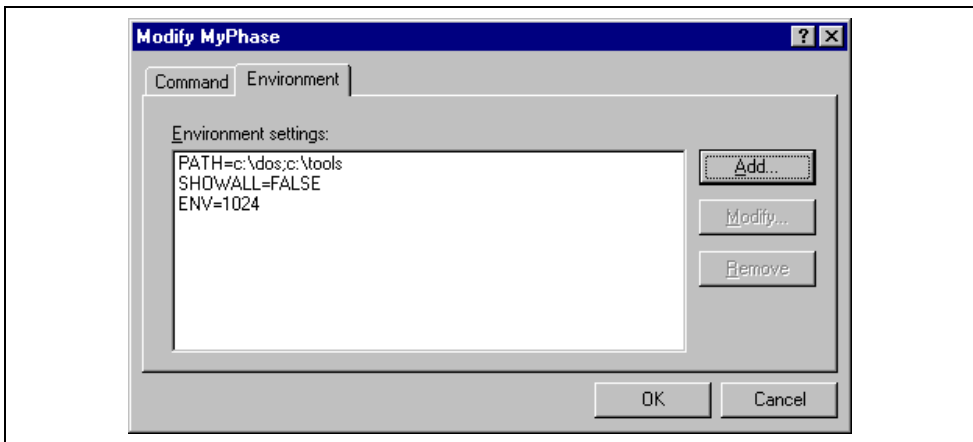



Figure 3.8: Modify Phase Dialog Environment Tab

3.3.2 Build File Phase Order

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

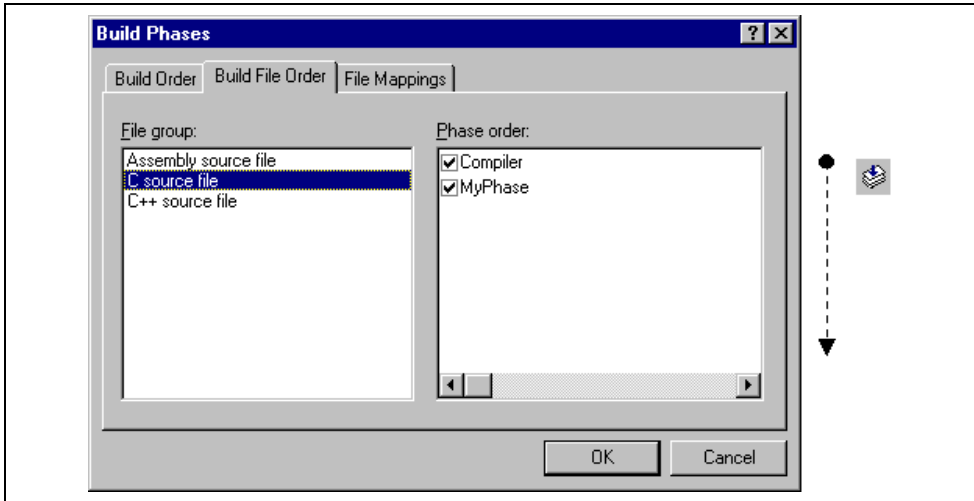


Figure 3.9: Build Phases Dialog Build File Order Tab

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

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

3.4 Setting Custom Build Phase Options

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

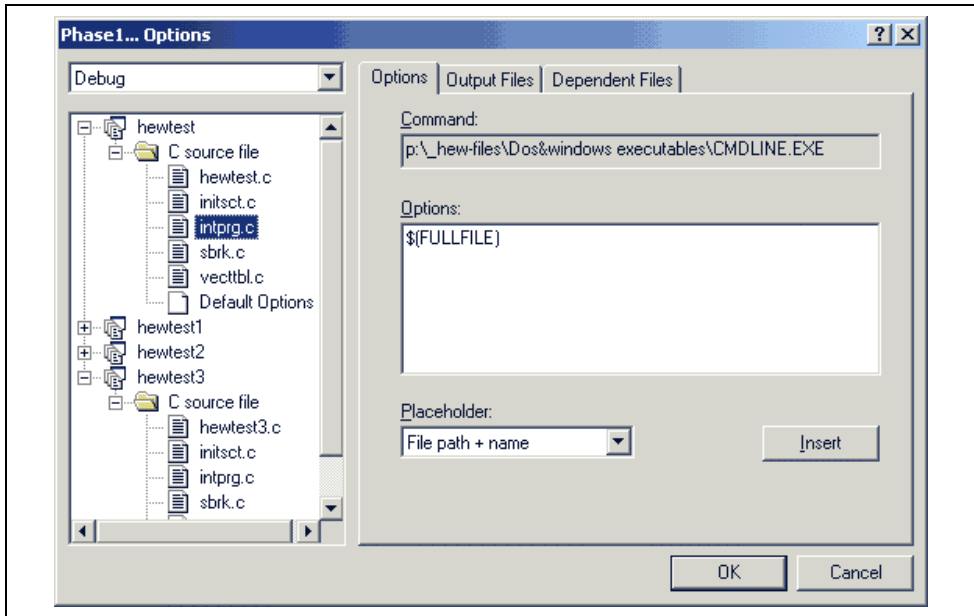


Figure 3.10: Custom Options Dialog

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

3.4.1 Options Tab

The “Options” tab (figure 3.11) allows you to define the command line options that will be passed to the phase. The “Command” field displays the command, which was entered when you defined the phase (figure 3.3d). Enter into the “Options” field the command line arguments that you would like to pass to the command. If you want to insert a placeholder, select the relevant placeholder from the “Placeholder” drop-down list box and then click the “Insert” button. For a detailed description of placeholders see appendix C, “Placeholders” in the Hitachi Embedded Workshop 2.1 User’s Manual.

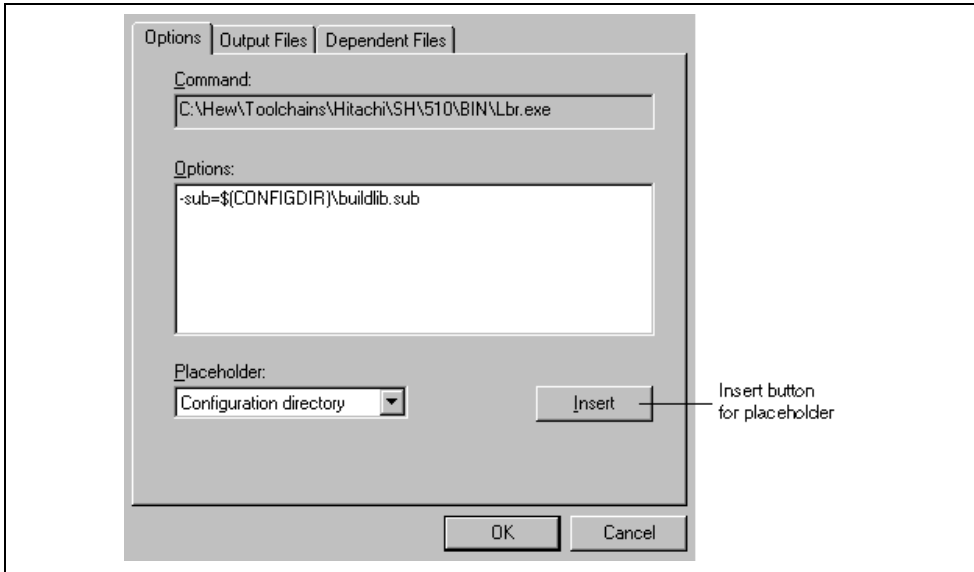


Figure 3.11: Custom Options Options Tab

3.4.2 Output Files Tab

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

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

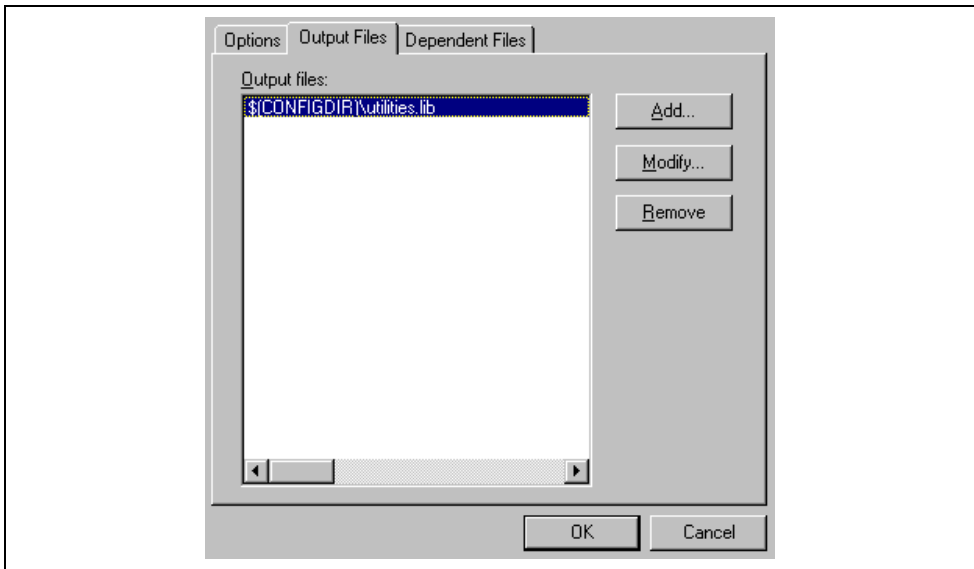


Figure 3.12: Custom Options Output Files Tab

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

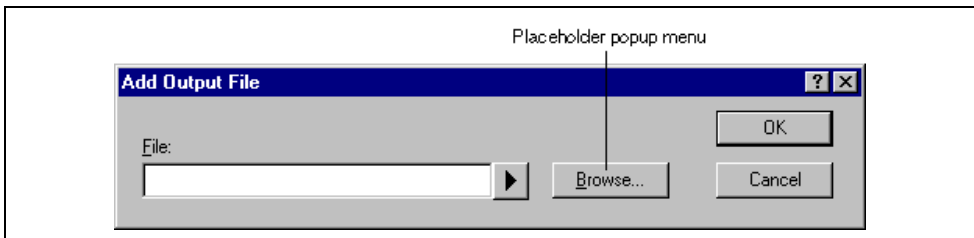


Figure 3.13: Add Output File Dialog

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

3.4.3 Dependent Files Tab

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

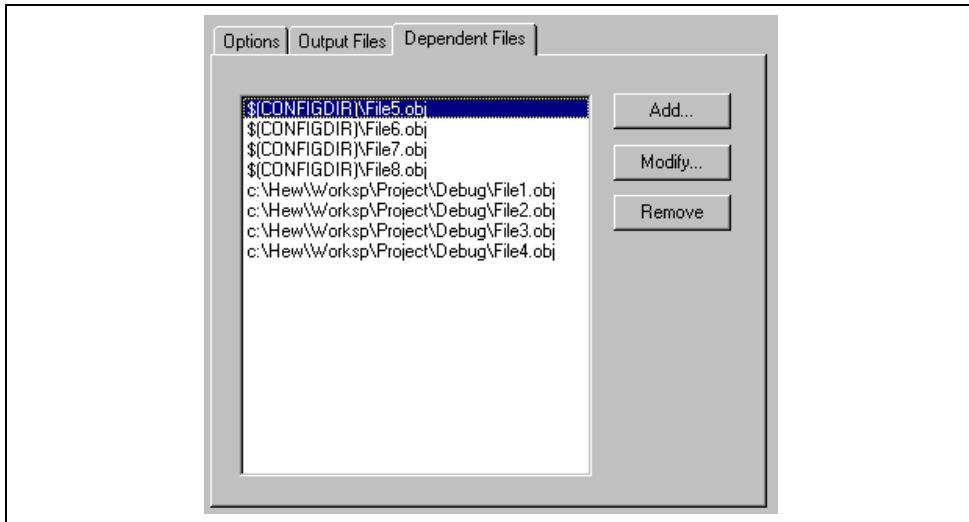


Figure 3.14: Dependent Files Tab in Custom Options

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

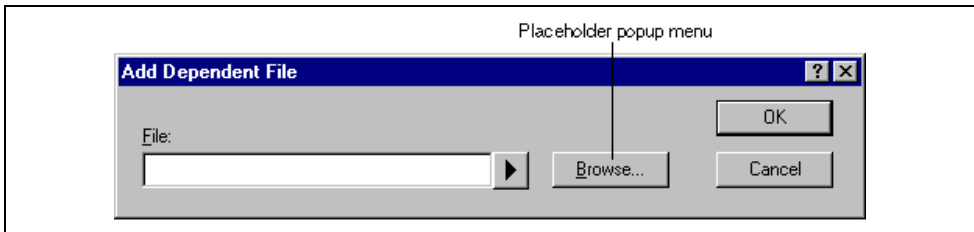


Figure 3.15: Add Dependent File Dialog

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

3.5 File Mappings

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

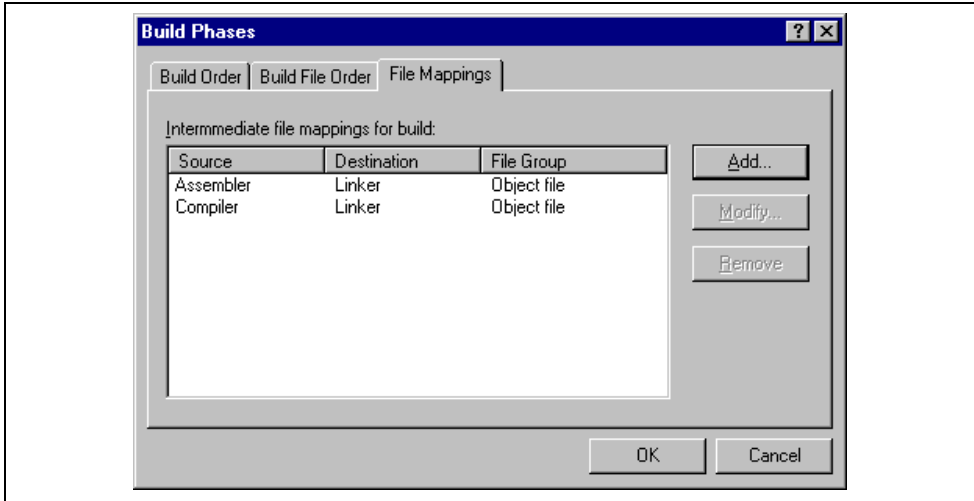


Figure 3.16: Build Phases Dialog File Mappings Tab

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

➡ To add a file mapping:

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

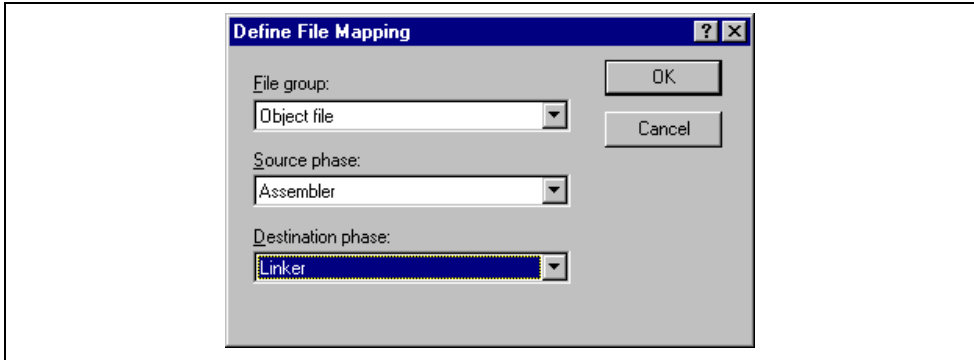


Figure 3.17: Define File Mapping Dialog

➡ To modify a file mapping:

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

3.6 Controlling the Build

By default, the 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 behavior by setting the controls on the “Build” tab of the “Options” dialog (figure 3.18).

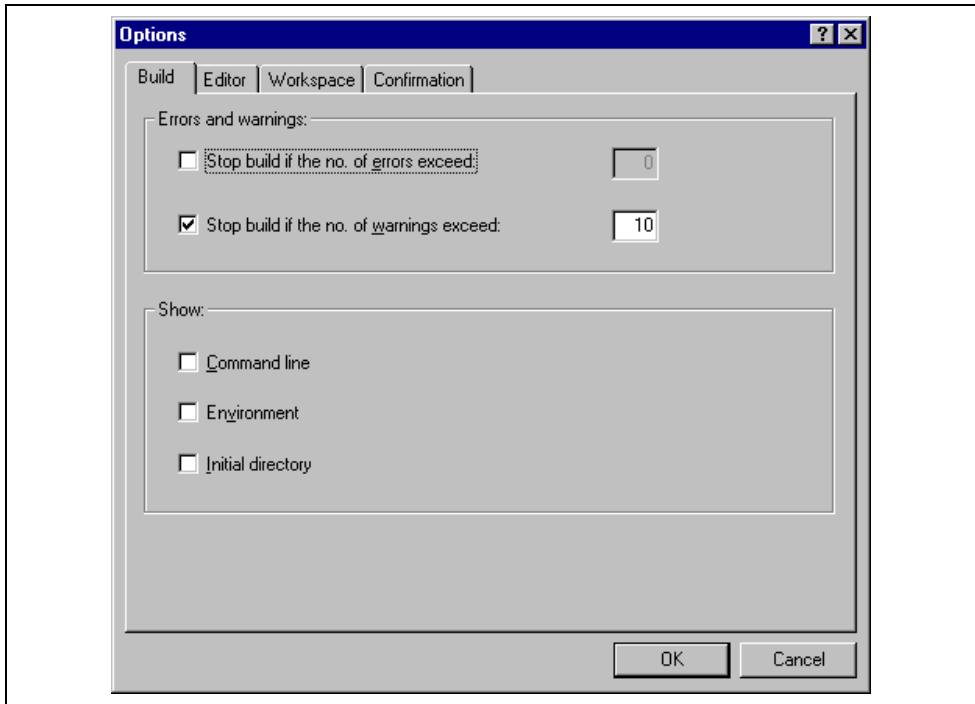


Figure 3.18: Options Dialog Build Tab

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

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

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

3.7 Logging Build Output

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

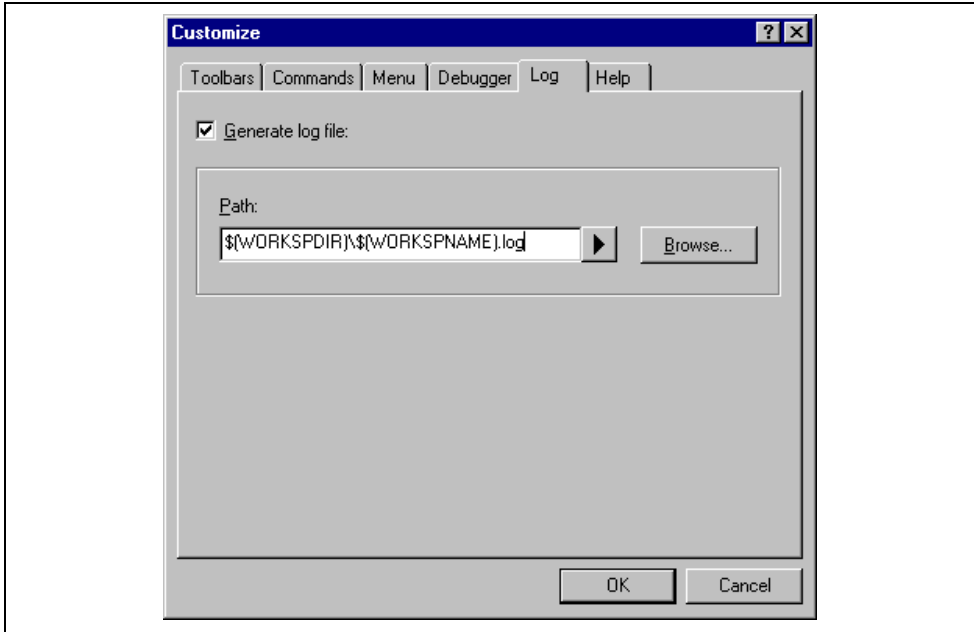


Figure 3.19: Tools Customize Dialog Log Tab

3.8 Changing Toolchain Version

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

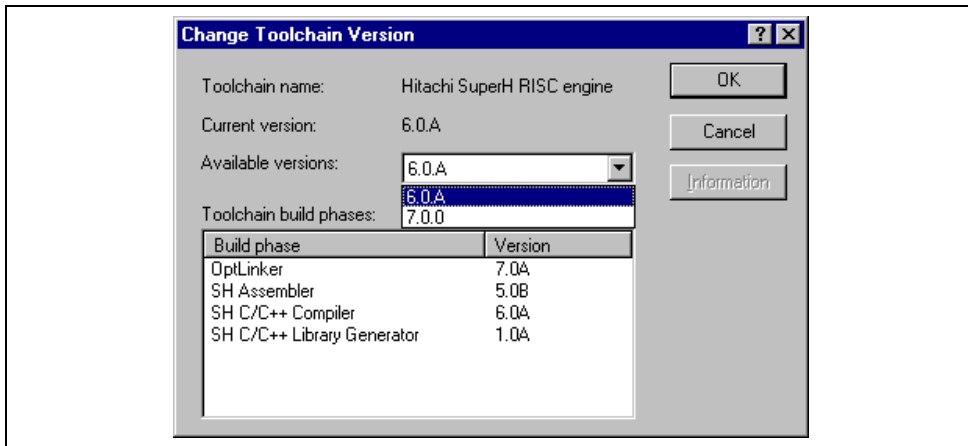


Figure 3.20: Change Toolchain Version Dialog

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

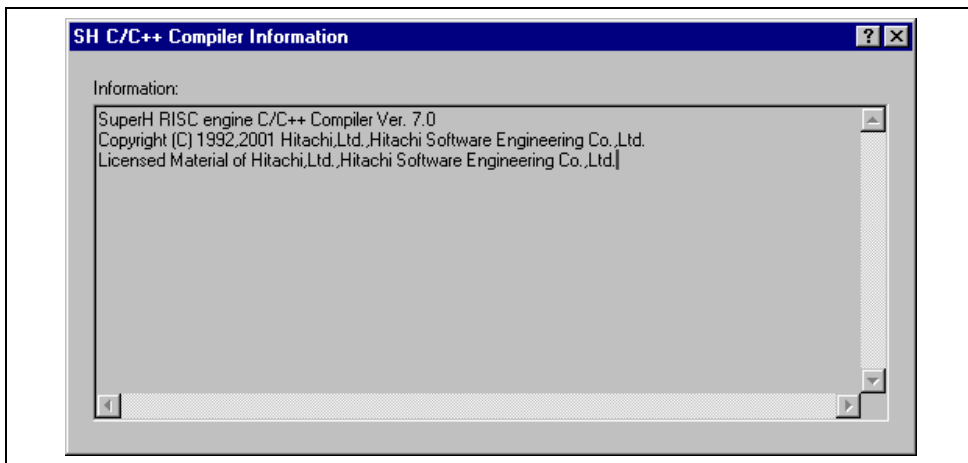


Figure 3.21: Toolchain Information Dialog

3.9 Using an External Debugger

The Hitachi Embedded Workshop can launch an external debugger tool. If you want to use another debugger then you must add it to the Tools menu (as described in chapter 5, “*Customizing the Environment*”, in the Hitachi Embedded Workshop 2.1 User’s Manual).

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

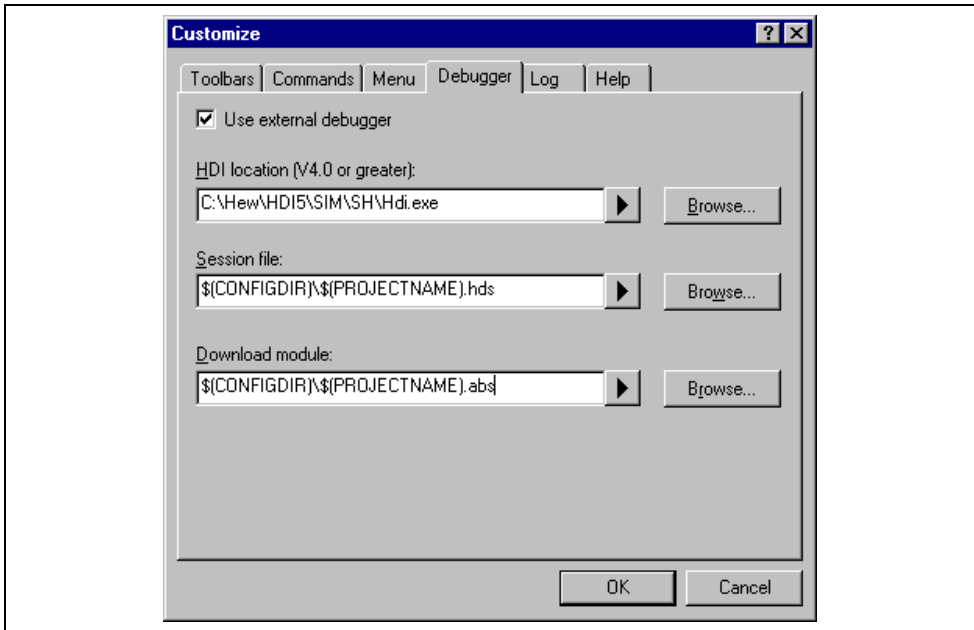


Figure 3.22: Customize Dialog Debugger Tab

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

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



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

3.10 Generating a Makefile

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

➤ To generate a makefile:

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

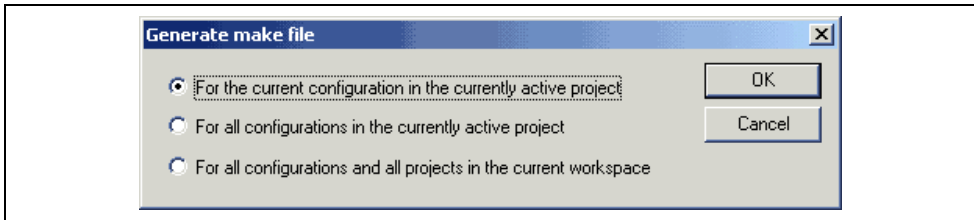


Figure 3.23: Generate makefile Dialog

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

➤ To execute a makefile:

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

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