ARM® DS-5

Version 5

Eclipse for DS-5 User Guide

ARM

ARM DUI0480U

ARM® DS-5

Eclipse for DS-5 User Guide

Release Information

Document History

Issue	Date	Confidentiality	Change
A	June 2010	Non-Confidential	First release
B	September 2010	Non-Confidential	Update for DS-5 version 5.2
C	November 2010	Non-Confidential	Update for DS-5 version 5.3
D	January 2011	Non-Confidential	Update for DS-5 version 5.4
E	May 2011	Non-Confidential	Update for DS-5 version 5.5
F	July 2011	Non-Confidential	Update for DS-5 version 5.6
G	September 2011	Non-Confidential	Update for DS-5 version 5.7
H	November 2011	Non-Confidential	Update for DS-5 version 5.8
I	February 2012	Non-Confidential	Update for DS-5 version 5.9
J	May 2012	Non-Confidential	Update for DS-5 version 5.10
K	July 2012	Non-Confidential	Update for DS-5 version 5.11
L	October 2012	Non-Confidential	Update for DS-5 version 5.12
M	December 2012	Non-Confidential	Update for DS-5 version 5.13
N	March 2013	Non-Confidential	Update for DS-5 version 5.14
O	March 2014	Non-Confidential	Update for DS-5 version 5.18
P	31 March 2015	Non-Confidential	Update for DS-5 version 5.21
Q	15 July 2015	Non-Confidential	Update for DS-5 version 5.22
R	15 October 2015	Non-Confidential	Update for DS-5 version 5.23
S	15 March 2016	Non-Confidential	Update for DS-5 version 5.24
T	15 July 2016	Non-Confidential	Update for DS-5 version 5.25
U	18 November 2016	Non-Confidential	Update for DS-5 version 5.26

Non-Confidential Proprietary Notice

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Unrestricted Access is an ARM internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com

Contents
ARM® DS-5 Eclipse for DS-5 User Guide

List of Figures
ARM® DS-5 Eclipse for DS-5 User Guide

List of Tables
ARM® DS-5 Eclipse for DS-5 User Guide

Preface

This preface introduces the *ARM® DS-5 Eclipse for DS-5 User Guide*.

It contains the following:

- *About this book* on page 10.

About this book

Eclipse for DS-5 Using Eclipse. This book introduces the ARM plug-ins for use with Eclipse, and describes how you can use them with other tools from Development Studio 5. It describes how to build, debug, monitor, and manage projects for ARM targets. It is also available as a PDF.

Using this book

This book is organized into the following chapters:

Chapter 1 Getting started with Eclipse

The following topics describe how to get started with Eclipse, the C/C++ perspective, the DS-5 Debug perspective, and associated features.

Chapter 2 Working with projects

You can use Eclipse to create projects for ARM targets. Projects are top level folders in your workspace that contain related files and sub-folders. A project must exist in your workspace before a new file can be added or an existing file can be imported.

Chapter 3 Working with editors

The following topics describe how to use the editors when developing a project for an ARM target.

Chapter 4 Terminology, shortcuts and icons

This following topics describe some of the terminology used in Eclipse for DS-5 User Guide, useful keyboard shortcuts, and menu and toolbar icons.

Glossary

The ARM Glossary is a list of terms used in ARM documentation, together with definitions for those terms. The ARM Glossary does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

See the *ARM Glossary* for more information.

Typographic conventions

italic
: Introduces special terminology, denotes cross-references, and citations.

bold
: Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`
: Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`mono`space
: Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`
: Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`
: Denotes language keywords when used outside example code.

`<and>`
: Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
: Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *ARM® DS-5 Eclipse for DS-5 User Guide*.
- The number ARM DUI0480U.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Other information

- *ARM Information Center*.
- *ARM Technical Support Knowledge Articles*.
- *Support and Maintenance*.
- *ARM Glossary*.

Chapter 1
Getting started with Eclipse

The following topics describe how to get started with Eclipse, the C/C++ perspective, the DS-5 Debug perspective, and associated features.

It contains the following sections:

1.1 About Eclipse

Eclipse for DS-5 is an *Integrated Development Environment* (IDE) that combines the Eclipse IDE from the Eclipse Foundation with the compilation and debug technology of the ARM tools. It also combines the GNU toolchain for ARM Linux targets.

You can use Eclipse for DS-5 as a project manager to create, build, debug, monitor, and manage projects for ARM targets. It uses a single folder called a workspace to store files and folders related to specific projects.

Related information

Eclipse Foundation.

GCC, the GNU Compiler Collection.

1.2 Launching Eclipse

How you launch Eclipse depends on your platform.

To launch Eclipse:

- On Windows, select **Start** > **All Programs** > **ARM DS-5** > **Eclipse for DS-5**.
- On Linux:
  - If you installed the shortcut during installation, you can select **Eclipse for DS-5** in the **Applications** menu.
  - If you did not install the shortcut during installation:
    1. Add the *install_directory*/bin directory to your PATH environment variable. If it is already configured then you can skip this step.
    2. Open Unix bash shell.
    3. Enter `eclipse` at the prompt.

When Eclipse launches for the first time, the **Workspace Launcher** dialog box opens. This enables you to select your workspace and accept the default settings. After you select your workspace, click **OK**. You are then presented with the **Welcome** screen.

Language packs

Some features of the workbench are translated into different languages. You can download language packs from the Eclipse website when available.

There are two ways to launch the workbench with a different language pack:

- If your operating system is running in the language that you want to use, then the workbench automatically displays the translated features.
- If your operating system is not running in the language that you want to use, then you must specify the *-nl* command-line argument when launching the workbench.

  For example, to use the Japanese language pack, enter:

  ```
  eclipse -nl ja
  ```

1.3 The welcome screen

The welcome screen contains an overview of the main development environment and links to other pages to help you get started.

Note

Some of the links require an active internet connection.

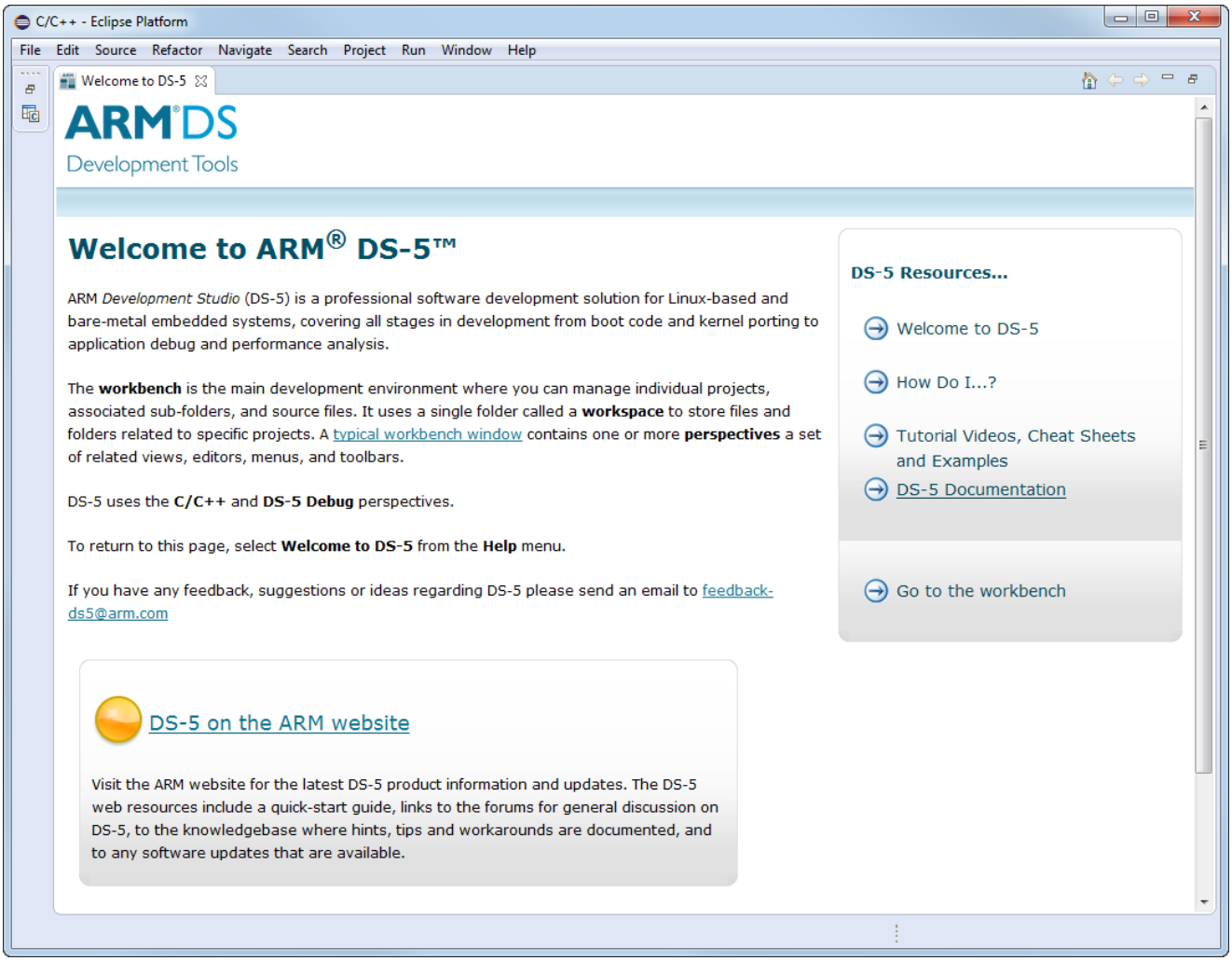

Figure 1-1 Welcome screen

To access the main development environment, either:

- Click on the close icon in the **Welcome to DS-5** tab.
- Click on the **Go to the Workbench** link on the welcome screen.

Select **Welcome to DS-5** from the **Help** menu at any time to return to the welcome screen.

1.4 Overview of the workbench window

A typical workbench window contains a collections of views that are associated with a specific perspective.

For example, the **Project Explorer** view, **C/C++ editor**, **Progress** view, **Console** view, and **Problems** view are all associated with the **C/C++ perspective**. Each perspective automatically adds the most useful views when you select it.

DS-5 uses the **C/C++** and **DS-5 Debug** perspectives.

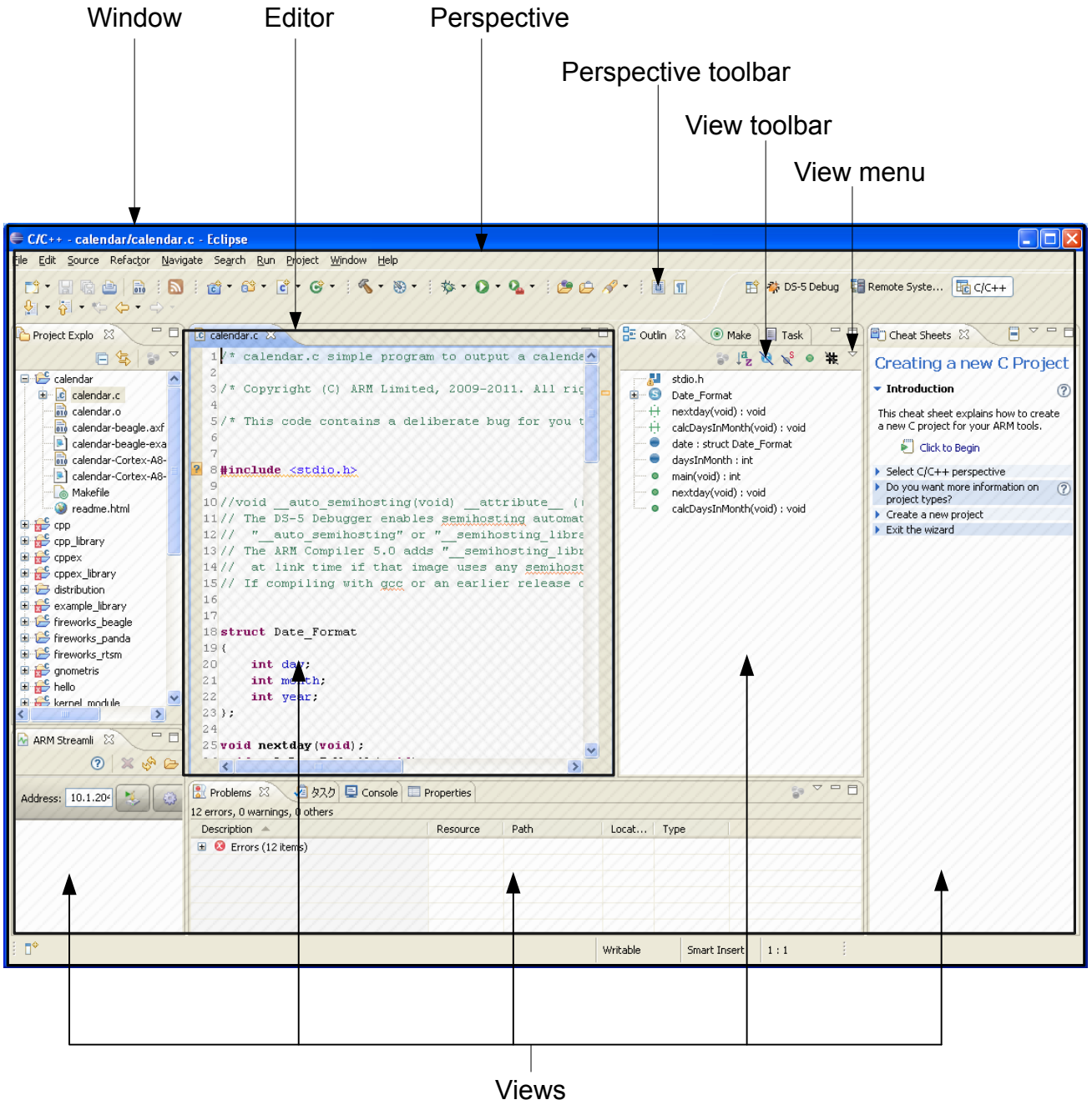

Figure 1-2 Typical workbench window

To close the workbench window and exit Eclipse, select **Exit** from the **File** menu or click on the close icon in the top corner of the window. On exit, Eclipse saves your settings automatically so that when you next open it, the window returns to the same perspective and views.

1.5 Workbench features

The workbench is the main development environment where you can manage individual projects, associated sub-folders, and source files.

Each workbench window is linked to one workspace. If you want to use different workspaces at the same time, you can launch several workbench windows and link each one to a different workspace.

The main workbench features are:

Editors
Editors enable you to view and modify the content of a file, for example source files. The tabs in the editor area show files that are currently open for editing.

Menus and Toolbars
The main menu and toolbar are located at the top of the Eclipse window. Other toolbars associated with specific features are located at the top of each perspective or view.

Perspectives
Perspectives define the layout of your selected views and editors in Eclipse. They also have their own associated menus and toolbars.

Resources
Resources are projects, files, and folders that exist in your workbench.

Views
Views provide related information corresponding to the active file in the editor. They also have their own associated menus and toolbars.

Workspace
Workspace is an area designated on your file system to store files and folders related to your workbench projects and also your personal workbench settings.

1.6 Workspace

The workspace is an area designated on your file system to store files and folders related to your workbench projects and also your personal workbench settings.

Note

ARM recommends that you select a dedicated workspace folder for your workbench projects only. If you select an existing folder containing resources that are not related to workbench projects, you cannot access them in Eclipse. These resources might also cause a conflict later when you create and build projects.

Changes to the customized settings in the **Window** > **Preferences** dialog box are saved in your workspace. If you select a different workspace then these settings might be different.

When Eclipse launches for the first time, the **Workspace Launcher** dialog box opens enabling you to select your workspace.

On subsequent launches the last saved workspace is shown as the default selection in the drop-down menu. You can select another workspace by clicking on the down arrow or the **Browse...** button.

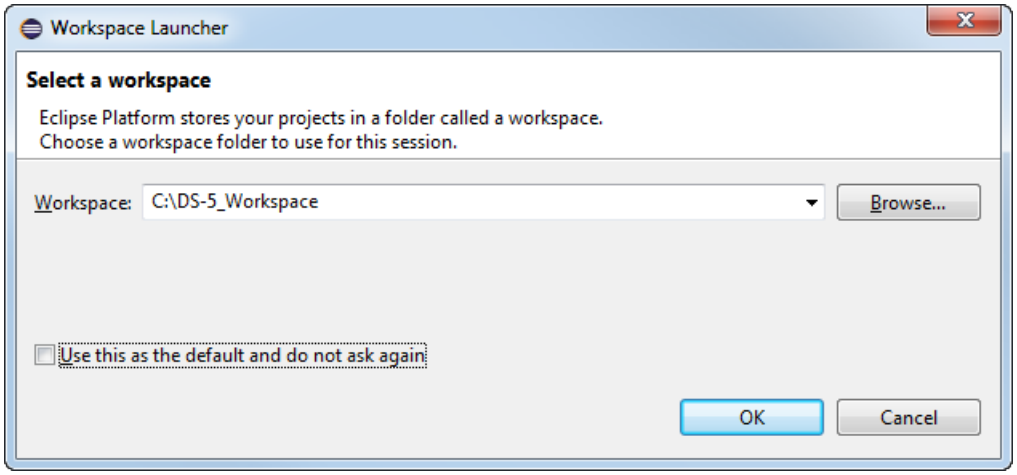

Figure 1-3 Workspace Launcher dialog box

If required, select the checkbox to disable the **Workspace Launcher** dialog box from opening on subsequent launches.

Note

You can change the default workspace at any time by selecting **Switch Workspace...** from the **File** menu.

Alternatively, to open Eclipse and automatically link to a specific workspace you can use the `-data` command-line argument. For example:

```
eclipse -data h:\workspace
```

Editing files outside Eclipse

Project sub-folders and files can be edited even when Eclipse is not running. When you next launch Eclipse, the default preferences enable the relevant views to refresh and update. Alternatively, if you change the default preferences to disable auto-refresh, you can click on the updated sub-folder or file in the **Project Explorer** view and select **Refresh** from the **File** menu.

1.7 Resources

A resource is a generic term used to describe a project, file, folder or a combination of these. Resources exist in Eclipse but might not always exist in the workspace.

The types of resource are:

Project

A project is displayed in the **Project Explorer** view and can be stored within the workspace folder or can be a linked resource.

A project must exist in Eclipse before other resources can be imported or linked to that project. The project creation process creates additional configuration files and folders, for example, build properties. These additional files and folders must not be edited or deleted.

Folder

A folder is displayed in the **Project Explorer** view and can be located within the project folder or can be a linked resource.

File

A file is displayed in the **Project Explorer** view and can be located within the project folder or can be a linked resource.

Linked resources

Resources can be shared between projects or they can exist in the file system outside of your selected workspace. To do this a link must be created within Eclipse.

——— **Note** ———

A linked file or folder must have a project as its parent resource.

Deleting, moving or copying a linked resource only affects the link in your workspace and not the resource that it links to. However, deleting a child resource from within a linked folder also deletes it from the file system.

———————————

Linked file

To link an existing file to a project in your workspace instead of copying it, you can use the advanced options of the **New File** wizard. By default the advanced options are not visible, click on the **Advanced** button to reveal them. A path variable can also be used to reference a file. For more information on referencing with path variables, use the dynamic help.

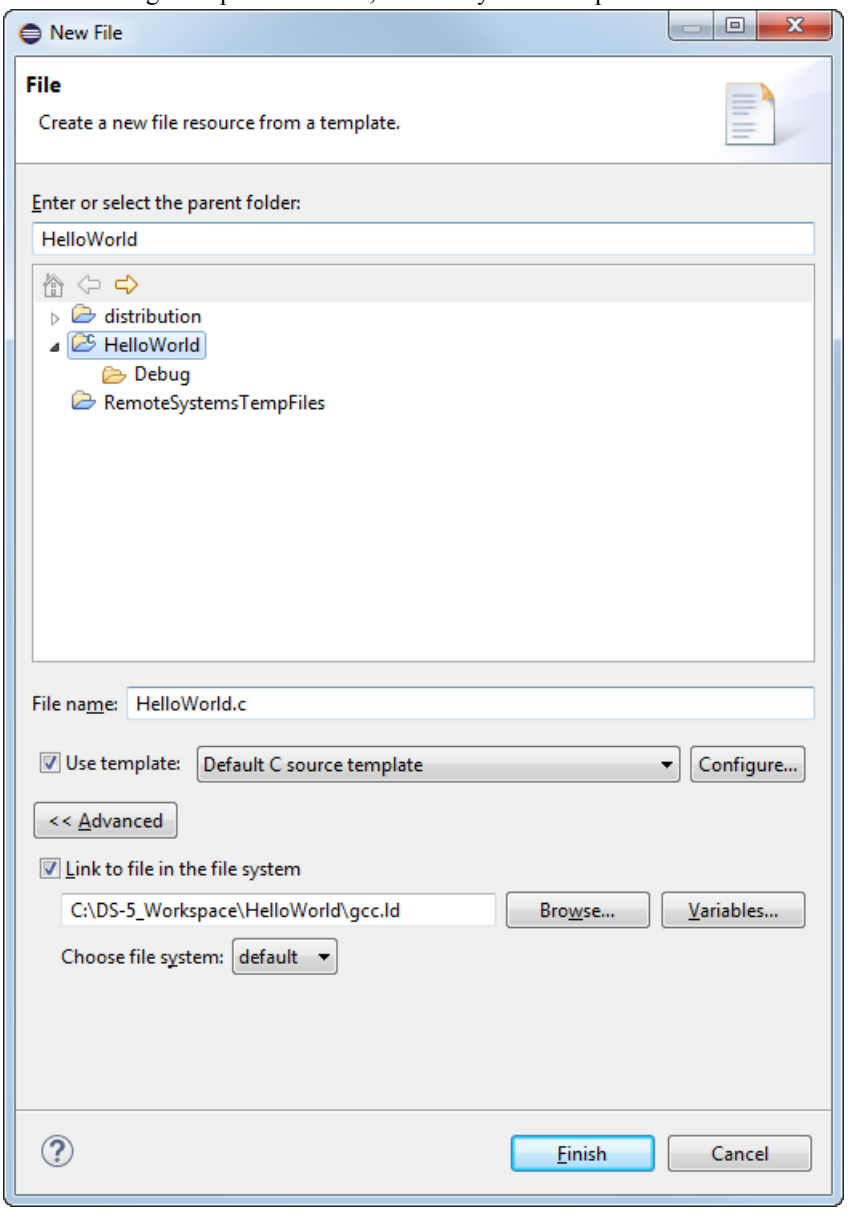

Figure 1-4 Linked file

Linked folder

To link an existing folder to a project in your workspace instead of copying it, you can use the advanced options of the **New Folder** wizard. By default, the advanced options are not visible, click on the **Advanced** button to reveal them. A path variable can also be used to reference a file. For more information on referencing with path variables, use the dynamic help.

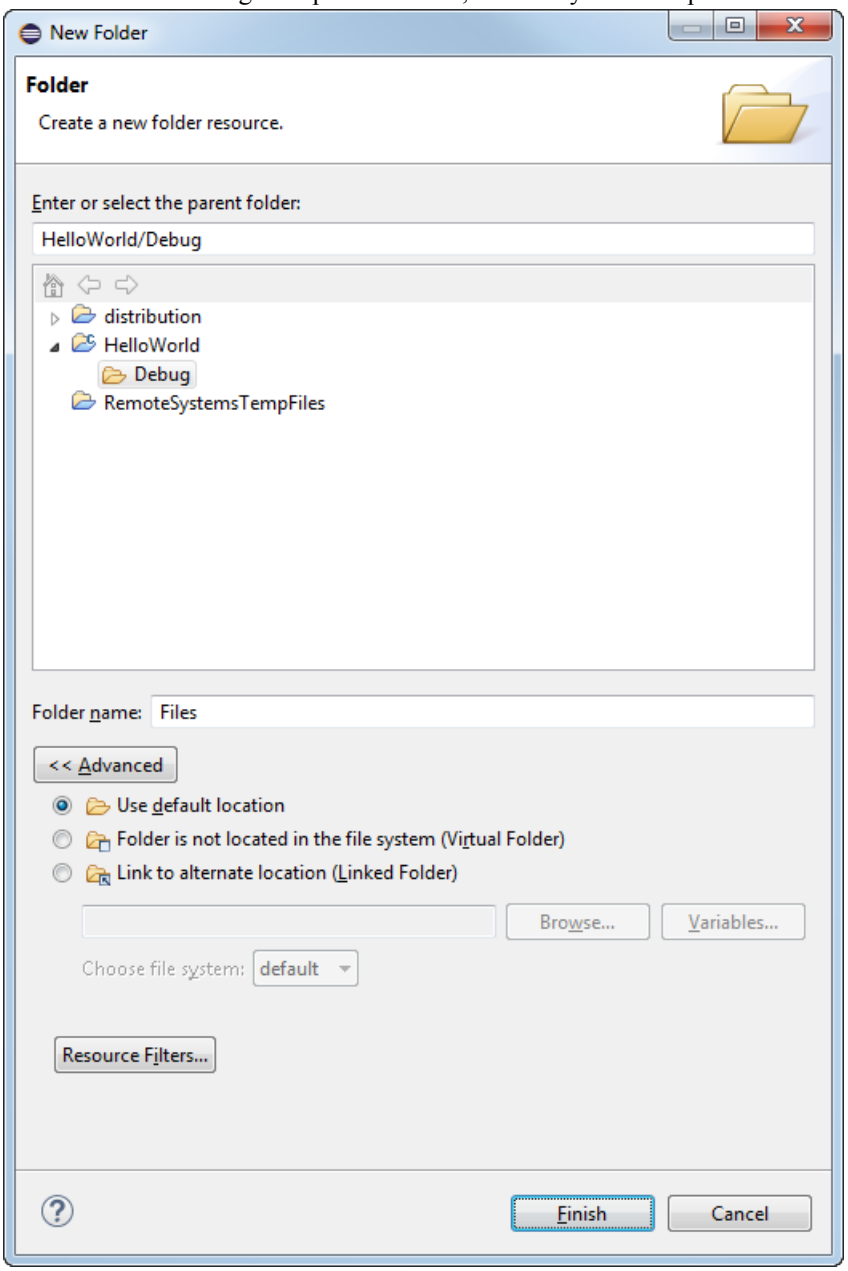

Figure 1-5 Linked folder

Linked project

Eclipse uses the **Import** wizard to create a link to an existing project. This can be useful if you have a central folder with shared projects. By default, the **Copy projects into workspace** option is selected to ensure that a copy of your project is placed in your current workspace. If you want to leave your project where it is and create a link to it, you must deselect this option.

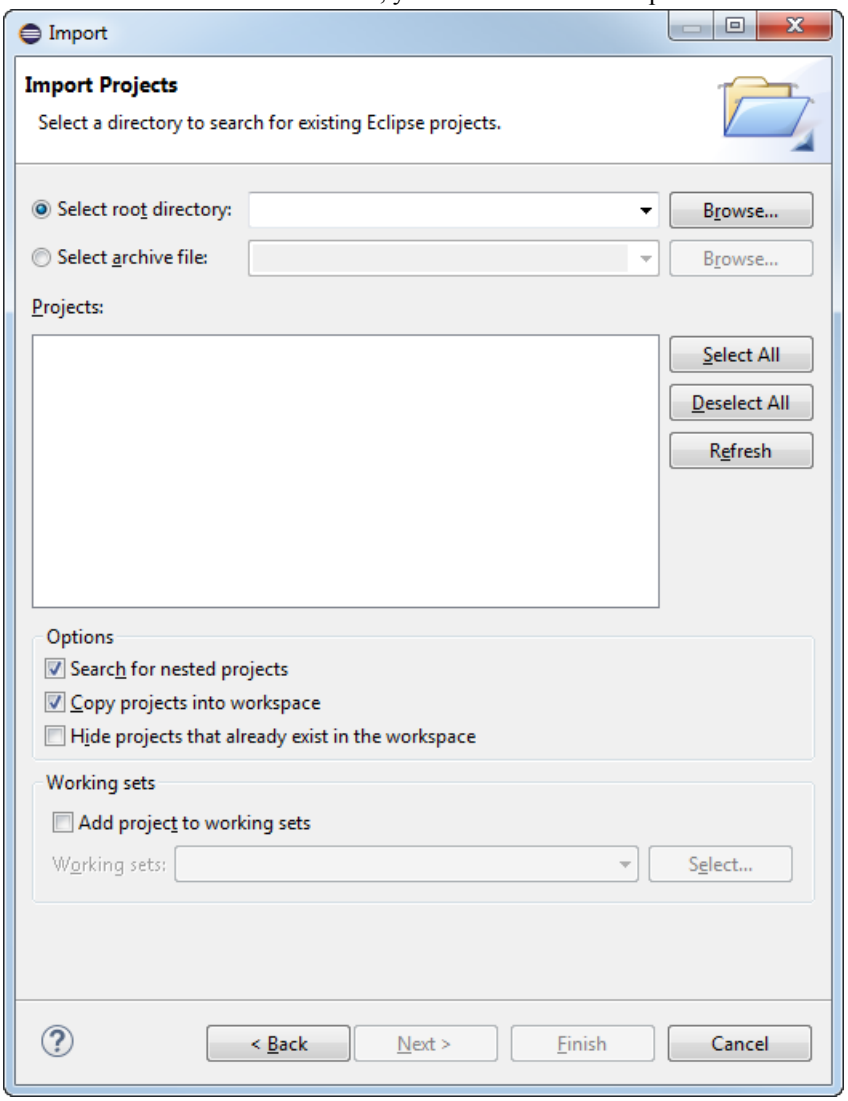

Figure 1-6 Linked project

Disabling the use of linked resources

You can disable the use of linked resources by changing the **General** > **Workspace** > **Linked Resources** settings in the **Preferences** dialog box.

1.8 About working sets

A working set enables you to group projects together and display a smaller subset of projects.

The **Project Explorer** view usually displays a full list of all your projects associated with the current workspace. If you have a lot of projects it can be difficult to navigate through the list to find the project that you want to use.

To make navigation easier, group your projects into working sets. You can select one or more working sets at the same time, or you can use the **Project Explorer View Menu** to switch between one set and another. To return to the original view, select the **Deselect Working Sets** options in the **View Menu**.

Working sets are also useful to refine the scope of a search or build projects in a specific working set.

1.9 Creating a working set

Create working sets to group related projects together.

To create a working set for related projects:

Procedure

1. Click the **View Menu** icon in the **Project Explorer** view toolbar.
2. Select the **Select Working Set...** option.
3. In the **Select Working Set** dialog box, click **New...**.

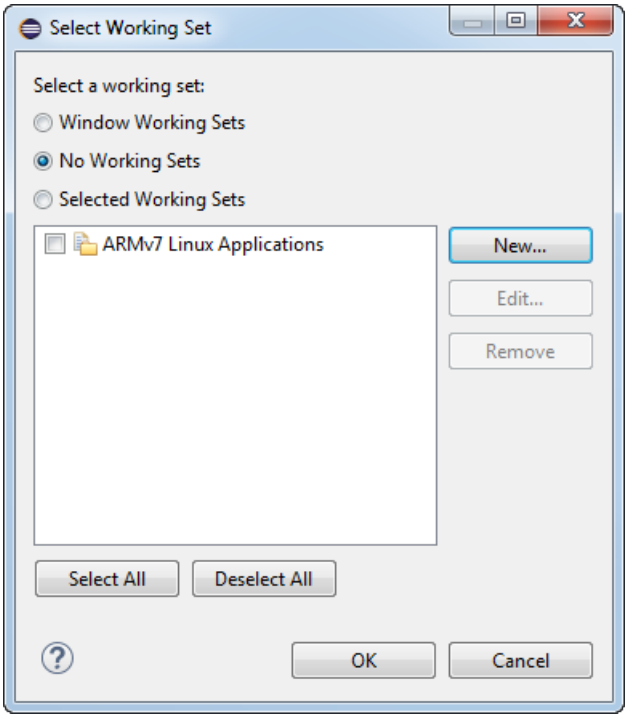

Figure 1-7 Creating a new working set

4. Under **Working set type**, select **Resource** to group related projects.

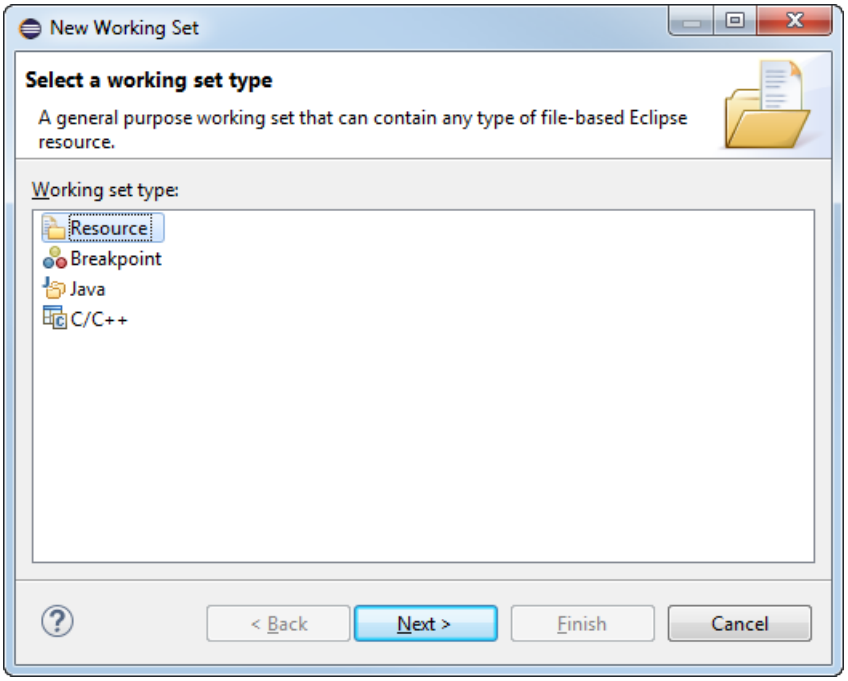

Figure 1-8 Selecting the resource type for the new working set

5. Click **Next**.
6. In the **Working set** name field, enter a suitable name.
7. In the **Working set contents** panel, you can select existing projects that you want to associated with this working or you can return to the wizard later to add projects.

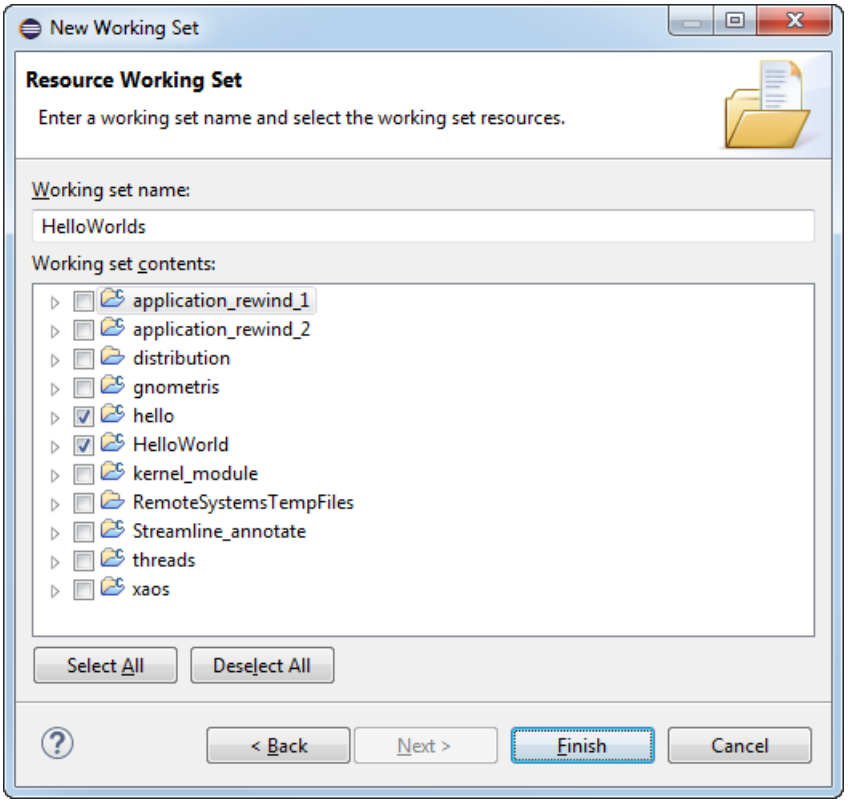

Figure 1-9 Adding new resources to a working set

8. Click **Finish**.

9. If required, repeat these steps to create more working sets.
10. In the **Select Working Set** dialog box, select the working sets that you want to display in the **Project Explorer** view.

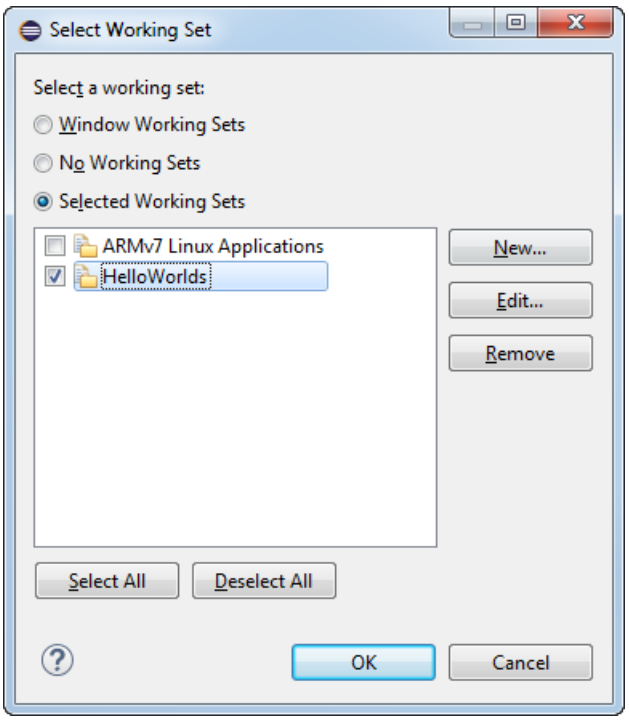

Figure 1-10 Select the required working set

11. Click **OK**.

    The filtered list of projects are displayed in the **Project Explorer** view. Another feature of working sets that can help with navigation is the option to change the top level element in the **Project Explorer** view.

1.10 Changing the top level element when displaying working sets

In the **Project Explorer** view, if you have more than one working set then you might want to display the projects in a hierarchical tree with the working set names as the top level element. This is not selected by default.

To change the top level element in the **Project Explorer** view:

Procedure

1. Click on the **View Menu** icon in the **Project Explorer** view toolbar.
2. Select **Top Level Elements** from the context menu.
3. Select either **Projects** or **Working Sets**.

1.11 Deselecting a working set

You can change the display of projects in the **Project Explorer** view and return to the full listing of all the projects in the workspace.

To do this:

Procedure

1. Click on the **View Menu** icon in the **Project Explorer** view toolbar.
2. Select **Deselect Working Set** from the context menu.

1.12 Perspectives and views

The main workbench window contains one or more perspectives and each perspective contains one or more views.

Perspectives

Perspectives can be opened and customized using the options from the **Window** menu or you can use the **Perspective** toolbar. By default, a perspective opens in the same window, however, you can change the default settings in the **Preferences** dialog box.

Eclipse for DS-5 focuses on the C/C++ perspective and the **DS-5 Debug** perspective. You can change perspective by using the **Perspective** toolbar or you can select **Window** > **Open Perspective** from the main menu.

Figure 1-11 Changing perspective using the toolbar menu

Views

A view is a small visual component within Eclipse to navigate through resources or to display specific properties. An editor is a special type of view that enables viewing and editing source files.

Views can be moved or docked as applicable by dragging and dropping them into position. Double-clicking on a view tab toggles the maximize/restore options for that view or you can use the relevant toolbar icons.

Tip

To reset all the views in a perspective to the initial settings and layout, click **Reset Perspective...** in the **Window** menu.

The following views are useful when building and debugging C/C++ applications:

Editors
Editors are associated with specific file types and open the related editor view automatically when you open an editable file from the **Project Explorer** view.

General views

Console
Displays output messages when building projects and also execution messages when running or debugging an application.

Outline
Displays a structured list of C/C++ elements in the active file. Clicking on an element changes the editor focus to the position of that element in the active file.

Problems
Displays error messages encountered during a build. Selecting an error message opens the associated file and moves the focus to the line causing the problem.

Progress
Displays the progress meter when a task is in progress.

Project Explorer
Provides a hierarchical view of resources relevant to C/C++ project files. Right-clicking on a resource produces a context menu for specific tasks.

Properties
Displays names and values for the selected item. For example, the last modified time/date for a file.

Help views

Help Contents
Displays the installed documentation.

Dynamic Help
Displays dynamic help for the selected feature.

Cheat Sheets
Displays step by step instructions to help you complete a specific task. Use the view menu to access the list of cheat sheets.

DS-5 Debugger views

App Console
Enables you to interact with the I/O capabilities provided by the semihosting implementation in the ARM® C libraries.

ARM Asm Info
Enables you to view more information on an ARM or Thumb® instruction or directive.

Breakpoints
Displays all the breakpoints and the respective locations in source code or target memory. Using this view you can add, edit, delete, enable, or disable breakpoints.

Commands
Provides access to the scripting functionality of the debugger. Most debugger actions are echoed here and you can also execute DS-5 Debugger commands.

Debug Control
Contains a tree hierarchy showing debug connection states, processors, threads and stack frames as applicable.

Disassembly
Displays disassembled code.

Expressions
Enables you to view and edit expressions.

Functions
Enables you to view ELF data associated with function symbols for all loaded images.

History
Provides a historical log of all the executed debugger commands.

Memory
Enables you to view and edit the contents of memory.

Modules
Enables you to view shared libraries or loaded *Operating System* (OS) modules.

Registers
Enables you to view and edit registers associated with the selected stack frame. When your application stops, modified values are highlighted.

Screen
Enables you to display a block of memory as a graphical image. For example, a frame buffer.

Scripts
Provides access to run and edit script files.

Target
Displays a list of capabilities for the target device that is currently connected to the debugger.

Trace
Displays the captured trace information from the debug agent.

Variables
Enables you to view and edit variables associated with the selected stack frame.

Remote Systems views

Remote Scratchpad
Enables you to copy and paste or drag and drop local resources to an electronic clipboard for use at a later point in time.

Remote Systems
Enables you to create connections and access resources on the host workstation and remote targets.

Terminals
Enables you to enter shell commands directly on the target without launching any external application.

Other views can be added to a perspective but they might not be compatible with all perspectives. To add a view to a perspective:

1. Ensure that you are in the relevant perspective.
2. Select **Window** > **Show View** > **Other...** to open the **Show View** dialog box.
3. Select the required view.
4. Click **OK**.

For more information on the other views not listed here, use the dynamic help.

Trim bar

Minimized views are placed in the trim bar at the edge of the current perspective. By default, the trim bar is on the right-hand margin of the perspective but you can drag and drop it to another location if you prefer. In the trim bar, you can double-click on a view to restore it or click on the restore icon to restore all the minimized views.

Figure 1-12 Minimized views in the trim area

1.13 Menus

The main menu is located at the top of the Eclipse window and can be customized to your personal preferences. The contents might vary depending on the installed plug-ins and also the active perspective.

Eclipse supports the following options from the main menu:

File
: Enables you to create, save, close, print, import, and export resources. You can also manage project and file property settings.

Edit
: Enables you to cut, copy, paste, find, and replace text within a resource.

Navigate
: Enables you to navigate and quickly find specific resources.

Search
: Provides an advanced filter for searching through resources.

Project
: Enables you to manage project build configurations and perform specific builds. You can also customize the build settings for ARM tools.

Run
: Enables you to run, send to, debug or configure external tools. You can also manage breakpoints and watchpoints.

Window
: Enables you to open, close, and customize perspectives, views and editors.

Help
: Provides documentation on Eclipse and the ARM tools, access to the ARM cheat sheets, and software updates. You can also access the menu option **About ARM DS-5** to open a dialog box that displays the version and build number for the current product.

Right-clicking on a resource produces a context menu for specific tasks. For more information on the other menu options not listed here, use the dynamic help.

1.14 Toolbars

The main toolbar is located at the top of the Eclipse window and can be customized to your personal preferences. The toolbar icons vary depending on the installed plug-ins and also the active perspective.

Figure 1-13 Workbench toolbar

Other toolbars associated with specific features are located at the top of each perspective or view.

Figure 1-14 Perspective toolbar

Figure 1-15 View toolbar

1.15 Editing source code

You can use the editors provided with Eclipse to edit your source code or you can use an external editor. If you work with an external editor you must refresh Eclipse to synchronize the views with the latest updates.

To do this, in the Project Explorer view, you can click on the updated project, sub-folder, or file and select **Refresh** from the **File** menu. Alternatively you can enable the automatic refresh option by selecting **General** > **Workspace** > **Refresh automatically** in the Preferences dialog box.

When you open a file in Eclipse, a new editor tab appears with the name of the file. An edited file displays an asterisk (*) in the tab name to show that it has unsaved changes.

When you have two or more editor tabs open, you can tile them for side-by-side viewing by clicking on a tab and dragging it over an editor border.

In the left-hand margin of the editor tab you can find a vertical bar that displays markers relating to the active file.

Navigating

There are several ways to navigate to a specific resource within Eclipse. You can use the Project Explorer view to open a resource by browsing through the resource tree and double-clicking on a file. An alternative is to use the keyboard shortcuts or use the options from the **Navigate** menu.

Searching

To locate information or specific code contained within one or more files in Eclipse, you can use the options from the **Search** menu. Textual searching with pattern matching and filters to refine the search fields are provided in a customizable Search dialog box. You can also open this dialog box from the main workbench toolbar.

Content assist

The C/C++ editor, ARM assembler editor, and the DS-5 Debugger Commands view provide content assistance at the cursor position to auto-complete the selected item. Using the Ctrl+Space keyboard shortcut produces a small dialog box with a list of valid options to choose from. You can shorten the list by partially typing a few characters before using the keyboard shortcut. From the list you can use the Arrow Keys to select the required item and then press the Enter key to insert it.

Bookmarks

You can use bookmarks to mark a specific position in a file or mark an entire file so that you can return to it quickly. To create a bookmark, select a file or line of code that you want to mark and select **Add Bookmark** from the **Edit** menu. The Bookmarks view displays all the user defined bookmarks and can be accessed by selecting **Window** > **Show View** > **Bookmarks** from the main menu. If the Bookmarks view is not listed then select **Others...** for an extended list.

To delete a bookmark, open the Bookmarks view, click on the bookmark that you want to delete and select **Delete** from the **Edit** menu.

1.16 Configuring Eclipse

You can customize Eclipse to your own settings by changing the layout, key bindings, file associations, and color schemes.

You can also:

- Configure Projects and files to use the build system in different ways by modifying the properties for the selected resource.
- Open and customize perspectives using the options from the **Window** menu or you can use the perspective toolbar. By default a perspective opens in the same window, however, you can change the default settings in the Preferences dialog box.
- Move or dock views as applicable by dragging and dropping them into position. Double-clicking on the title bar of a view toggles the maximize/restore options or you can use the relevant toolbar icons. If you have minimized a view you can restore it by clicking on the restore icon in the toolbar at the side of the perspective. By default this is on the right-hand side of the perspective but you can drag and drop it to another location if you prefer. To reset all the views in a perspective to the initial settings and layout select **Reset Perspective...** in the **Window** menu.

1.17 Preferences dialog box

You can be customize workbench settings using the **Preferences** dialog box.

To access the **Preferences** dialog box select **Preferences...** from the **Window** menu. Changes to these settings are saved in the current workspace. If you want to copy your workbench settings to another workspace, use the **Export** wizard.

The contents of the preferences hierarchy tree include the following groups:

General
: Controls the workspace, perspectives, editors, build order, linked resources, file associations, path variables, background operations, keyboard and mouse settings.

C/C++
: Controls the C/C++ environment settings, CDT build variables, syntax formatting, and default project wizard settings.

DS-5
: Controls the default DS-5 environment settings, presentation and formatting for DS-5 editors and views, target configuration database search locations, and the automatic checks for DS-5 product updates.

Help
: Controls how the context help is displayed.

Install/Update
: Controls the update history, scheduler, and policy.

Remote Systems
: Controls the settings used by the **Remote System Explorer**.

Run/Debug
: Controls the default perspectives, breakpoint, build, and launch settings before running and debugging.

For more information on the other options not listed here, use the dynamic help.

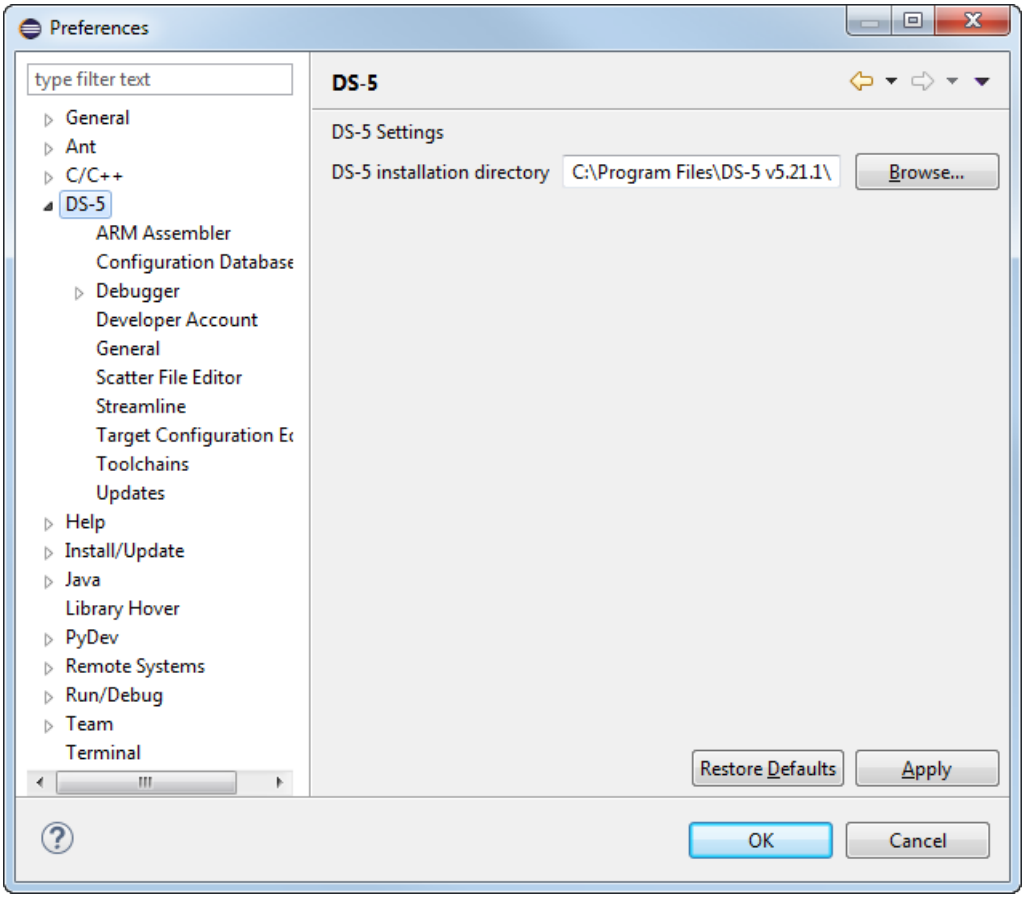

Figure 1-16 Window preferences dialog box

1.18 Properties dialog box

You can customize project settings using the Properties dialog box.

To access the Properties dialog box select a project and then select **Properties...** from the **Project** menu. Changes to the customized settings are saved in the project folder in your workspace. You can also customize the C/C++ properties for a single file for example, if you want to apply a specific compiler option to a file during the build.

——— **Note** ———

If you specify different options for a single file, it overrides the options specified in the project configuration panels that apply to all related source files.

———————————

The contents of the properties hierarchy tree for a project include the following:

Resource
: Displays the resource location, modification state, and file type.

Builders
: Controls builders available for the selected project.

C/C++ Build
: Controls the environment, build, and tool chain settings for the active configuration.

C/C++ General
: Controls documentation, file types, indexer and path/symbol settings.

Project References
: Controls project dependencies.

For more information on the other options not listed here, use the dynamic help.

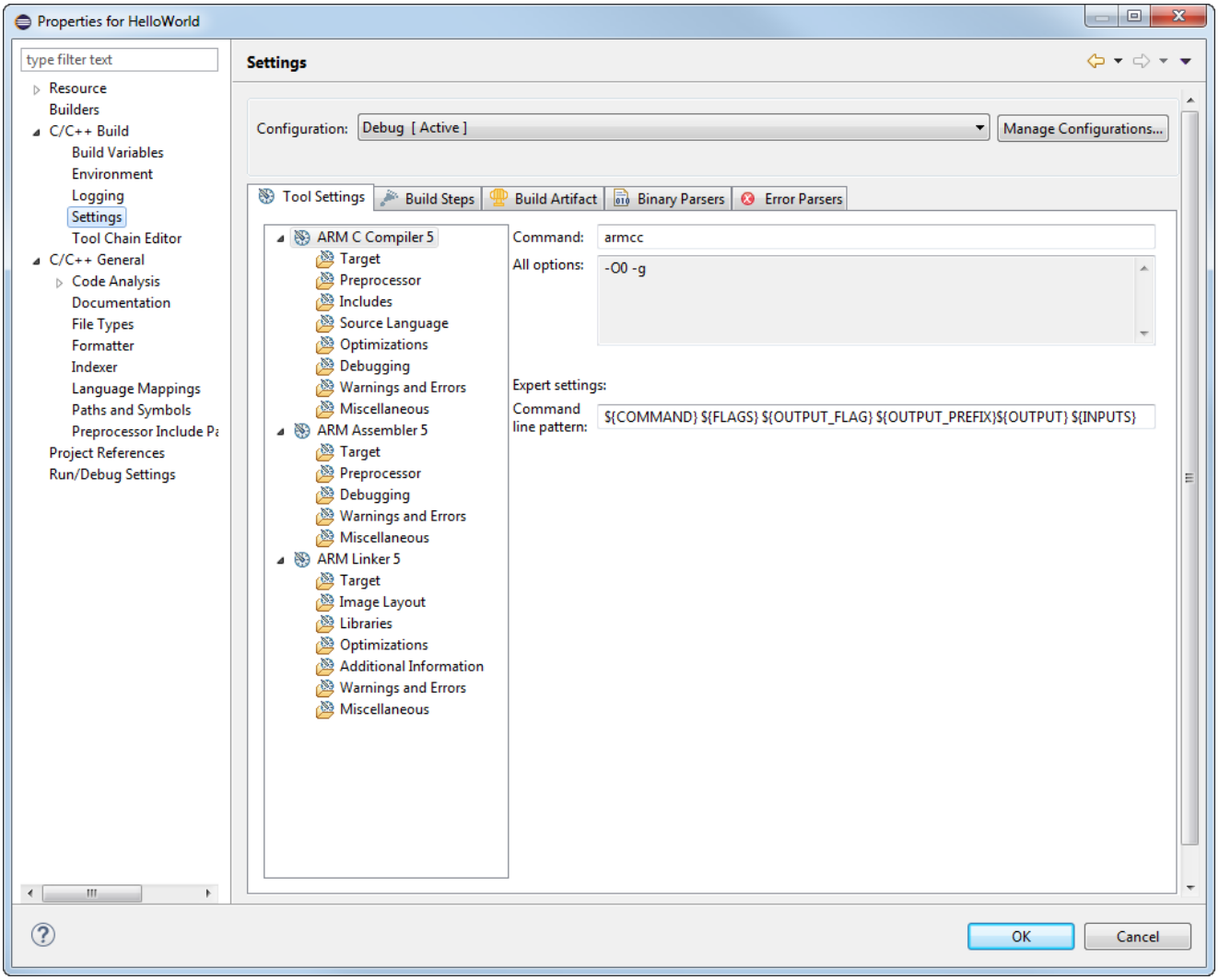

Figure 1-17 Project properties dialog box

1.19 Importing and exporting options

A resource must exist in a project within Eclipse before you can use it in a build.

If you want to use an existing resource from your file system in one of your projects, the recommended method is to use the Import wizard. To do this, select **Import...** from the **File** menu.

If you want to use a resource externally from Eclipse, the recommended method is to use the Export wizard. To do this, select **Export...** from the **File** menu.

There are several options available in the import and export wizards:

General

This option enables you to import and export the following:

- Files from an archive zip file.
- Complete projects.
- Selected source files and project sub-folders.
- Workbench preference settings.

C/C++

This option enables you to import the following:

- C/C++ executable files.
- C/C++ project settings.
- Existing code as Makefile project.

You can also export C/C++ project settings and indexes.

Remote Systems

This option enables you to transfer files between the local host and the remote target.

Run/Debug

This option enables you to import and export the following:

- Breakpoint settings.
- Launch configurations.

Scatter File Editor

This option enables you to import the memory map from a BCD file and convert it into a scatter file for use in an existing project.

For information on the other options not listed here, use the dynamic help.

1.20 Using the Import wizard

You can be use the Import wizard to import complete projects, source files and, project sub-folders in addition to breakpoint and preference settings.

Select **Import...** from the **File** menu to display the **Import** wizard.

Importing complete projects

To import a complete project either from an archive zip file or an external folder from your file system, you must use the Existing Projects into Workspace wizard. This ensures that the relevant workbench project files are also imported into your workspace.

Importing source files and project sub-folders

Individual source files and project sub-folders can be imported using either the **Archive File** or **File System** wizard. Both options produce a dialog box similar to the following example. Using the options provided you can select the required resources and specify the relevant options, filename, and destination path.

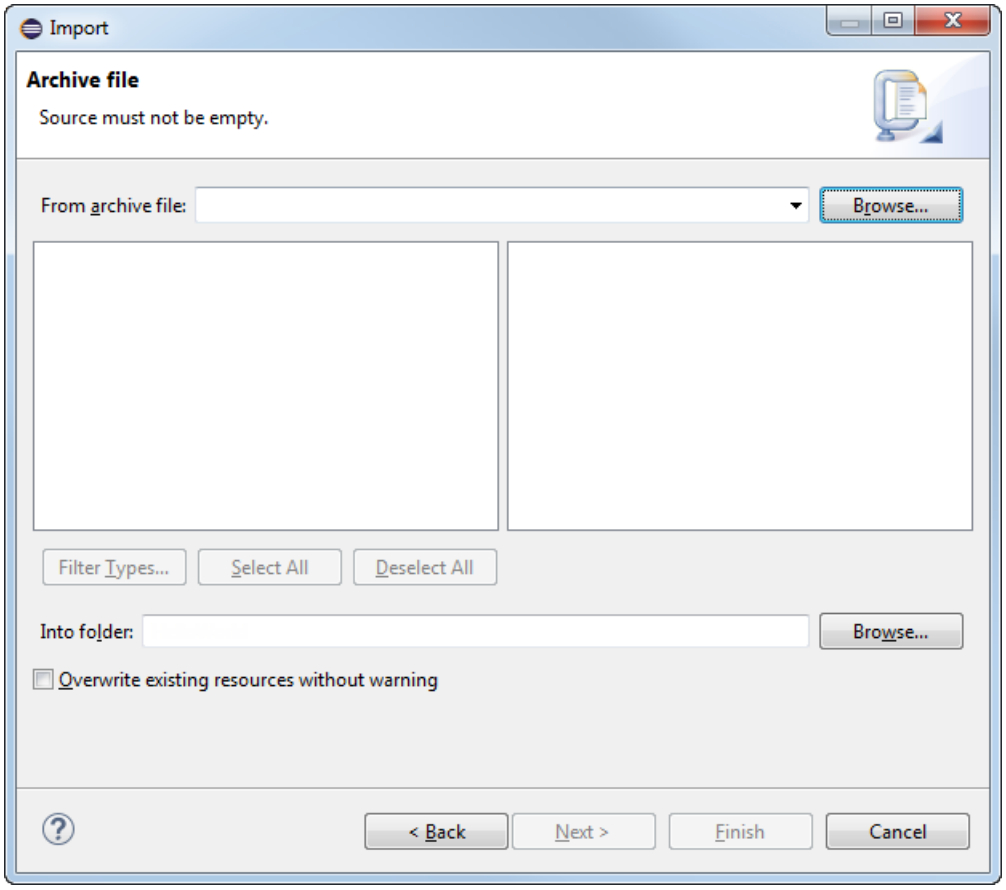

Figure 1-18 Typical example of the import wizard

With the exception of the **Existing Projects** into **Workspace** wizard, files and folders are copied into your workspace when you use the **Import** wizard. To create a link to an external file or project sub-folder you must use the **New File** or **New Folder** wizard.

1.21 Using the Export wizard

You can use the **Export** wizard to export complete projects, source files and, project sub-folders in addition to breakpoint and preference settings.

Select **Export...** from the **File** menu to display the **Export** wizard.

The procedure is the same for exporting a complete project, a source file, and a project sub-folder. If you want to create a zip file you can use the **Archive File** wizard, or alternatively you can use the **File System** wizard. Both options produce a dialog box similar to the example shown here. Using the options provided you can select the required resources and specify the relevant options, filename, and destination path.

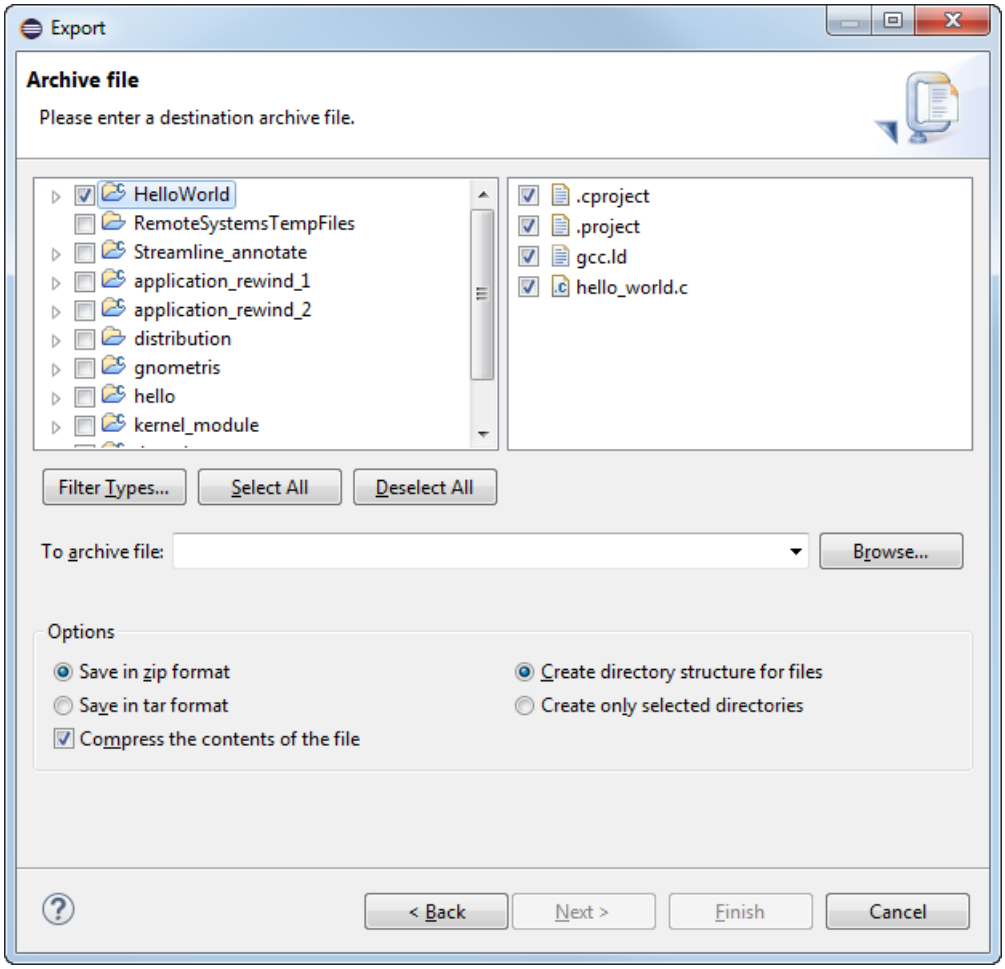

Figure 1-19 Typical example of the export wizard

1.22 Accessing the dynamic help

Dynamic help provides help on the currently selected view. When you select a different view, the help changes to help for that view.

To access the dynamic help for a specific workbench feature you must:

Procedure

1. Click on an editable field for the feature that you want to use.
2. Do one of the following:
   - Click on the question mark icon.
   - Select **Dynamic Help** from the **Help** menu.
   - Use a keyboard shortcut:
     — On Windows, **F1** key.
     — On Linux, **Shift+F1** key combination.

Dynamic help for the selected feature appears in a panel. Other possible search results are also listed.

1.23 Using ARM®-specific cheat sheets

Cheat sheets are working examples that you can use to guide you through a specific task. Each step in the task is listed in the **Cheat Sheets** view and the current step is highlighted and expanded. You must perform each step in turn to complete the task.

To use a cheat sheet:

Procedure

1. Select **Cheat Sheet...** from the **Help** menu.
2. Select a cheat sheet from the list.

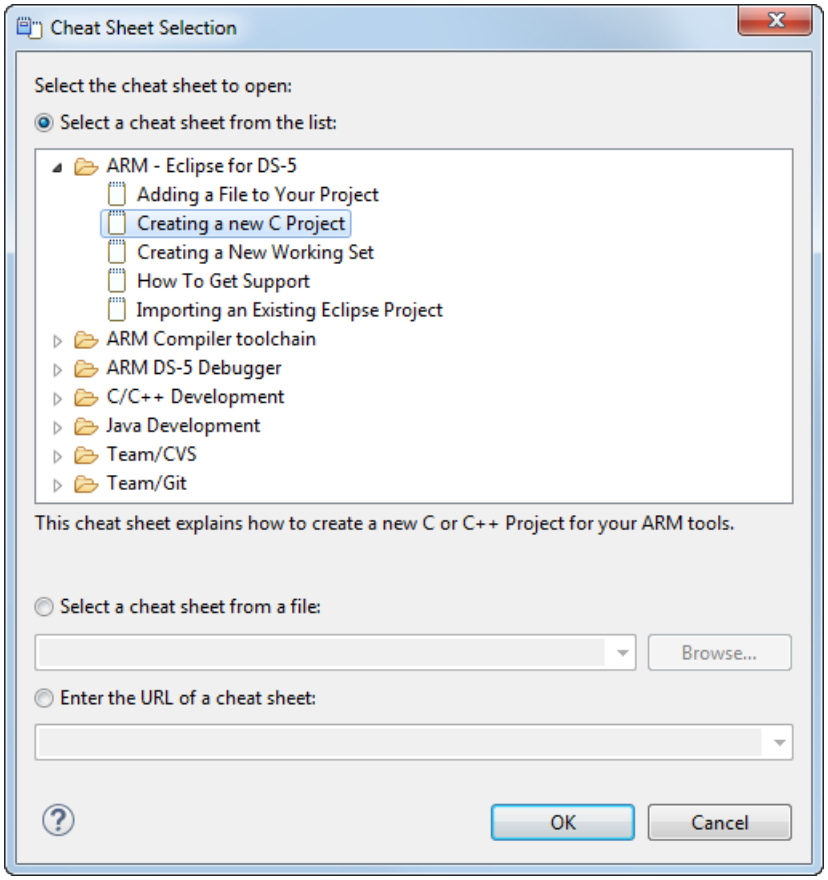

Figure 1-20 Selecting a cheat sheet

3. Click **OK**.
4. In the **Cheat Sheets** view, click the **Click to Begin** link on the **Introduction** step to start the task. If you open a cheat sheet more than once, the link on the **Introduction** step changes text to **Click to Restart**.
5. Follow the instructions step by step. When you complete an instruction, click the **Click to Complete** link to move on to the next instruction. Some of the instructions might have a **Click to Perform** link for you to use if you want that instruction to be performed automatically by the cheat sheet.

The task is complete when you have performed every step listed in the **Cheat Sheets** view.

The following figure shows a typical example of a cheat sheet. Steps one and two are shown as complete, step three is skipped, and step four (the current step) is expanded and ready for use. When step four is complete, the cheat sheet moves on to reveal the instructions in step five.

Note

Subsequent steps are not populated until you fully complete or skip the previous step.

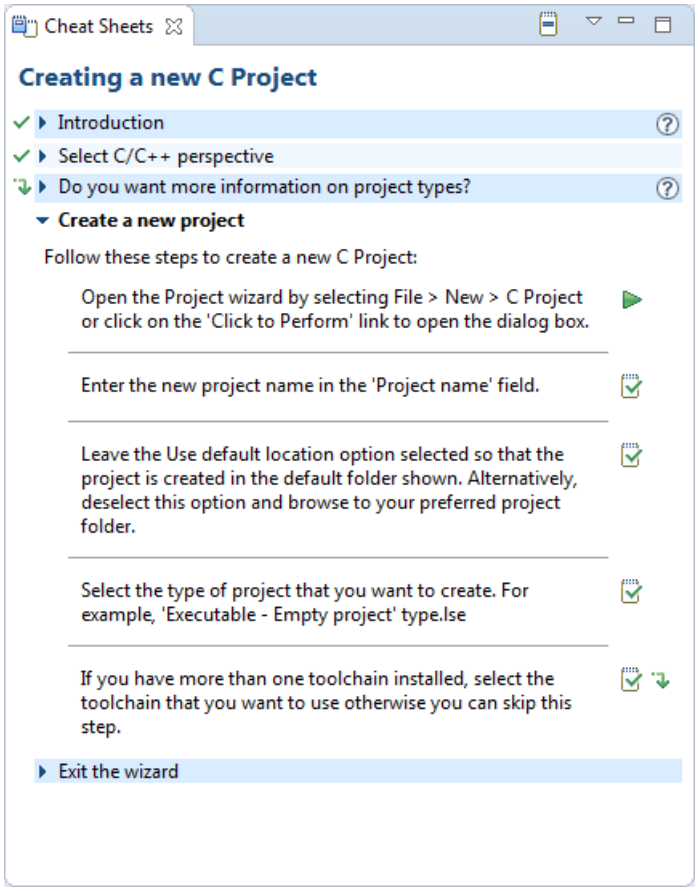

Figure 1-21 Typical example of a cheat sheet

1.24 Installing new features

When a new feature is available, you can install it using Eclipse.

To install new features:

Procedure

1. Select **Install New Software...** from the **Help** menu.
2. Select a site from the **Work with** drop-down list or click **Add...** to add a new site.
3. Select the required plug-in check boxes and click **Next >**.
4. Review the list of plug-ins that you are about to install and click **Next >**.
5. Read each of the license agreements and accept them. If you do not accept a license agreement, you cannot install that feature.
6. Click **Finish**.
7. Click **Yes** to restart Eclipse and complete the installation.

Note

- You can only update features that you have installed on top of the ones provided by DS-5. To update a feature, select **Check for Updates** from the **Help** menu.
- You can change the default setting for downloading and installing updated features by selecting **Install/Update** > **Automatic Updates** in the **Preferences** dialog box.

1.25 Restrictions of use

There are specific restrictions and peculiarities that apply when using Eclipse.

These are:

Organizing projects
: The recommended structure for project source files is to create them in the project folder or sub-folder. If a source file is created in a folder that is higher than the project, an absolute link is created.

Opening an existing Eclipse project
: You must use the import wizard to import an existing Eclipse project into your workspace.

Inter-project dependencies
: Nested projects are not supported. Each project must be organized as a discrete entity. Inter-project dependencies can be set up by referencing other projects that reside in your workspace. Select **Project** > **Properties** > **Project References** from the main menu to manually add references.

Link order
: Specifying the link order of your object files within the same project is not possible with Eclipse. As a workaround, if you split your object files into different projects, you can specify the project build order. Select **Window** > **Preferences...** > **General** > **Workspace** > **Build Order** from the main menu.

Restore Defaults
: Restoring the defaults of a project discards all information that is not a part of the project type. All settings changed in the New Project Wizard are lost.

1.26 Data collection in DS-5

ARM periodically collects anonymous DS-5 usage data.

Collecting anonymous data helps ARM better understand how you use DS-5 features so it can be improved.

The collected data contains no personal information and is anonymous.

To disable data collection, from the main menu, select **Window** > **Preferences** > **DS-5** > **General**, and deselect the **Allow collection of anonymous analytics data** option.

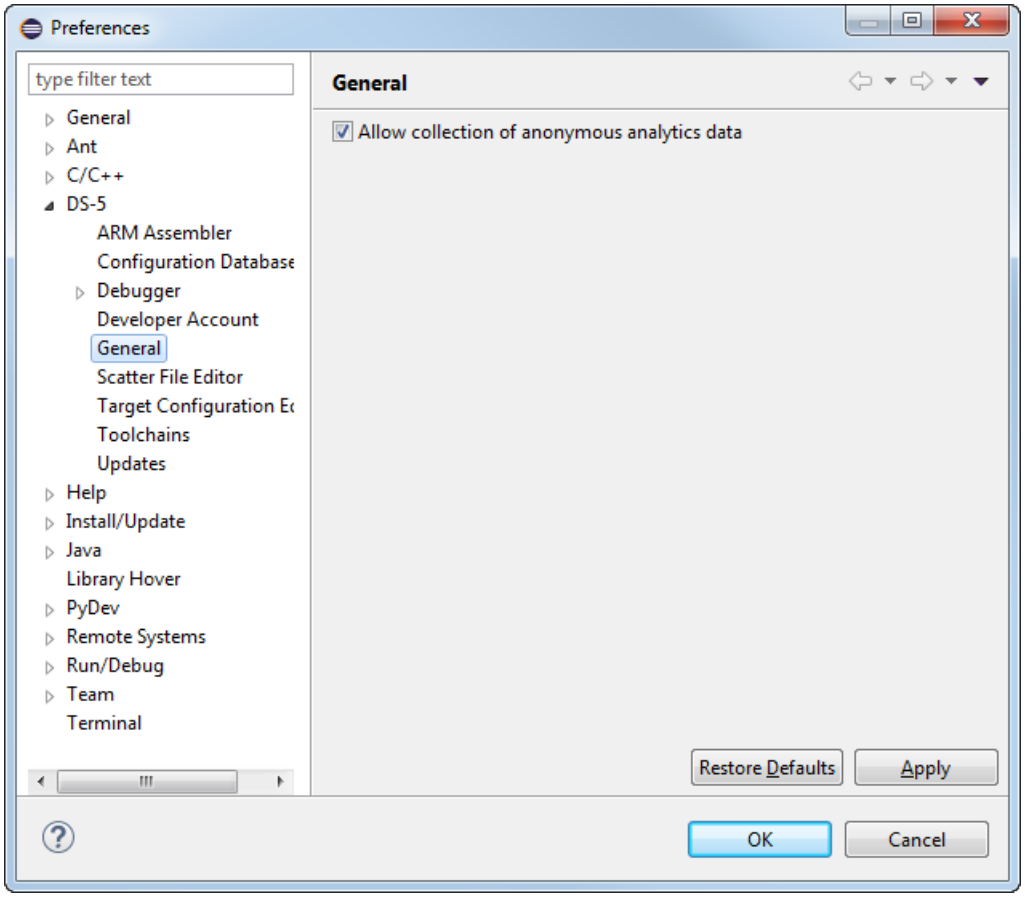

Figure 1-22 DS-5 General options - Data collection

Chapter 2
Working with projects

You can use Eclipse to create projects for ARM targets. Projects are top level folders in your workspace that contain related files and sub-folders. A project must exist in your workspace before a new file can be added or an existing file can be imported.

It contains the following sections:

2.1 Project types

Different project types are provided with Eclipse, depending on the requirements of your project.

——— **Note** ———

Bare metal projects require a software license for ARM Compiler to successfully build an ELF image.

———————————

Bare-metal Executable
Uses ARM Compiler to build a bare-metal executable ELF image.

Bare-metal Static library
Uses ARM Compiler to build a library of ELF object format members for a bare-metal project.

——— **Note** ———

It is not possible to debug or run a stand-alone library file until it is linked into an image.

———————————

Executable
Uses the GNU Compilation Tools to build a Linux executable ELF image.

Shared Library
Uses the GNU Compilation Tools to build a dynamic library for a Linux application.

Static library
Uses the GNU Compilation Tools to build a library of ELF object format members for a Linux application.

——— **Note** ———

It is not possible to debug or run a stand-alone library file until it is linked into an image.

———————————

Makefile project
Creates a project that requires a makefile to build the project. However, Eclipse does not automatically create a makefile for an empty Makefile project. You can write the makefile yourself or modify and use an existing makefile.

——— **Note** ———

Eclipse does not modify Makefile projects.

———————————

Build configurations

By default, the new project wizard provides two separate build configurations:

Debug
The debug target is configured to build output binaries that are fully debuggable, at the expense of optimization. It configures the compiler optimization setting to minimum (level 0), to provide an ideal debug view for code development.

Release
The release target is configured to build output binaries that are highly optimized, at the expense of a poorer debug view. It configures the compiler optimization setting to high (level 3).

In all new projects, the `Debug` configuration is automatically set as the active configuration. This can be changed in the C/C++ Build Settings panel of the Project Properties dialog box.

——— **Note** ———

C project
This does not select a source language by default and leaves this decision up to the compiler. Both GCC and ARM Compiler default to C for `.c` files and C++ for `.cpp` files.

C++ project
Selects C++ as the source language by default, regardless of file extension.

In both cases, the source language for the entire project a source directory, or individual source file can be configured in the build configuration settings.

2.2 Creating a new C or C++ project

Use the options in the **C Project** dialog to create a new C or C++ project.

To create a new C or C++ Project:

Procedure

1. Select **File** > **New** > **Project...** from the main menu.
2. Expand the **C/C++** group.
3. Select either **C Project** or **C++ Project**.
4. Click on **Next**.
5. Enter a **Project name**.
6. Leave the **Use default location** option selected so that the project is created in the default folder shown. Alternatively, deselect this option and browse to your preferred project folder.
7. Select the type of project that you want to create.

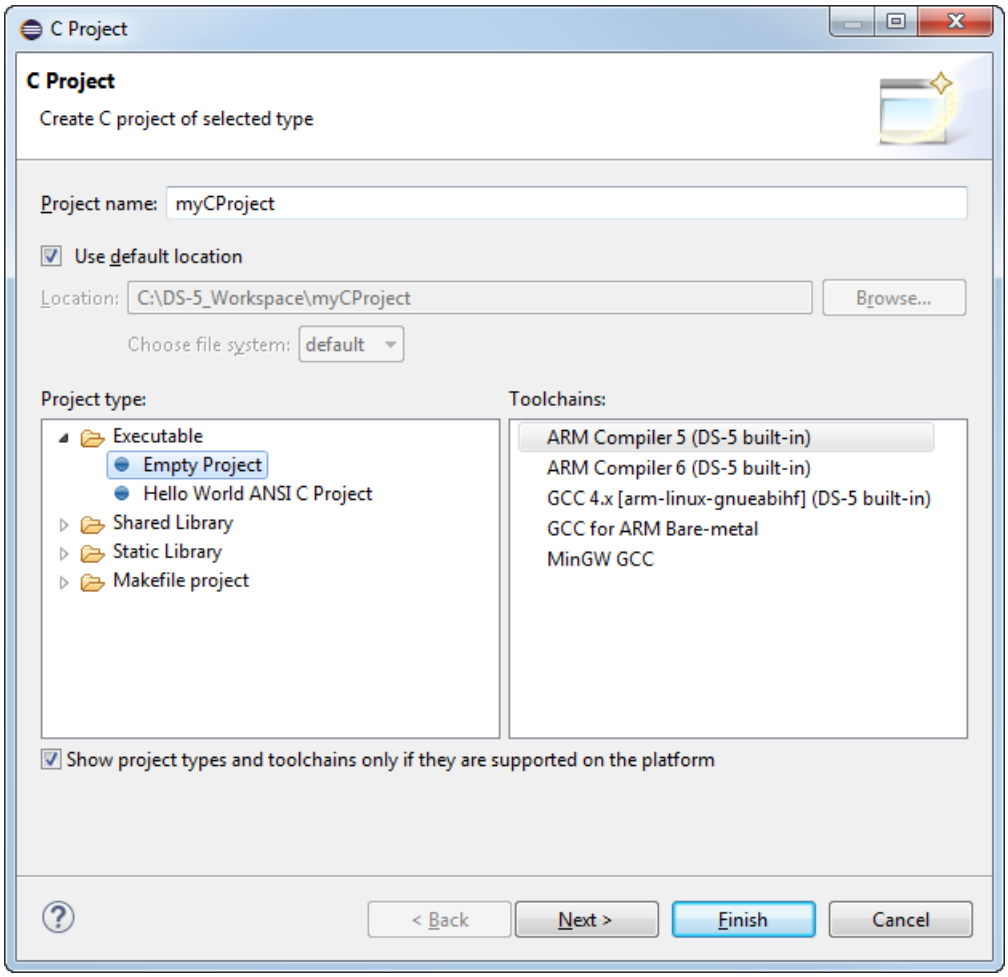

Figure 2-1 Creating a new C project

8. Select a **Toolchain**.
9. Click on **Finish** to create your new project.

You can view the project in the **Project Explorer** view.

Note

C project
This does not select a source language by default and leaves this decision up to the compiler. Both GCC and ARM Compiler default to C for `.c` files and C++ for `.cpp` files.

C++ project
Selects C++ as the source language by default, regardless of file extension.

In both cases, the source language for the entire project, a source directory or individual source file can be configured in the build configuration settings.

2.3 Creating an empty Makefile project

Sometimes, you might want to create an empty C or C++ Makefile project.

This example shows how to do this for an ARM Linux target:

Procedure

1. Select **File** > **New** > **Project...** from the main menu.
2. Expand the **C/C++** group.
3. Select either **C Project** or **C++ Project**.
4. Click on **Next**.
5. Enter a project name.
6. Leave the **Use default location** option selected so that the project is created in the default folder shown. Alternatively, deselect this option and browse to your preferred project folder.
7. Expand the **Makefile project** group.
8. Select **Empty project** in the Project type panel.
9. Select the toolchain that you want to use when building your project. For example if your project is for an ARM Linux target, select the appropriate GCC toolchain.
10. Click on **Finish** to create your new project. The project is visible in the Project Explorer view.
11. Before you can build the project, a `Makefile` containing all the compilation tool settings is required. The easiest way to create one is to copy the `Makefile` from the example project, `hello` and paste it into your new project. The `hello` project is in the Linux examples provided with DS-5.
12. Edit the `Makefile` as follows:
    a. Locate the line that contains `OBJS = hello.o`.
    b. Replace `hello.o` with the names of the object files corresponding to your source files.
    c. Locate the line that contains `TARGET = hello`.
    d. Replace `hello` with the name of the target image file corresponding to your source files.
    e. Save the file.
13. Right-click the project and then select **Properties** > **C/C++ Build** to access the build settings. In the **Builder Settings** tab, ensure that the **Build directory** points to the location of the `Makefile`.
14. Add your C/C++ files to the project.

Postrequisites

You can now build the new project:

1. In the **Project Explorer** view, select the project that you want to build.
2. Select **Project** > **Build Project** from the main menu.

2.4 Creating a new Makefile project with existing code

You can create a new Makefile project in Eclipse with your existing source code.

The following procedure shows how to create a new Makefile project in the same directory as your source code:

Procedure

1. Select **File** > **New** > **Project...** from the main menu.
2. Expand the **C/C++** group.
3. Select **Makefile Project with Existing Code**.
4. Click on **Next**.
5. Enter a project name and enter the location of your existing source code.
6. Select the toolchain that you want to use for Indexer Settings. Indexer Settings provide source code navigation in the Eclipse IDE.

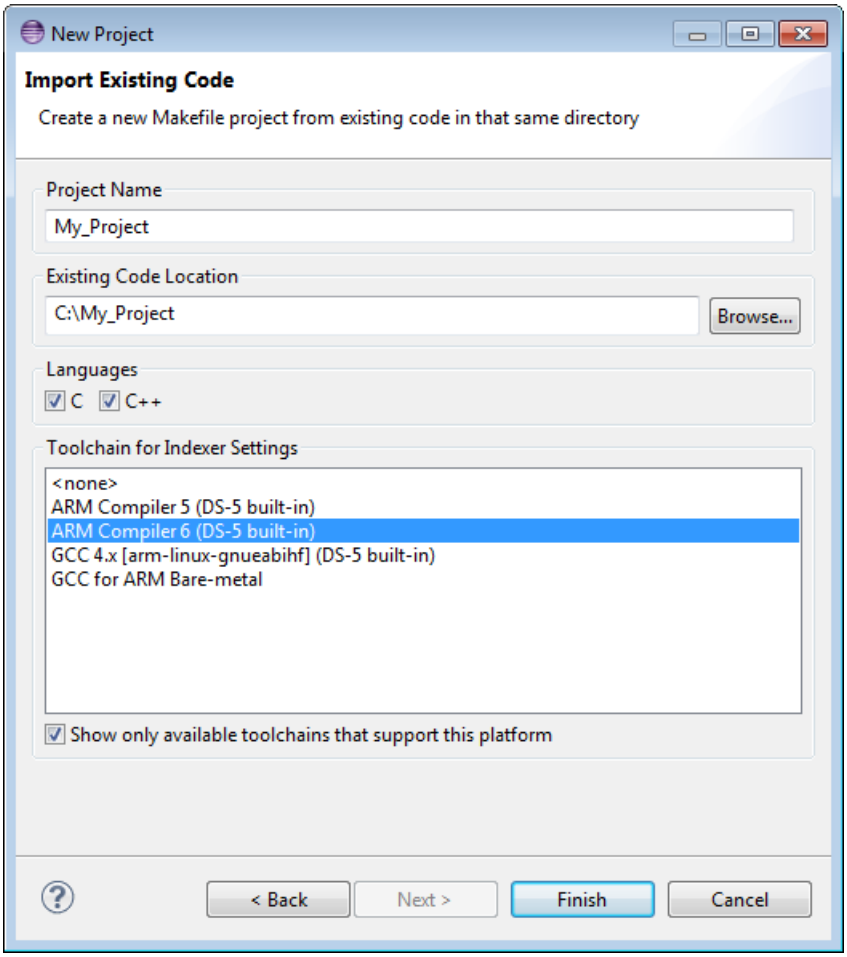

Figure 2-2 Creating a new Makefile project with existing code

7. Click on **Finish** to create your new project. The project and source files are visible in the **Project Explorer** view.
8. Before you can build the project, a `Makefile`, containing all the compilation tool settings, is required. The easiest way to create one is to copy the `Makefile` from an example project, and paste it into your new project.
9. Edit the `Makefile` for your new project.

10. Right-click the project and then select **Properties** > **C/C++ Build** to access the build settings. In the **Builder Settings** tab, ensure that the **Build directory** points to the location of the `Makefile`.
11. Add any other source files you need to the project.

Postrequisites

You can now build the new project:

1. In the Project Explorer view, select the project that you want to build.
2. Select **Project** > **Build Project** from the main menu.

2.5 Importing an existing Eclipse project

If you have an existing Eclipse project, you can import it into your workspace.

To import an existing Eclipse project into your workspace:

Procedure

1. Select **Import...** from the **File** menu.
2. To import an existing project, select **Existing Project into Workspace**. Click on **Next**.

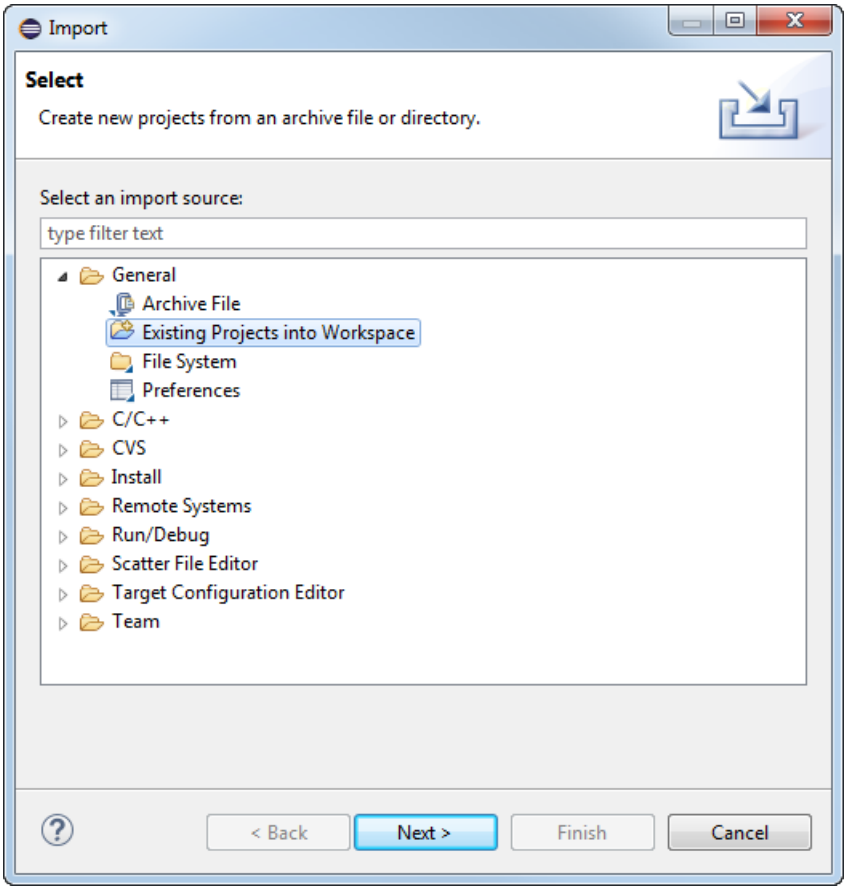

Figure 2-3 Selecting the import source type

3. Click on **Browse** to select the import folder containing the project that you want to import.
4. In the Projects panel select the project that you want to import.
5. Select **Copy projects into workspace** if required or deselect to create links to your existing project(s) and associated files.
6. If you are not using working sets to group your projects then you can skip this step.
   a. Select **Add project to working sets**.
   b. Click on **Select...**.
   c. Select an existing working set or create a new one and then select it.
   d. Click **OK**.
7. Click on **Finish**.

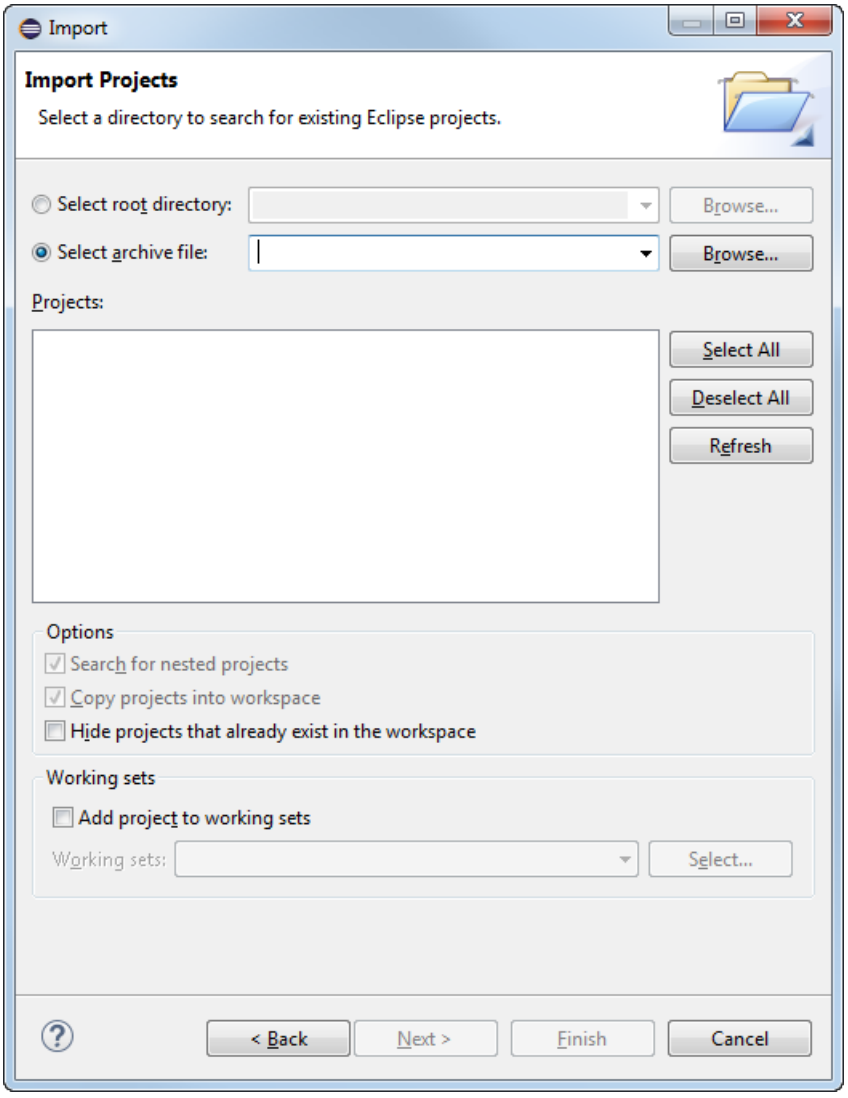

Figure 2-4 Selecting an existing Eclipse projects for import

Note

If your existing project contains project settings from an older version of the build system, you are given the option to update your project. Using the latest version means that you can access all the latest toolchain features.

The imported project is visible in the Project Explorer view.

2.6 Setting up the compilation tools for a specific build configuration

The C/C++ Build configuration panels enable you to set up the compilation tools for a specific build configuration, in your project. The settings in these panels determine how the compilation tools build an ARM executable image or library.

To access the build configuration panels:

Procedure

1. Select the source file or project in the Project Explorer view.
2. Select **Properties** from the **Project** menu.
3. Expand **C/C++ Build** in the Properties dialog box.
4. Select **Settings**.
5. The Configuration panel shows the active configuration. If required, click **Manage Configurations...** from the Configuration panel to create a new build configuration or change the active setting.
6. The compilation tools available for the current project, and their respective build configuration panels are displayed in the **Tool Settings** tab. Click on this tab and configure the build as required.

   ——— **Note** ———

   Makefile projects do not use these configuration panels. The Makefile must contain all the required compilation tool settings.

   ———————————

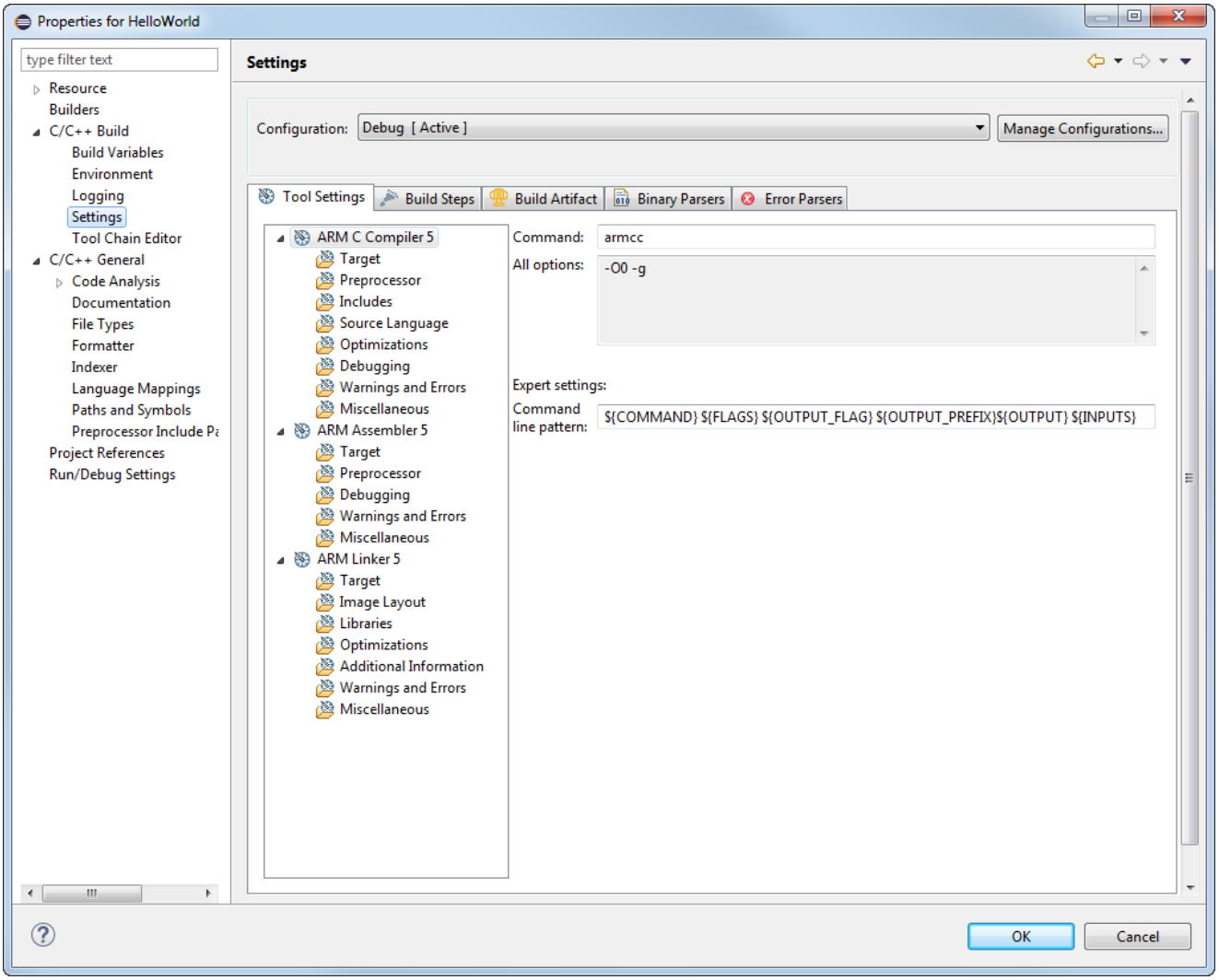

Figure 2-5 Typical build settings dialog box for a C project

7. Click **OK** to save the settings.

2.7 Configuring the C/C++ build behavior

A build is the process of compiling and linking source files to generate an output file. A build can be applied to either a specific set of projects or the entire workspace. It is not possible to build an individual file or sub-folder.

Eclipse provides an incremental build that applies the selected build configuration to resources that have changed since the last build. Another type of build is the Clean build that applies the selected build configuration to all resources, discarding any previous build states.

Automatic

This is an incremental build that operates over the entire workspace and can run automatically when a resource is saved. This setting must be enabled for each project by selecting **Build on resource save (Auto build)** in the **Behaviour** tab. By default, this behavior is not selected for any project.

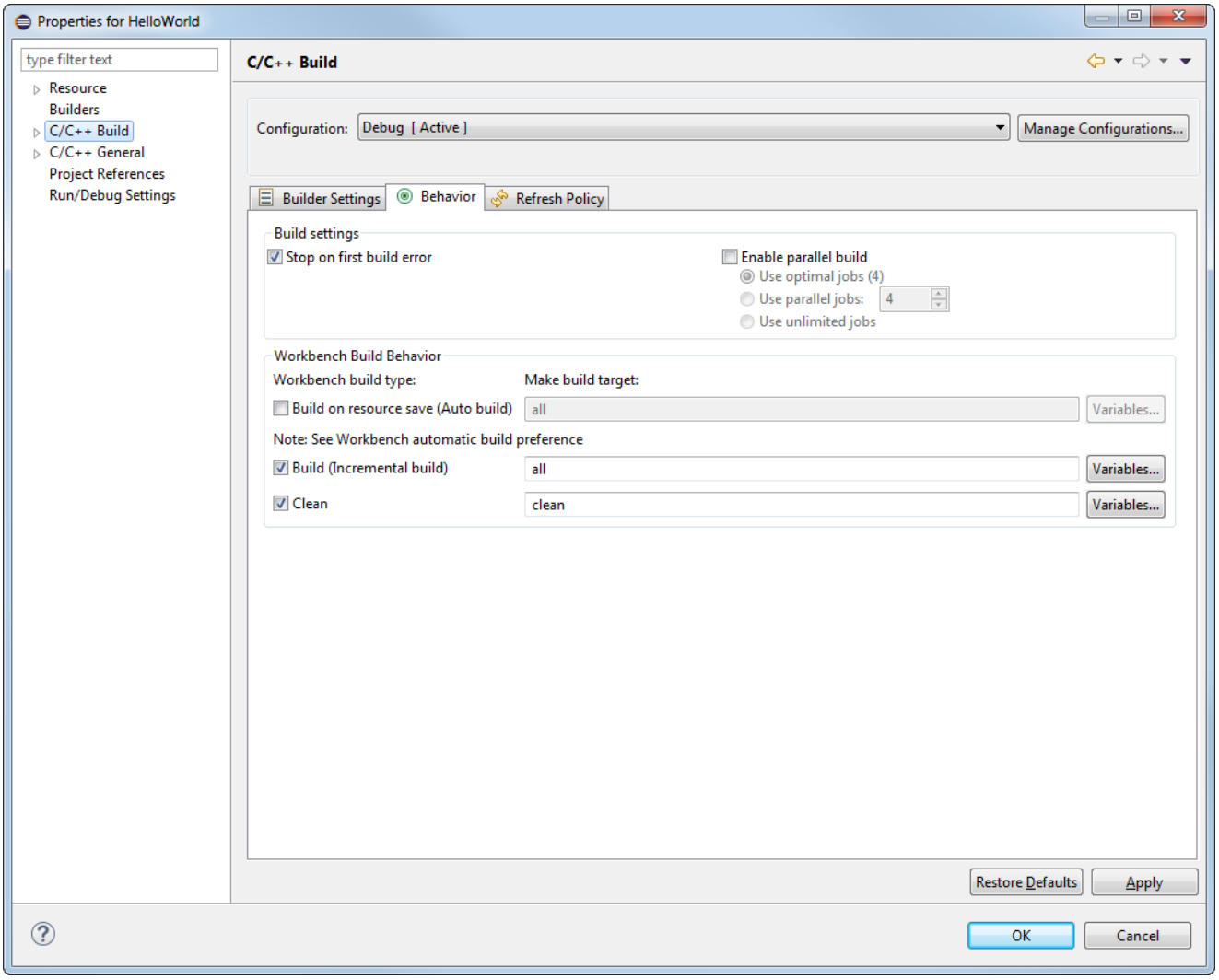

Figure 2-6 Workbench build behavior

You must also ensure that **Build Automatically** is selected from the **Project** menu. By default, this menu option is selected.

Manual

This is an incremental build that operates over the entire workspace on projects with **Build (Incremental build)** selected. By default, this behavior is selected for all projects.
You can run an incremental build by selecting **Build All** or **Build Project** from the **Project** menu.

——— **Note** ———

Manual builds do not save before running so you must save all related files before selecting this option! To save automatically before building, you can change your default settings by selecting **Preferences...** > **General** > **Workspace** from the **Window** menu.

———————————

Clean

This option discards any previous build states including object files and images from the selected projects. The next automatic or manual build after a clean, applies the selected build configuration to all resources.

You can run a clean build on either the entire workspace or specific projects by selecting **Clean...** from the **Project** menu. You must also ensure that **Clean** is selected in the **C/C++ Build** > **Behaviour** tab of the Preferences dialog box. By default, this behavior is selected for all projects.

Build order is a feature where inter-project dependencies are created and a specific build order is defined. For example, an image might require several object files to be built in a specific order. To do this, you must split your object files into separate smaller projects, reference them within a larger project to ensure they are built before the larger project. Build order can also be applied to the referenced projects.

2.8 Using Eclipse from the command-line to clean and build your projects

You can run Eclipse from the command-line to clean and build your projects. This might be useful, for example, when you want to create scripts to automate build procedures.

Procedure

1. Launch a DS-5 command-line console to load Eclipse, *make*, and other utilities on your *PATH* environment variable. To do this:
   - On Windows, select **Start** > **All Programs** > **ARM DS-5** > **DS-5 Command Prompt**.
   - On Linux, run *DS-5_install_directory*/bin/suite_exec <shell> to open a shell.
2. Run `eclipsec.exe` (on Windows) or `eclipse` (on Linux) with the following Eclipse arguments as required.

Table 2-1 Eclipse arguments

Argument	Description
`-nosplash`	Disables the Eclipse splash screen.
`--launcher.suppressErrors`	Causes errors to be printed to the console instead of being reported in a graphical dialog.
`-application org.eclipse.cdt.managedbuilder.core.headlessbuild`	Mandatory argument telling Eclipse to run the headless builder.
`-data {workspaceDir}`	Specify the location of your workspace.
`-import {projectDir}`	Import the project from the specified directory into your workspace. Use this option multiple times to import multiple projects.
`-build {projectName[/configName] | all}`	Build the project with the specified name, or all projects in your workspace. By default, this argument builds all the configurations within each project. You can limit this action to a single configuration, such as `Debug` or `Release`, by specifying the configuration name immediately after your project name, separated with '`/`'. Use this option multiple times to build multiple projects.
`-cleanBuild {projectName[/configName] | all}`	Clean and build the project with the specified name, or all projects in your workspace. By default, this argument cleans and builds all the configurations within each project. You can limit this action to a single configuration, such as `Debug` or `Release`, by specifying the configuration name immediately after your project name, separated with '`/`'. Use this option multiple times to clean and build multiple projects.

Example 2-1 Examples

To list and view the full set of available options, use the command:

```
eclipsec.exe -nosplash -application
org.eclipse.cdt.managedbuilder.core.headlessbuild
```

To clean and build all the projects in a workspace at the `C:\workspace` location, use the command:

```
eclipsec.exe -nosplash -application
org.eclipse.cdt.managedbuilder.core.headlessbuild -data C:\workspace
-cleanBuild
```

To build the Release configuration of project *`MyProject`* in workspace `C:\workspace`, use the command:

```
eclipsec.exe -nosplash -application
org.eclipse.cdt.managedbuilder.core.headlessbuild -data C:\workspace -build
MyProject/Release
```

2.9 Updating a project to a new toolchain

If you have several products installed, only the latest toolchain is listed in the New Project wizard. Therefore, if you have projects that use an older toolchain, you must update them to the latest toolchain.

To update them to use the new toolchain:

Procedure

1. Right-click on the project in the Project Explorer view.
2. Select **Convert To...** from the context menu to display the Project Converters dialog box.
3. Select the required toolchain.

   ——— **Note** ———

   Beware that if you convert back to an earlier version you might lose toolchain functionality.

   ———————————

4. Click **OK**.

2.10 Adding a new source file to your project

You can add a new source file to your project.

To do this:

Procedure

1. Select **File** > **New** > **DS-5 Debugger script** from the main menu.
2. Click **Workspace** or **File System...** in the **Location** field to add the project folder or sub-folder path in the **New Script** dialog box.

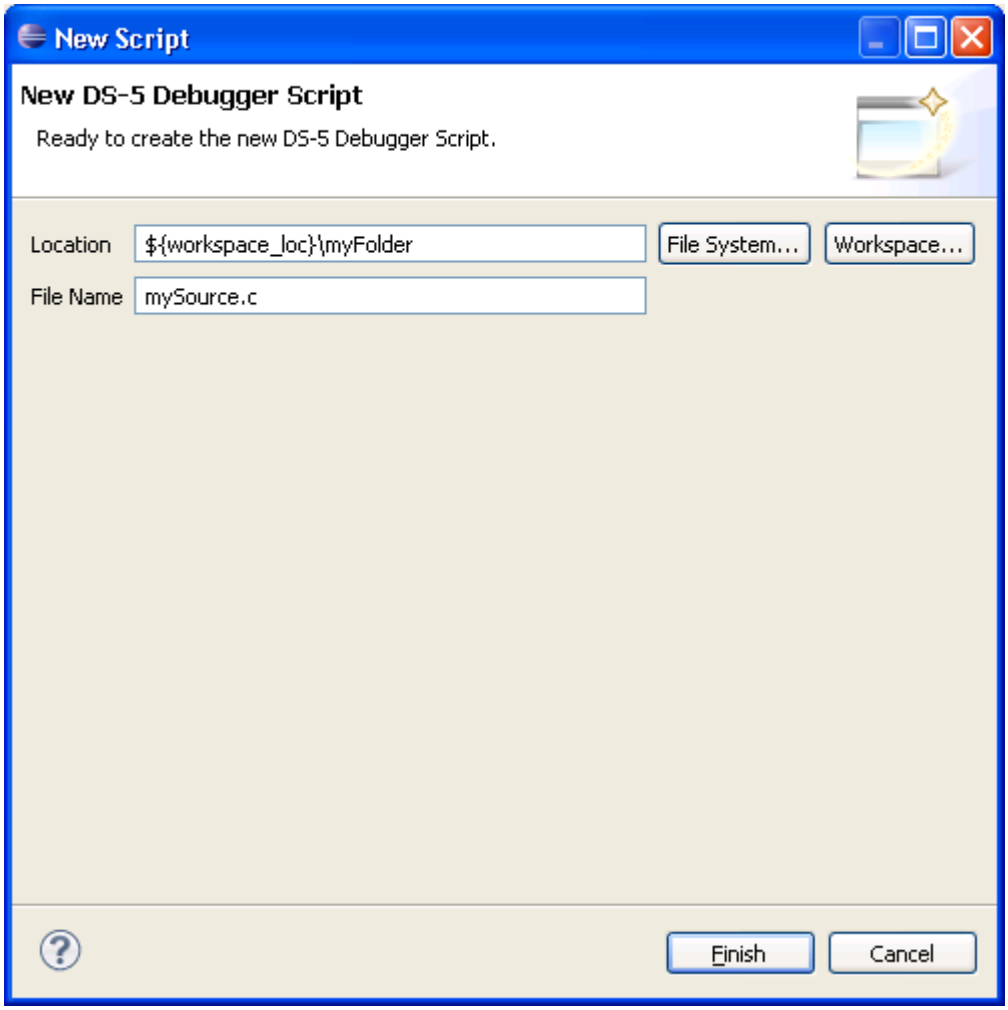

Figure 2-7 Adding a new file to your project

3. Enter a filename with the relevant extension in the **File Name** field.
4. Click on **Finish**.

   The new source file is visible in the **Project Explorer** view.

   - You can also create files, or drag and drop files directly into the project folder, using the file system. To update the views in Eclipse, click on the relevant project in the **Project Explorer** view and select **Refresh** from the **File** menu.
   - You can also drag and drop files directly into a project folder, in the **Project Explorer** view of Eclipse.

2.11 Sharing Eclipse projects

You can share Eclipse projects between users if necessary.

Note

- There are many different ways to share projects and files, for example, using a source control tool. This topic covers the general principles of sharing projects and files using Eclipse, and not the specifics of any particular tool.
- To share files, it is recommended to do so at the level of the project and not the workspace. Your source files within Eclipse are organized into projects, and projects exist within your workspace. An Eclipse workspace contains many files, including files in the `.metadata` directory, that are specific to an individual user or installation.

Within each project, the files that must be shared beyond just your source code are:

- `.project` - Contains general information about the project type, and the Eclipse plug-ins to use to edit and build the project.
- `.cproject` - Contains C/C++ specific information, including compiler settings.

Eclipse places built files into the project directory, including auto-generated makefiles, object files, and image files. Not all files have to be shared. For example, sharing an auto-generated makefile might be useful to allow building the project outside of Eclipse, but if projects are only built within Eclipse then this is not necessary.

You must be careful when creating and configuring projects to avoid hard-coded references to tools and files outside of Eclipse that might differ between users.

To ensure that files outside of Eclipse can be referenced in a user agnostic way, use the *`${workspace_loc}`* built-in variable or custom environment variables.

Chapter 3
Working with editors

The following topics describe how to use the editors when developing a project for an ARM target.

It contains the following sections:

3.1 About the C/C++ editor

The standard C/C++ editor is provided by the CDT plug-in that provides C and C++ extensions to Eclipse. It provides syntax highlighting, formatting of code and content assistance when editing C/C++ code.

Embedded assembler in C/C++ files is supported by the ARM Compiler but this editor does not support it and so an error is displayed. This type of code is ARM-specific and accepted Eclipse behavior so you can ignore the syntax error.

If this is not the default editor, right-click on a source file in the Project Explorer view and select **Open With** > **C/C++ Editor** from the context menu.

See the *C/C++ Development User Guide* for more information. Select **Help** > **Help Contents** from the main menu.

3.2 About the ARM assembler editor

The ARM assembler editor provides syntax highlighting, formatting of code and content assistance for labels in ARM assembly language source files. You can change the default settings in the Preferences dialog box.

If this is not the default editor, right-click on your source file in the Project Explorer view and select **Open With** > **ARM Assembler Editor** from the context menu.

The following shortcuts are also available for use:

Table 3-1 ARM assembler editor shortcuts

Content assist	Content assist provides auto-completion on labels existing in the active file. When entering a label for a branch instruction, Partially type the label and then use the keyboard shortcut Ctrl+Space to display a list of valid auto-complete options. Use the Arrow Keys to select the required label and press Enter to complete the term. Continue typing to ignore the auto-complete list.
Editor focus	The following options change the editor focus: • Outline View provides a list of all areas and labels in the active file. Click on an area or label to move the focus of the editor to the position of the selected item. • Select a label from a branch instruction and press F3 to move the focus of the editor to the position of the selected label.
Formatter activation	Press Ctrl+Shift+F to activate the formatter settings.
Block comments	Block comments are enabled or disabled by using Ctrl+Semicolon. Select a block of code and apply the keyboard shortcut to change the commenting state.

3.3 About the ELF content editor

The ELF content editor creates forms for the selected ELF file. You can use this editor to view the contents of image files and object files. The editor is read-only and cannot be used to modify the contents of any files.

If this is not the default editor, right-click on your source file in the Project Explorer view and select **Open With** > **ELF Content Editor** from the context menu.

The ELF content editor displays one or more of the following tabs depending on the selected file type:

Header
Form view showing the header information.

Sections
Tabular view showing the breakdown of all section information.

Segments
Tabular view showing the breakdown of all segment information.

Symbol Table
Tabular view showing the breakdown of all symbols.

Disassembly
Textual view of the disassembly with syntax highlighting.

3.4 ELF content editor - Header tab

The **Header** tab provides a form view of the ELF header information.

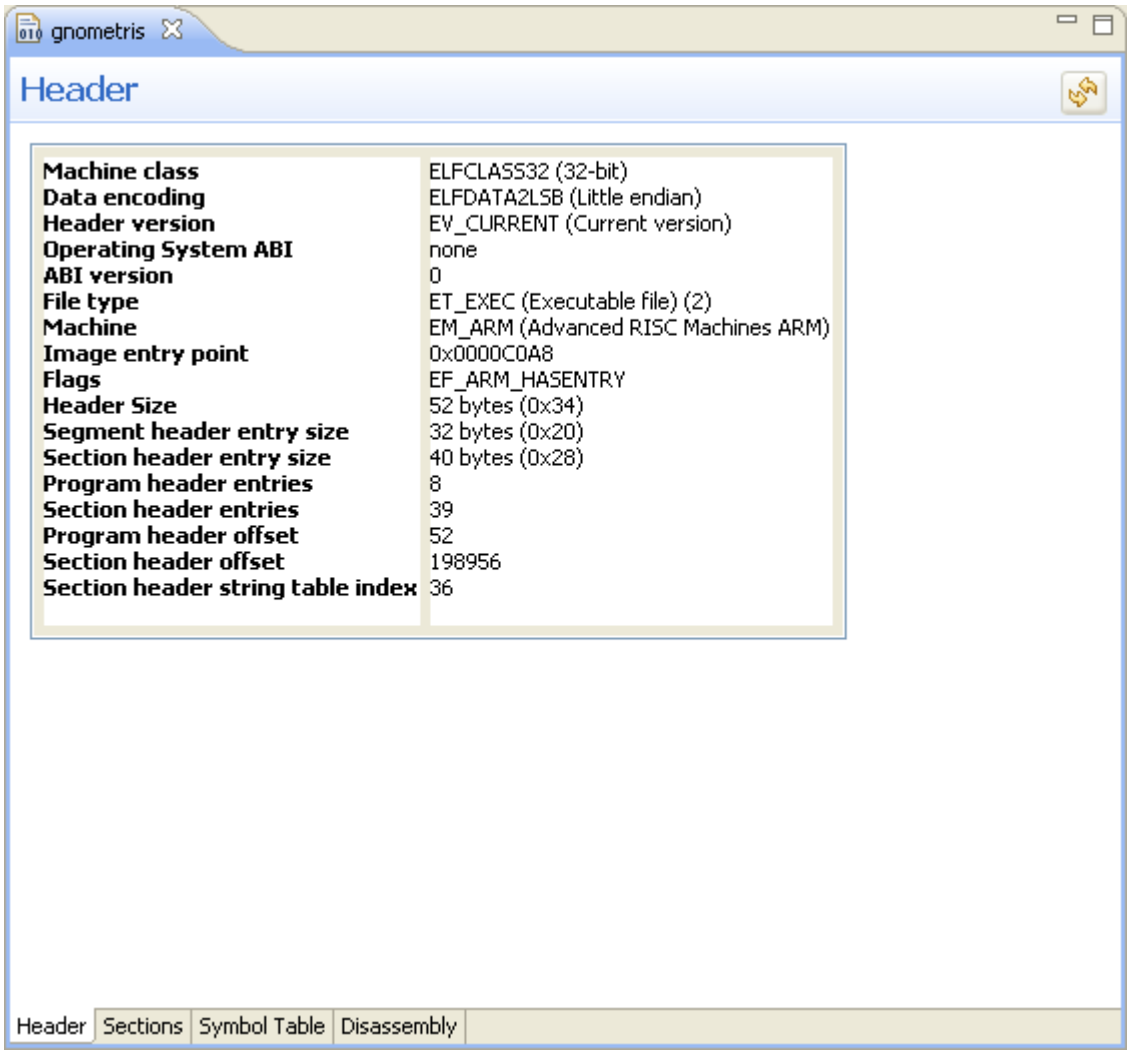

Figure 3-1 Header tab

3.5 ELF content editor - Sections tab

The **Sections** tab provides a tabular view of the ELF section information.

To sort the columns click on the column headers.

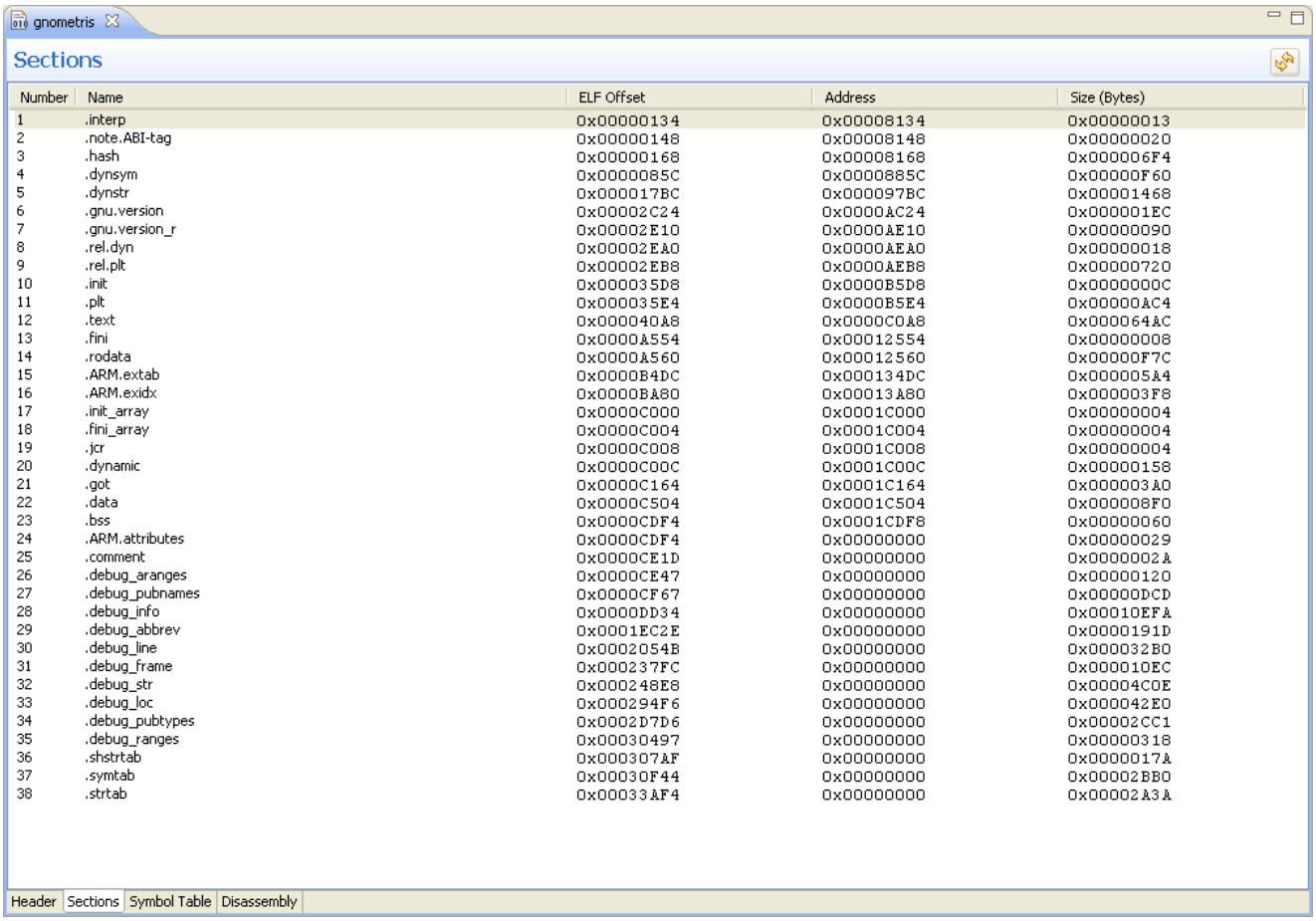

Figure 3-2 Sections tab

3.6 ELF content editor - Segments tab

The **Segments** tab provides a tabular view of the ELF segment information.

To sort the columns click on the column headers.

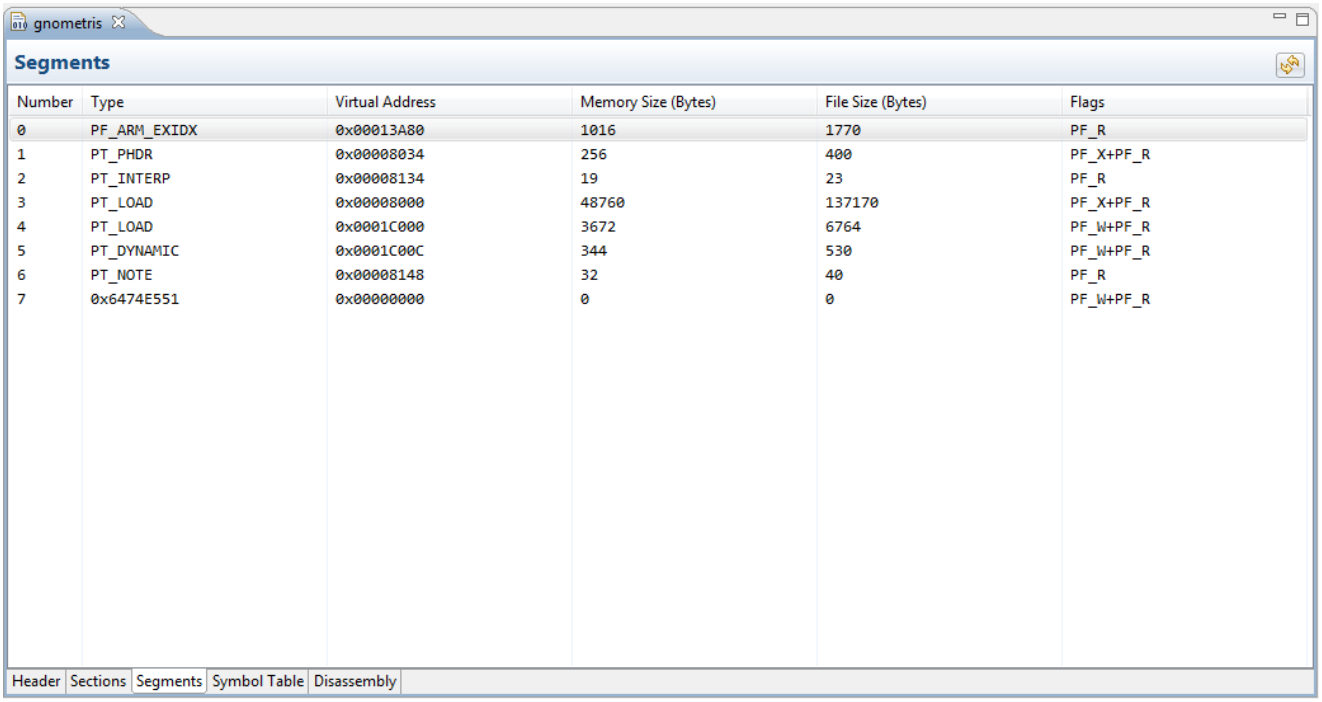

Figure 3-3 Segments tab

3.7 ELF content editor - Symbol Table tab

The **Symbol Table** tab provides a tabular view of the symbols.

To sort the columns click on the column headers.

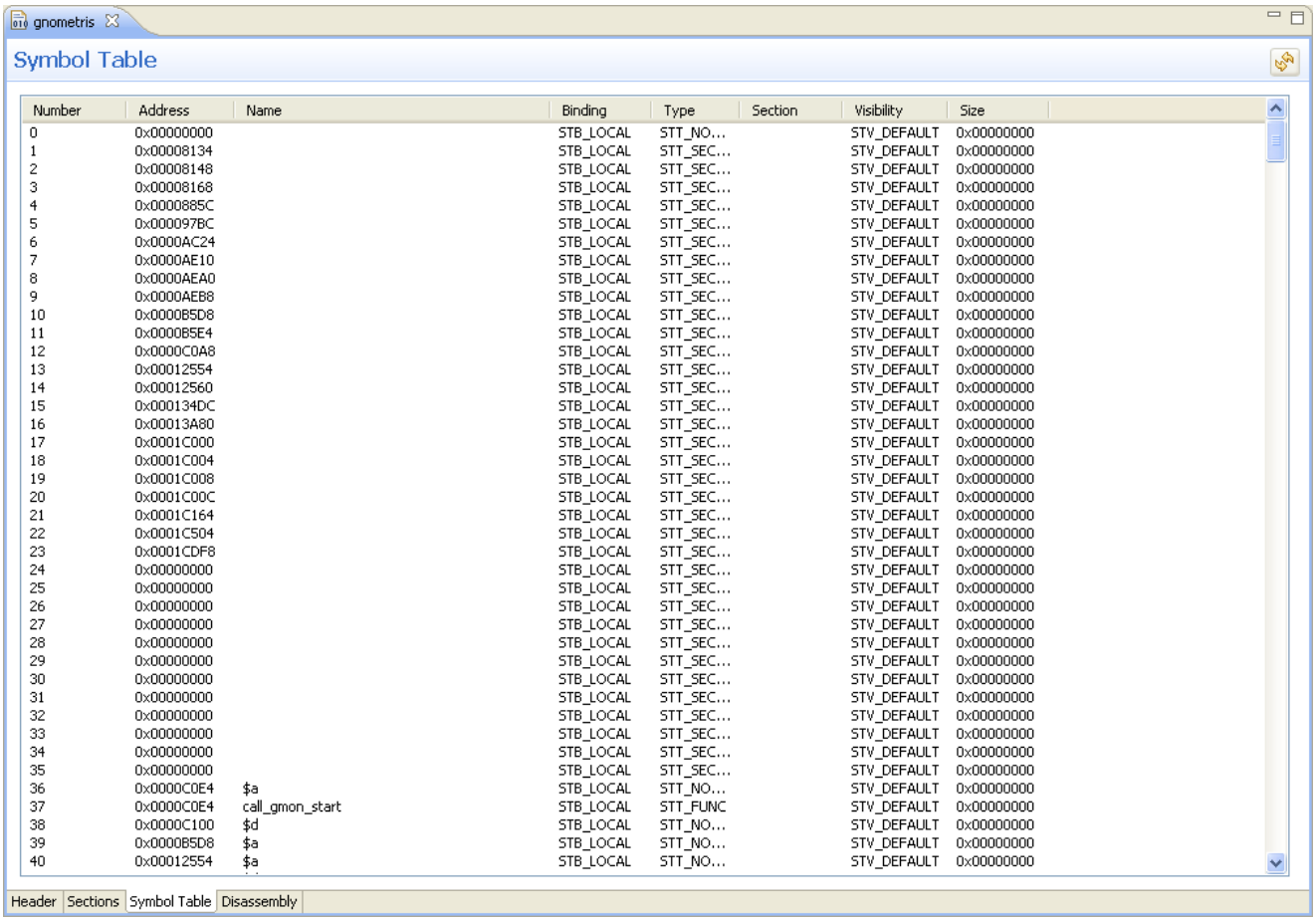

Figure 3-4 Symbol Table tab

3.8 ELF content editor - Disassembly tab

The **Disassembly** tab displays the output with syntax highlighting. The color schemes and syntax preferences use the same settings as the ARM assembler editor.

There are several keyboard combinations that can be used to navigate around the output:

- Use Ctrl+F to open the **Find** dialog box to search the output.
- Use Ctrl+Home to move the focus to the beginning of the output.
- Use Ctrl+End to move the focus to the end of the output.
- Use Page Up and Page Down to navigate through the output one page at a time.

You can also use the **Copy** and **Find** options in the context menu by right-clicking in the **Disassembly** view.

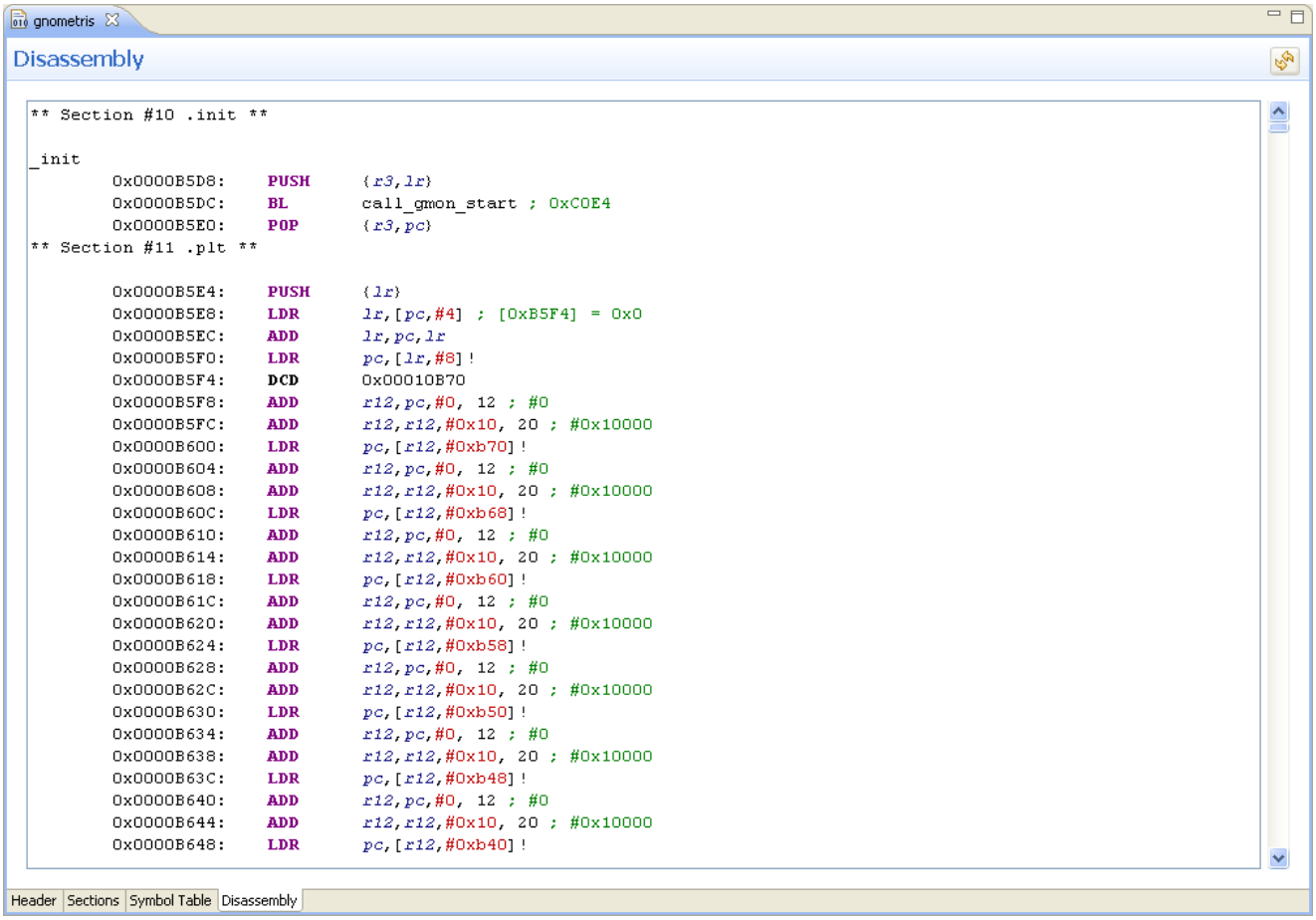

Figure 3-5 Disassembly tab

3.9 About the scatter file editor

The scatter file editor enables you to easily create and edit scatter files for use with the ARM linker to construct the memory map of an image.

It provides a text editor, a hierarchical tree and a graphical view of the regions and output sections of an image. You can change the default syntax formatting and color schemes in the **Preferences** dialog box.

If this is not the default editor, right-click on your source file in the **Project Explorer** view and select **Open With** > **Scatter File Editor** from the context menu.

The scatter file editor displays the following tabs:

Source
: Textual view of the source code with syntax highlighting and formatting.

Regions/Sections
: A graphical view showing load and execute memory maps. These are not editable, however, you can select a load region to show the related memory blocks in the execution regions.

The scatter file editor also provides a hierarchical tree with associated toolbar and context menus using the **Outline** view. Clicking on a region or section in the **Outline** view moves the focus of the editor to the relevant position in your code. If this view is not visible, from the **Window** menu, select **Show View** > **Outline**.

——— **Note** ———

The linker documentation for ARM Compiler describes in more detail how to use scatter files.

———————————

Before you can use a scatter file you must add the `--scatter=file` option to the project within the **C/C++ Build** > **Settings** > **Tool settings** > **ARM Linker** > **Image Layout** panel of the **Properties** dialog box.

3.10 Creating a scatter file

Create a scatter file to specify more complex memory maps that cannot be specified using compiler command-line memory map options.

To create a scatter file:

Procedure

1. Use an existing project or create a new project.
2. Create a new project and add a new empty text file with the extension `.scat`. For example `scatter.scat`.
3. Using the **Outline** view, click on the **Add load region** toolbar icon or right-click and select **Add load region** from the context menu.

   Enter a load region name, for example, `LR1`.

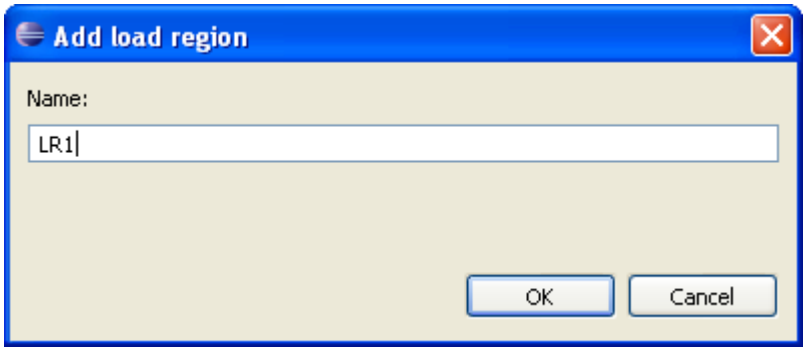

Figure 3-6 Add load region dialog box

4. Click **OK**.
5. Modify the load region as shown in the following example.

   Simple scatter file

```
LR1 0x0000 0x8000
{
    LR1_er1 0x0000 0x8000
    {
        * (+RO)
    }
    LR1_er2 0x10000 0x6000
    {
        * (+RW,+ZI)
    }
}
```

6. Click on the **Regions/Sections** tab to view a graphical representation.

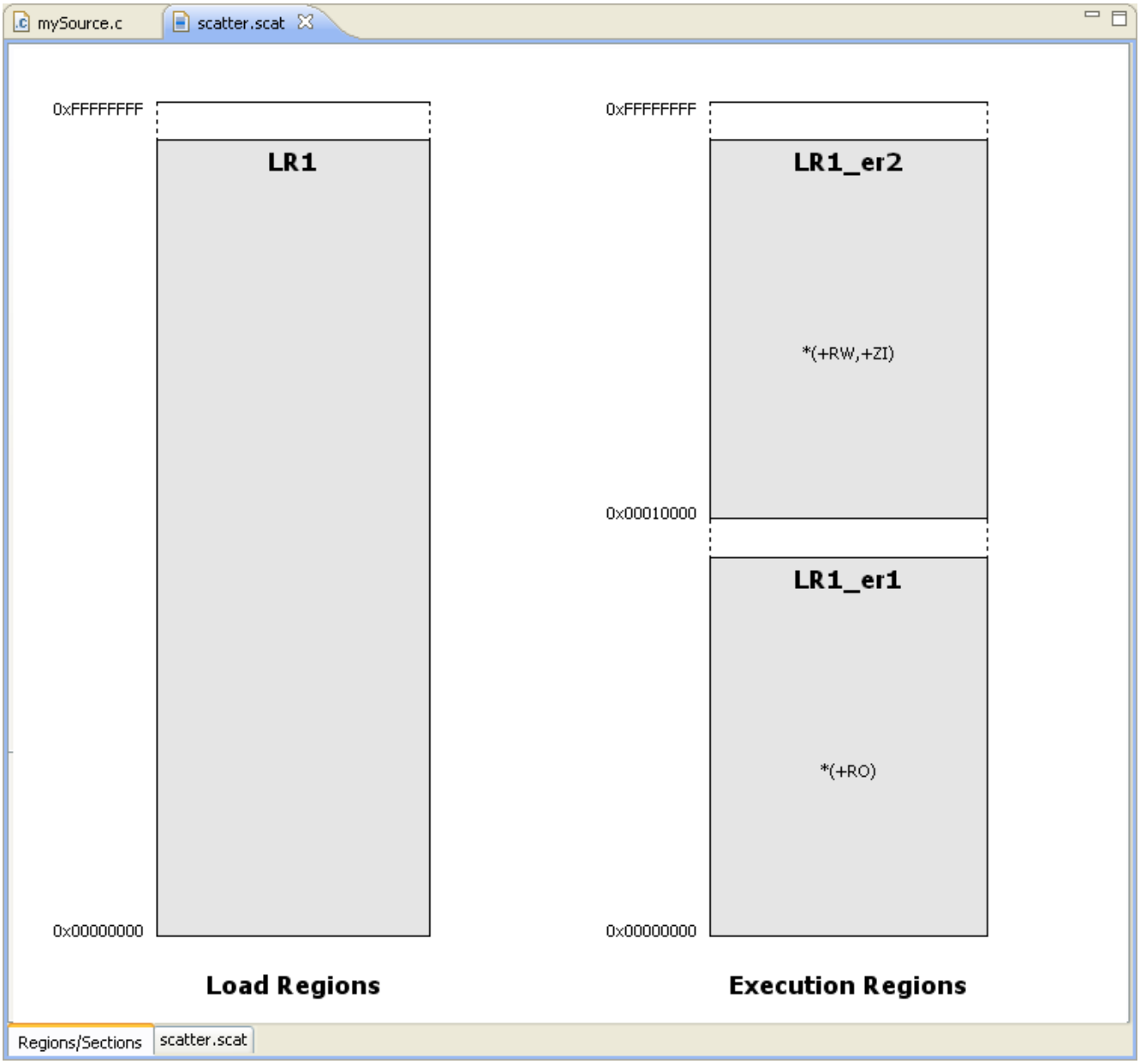

Figure 3-7 Graphical view of a simple scatter file

7. Save your changes.

Note

Before you can use a scatter file you must add the *--scatter=file* option to the project within the **C/C++ Build** > **Settings** > **Tool settings** > **ARM Linker** > **Image Layout** panel of the **Properties** dialog box.

3.11 Importing a memory map from a BCD file

If you have a BCD file that defines a memory map, you can import this into the **Scatter file** editor.

To import a memory map from a BCD file:

Procedure

1. Select **Import** from the **File** menu.
   Select **Scatter File Editor** > **Memory from a BCD File**.

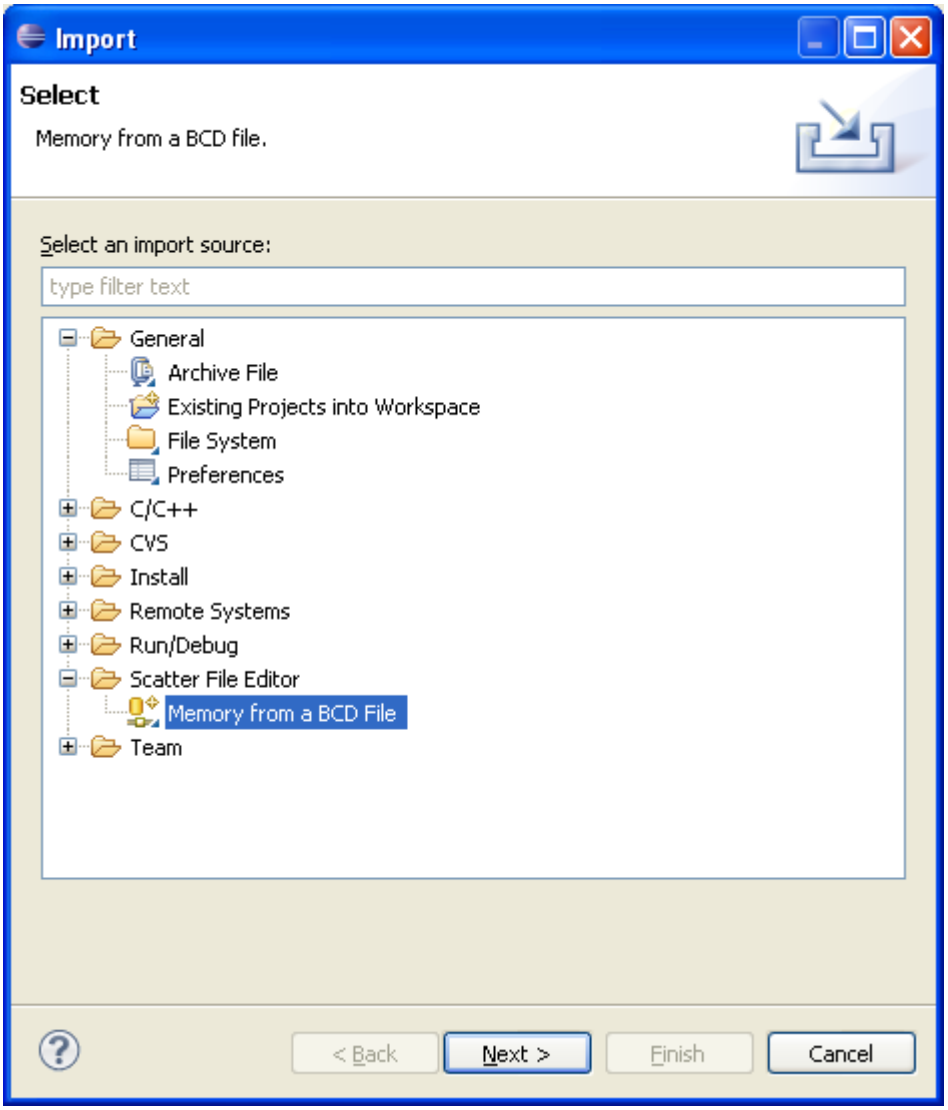

Figure 3-8 Import memory map for the scatter file editor

2. Enter the location of the BCD file or click on **Browse...** to select the folder.
3. Select the required file containing the memory map that you want to import.

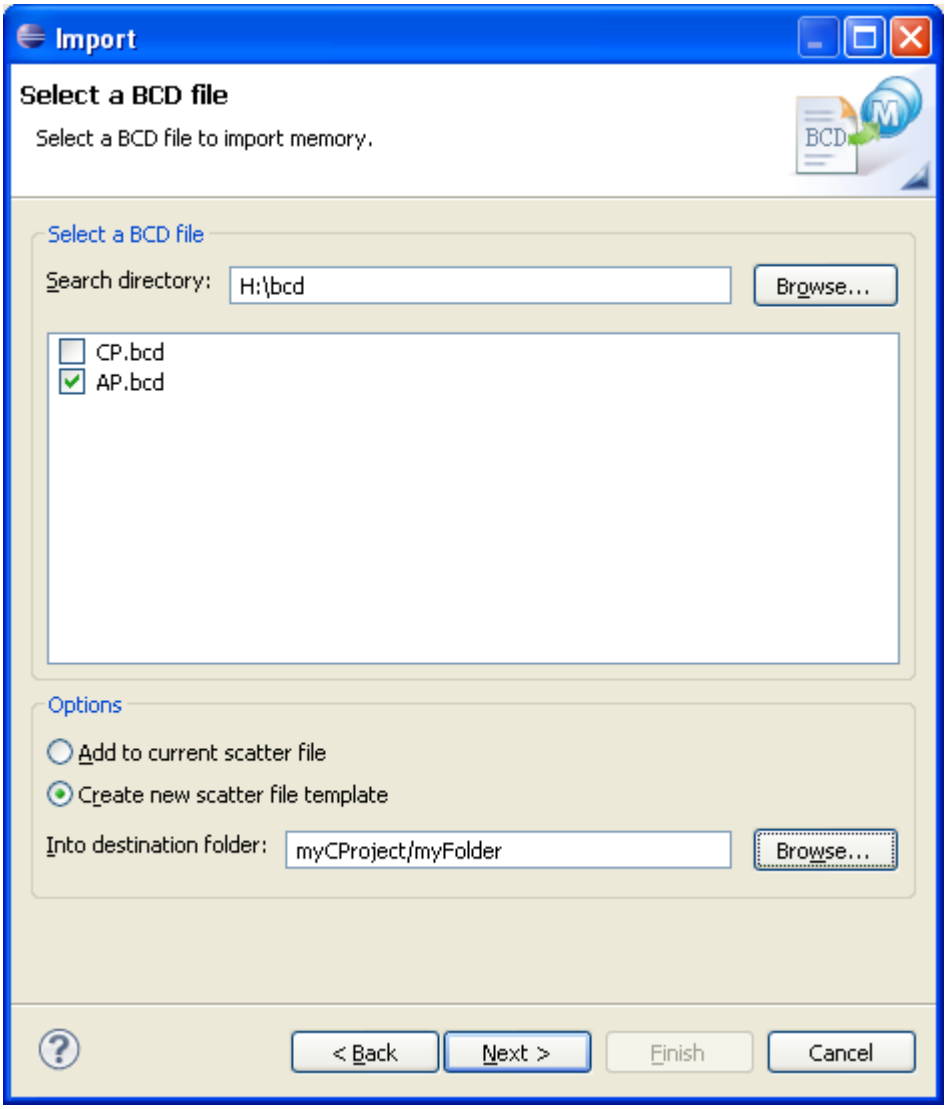

Figure 3-9 BCD file selection for the scatter file editor

4. Select **Add to current scatter file** if you want to add specific memory regions to an existing scatter file.

   ——— **Note** ———

   The scatter file must be open and active in the editor view before you can use this option.

   ———————————

5. Select **Create new scatter file template** if you want the wizard to create a new file with the same name as the BCD file but with a `.scat` file extension.
6. Select the destination project folder.
7. By default, all the memory regions are selected. Modify the selections and table content as required.

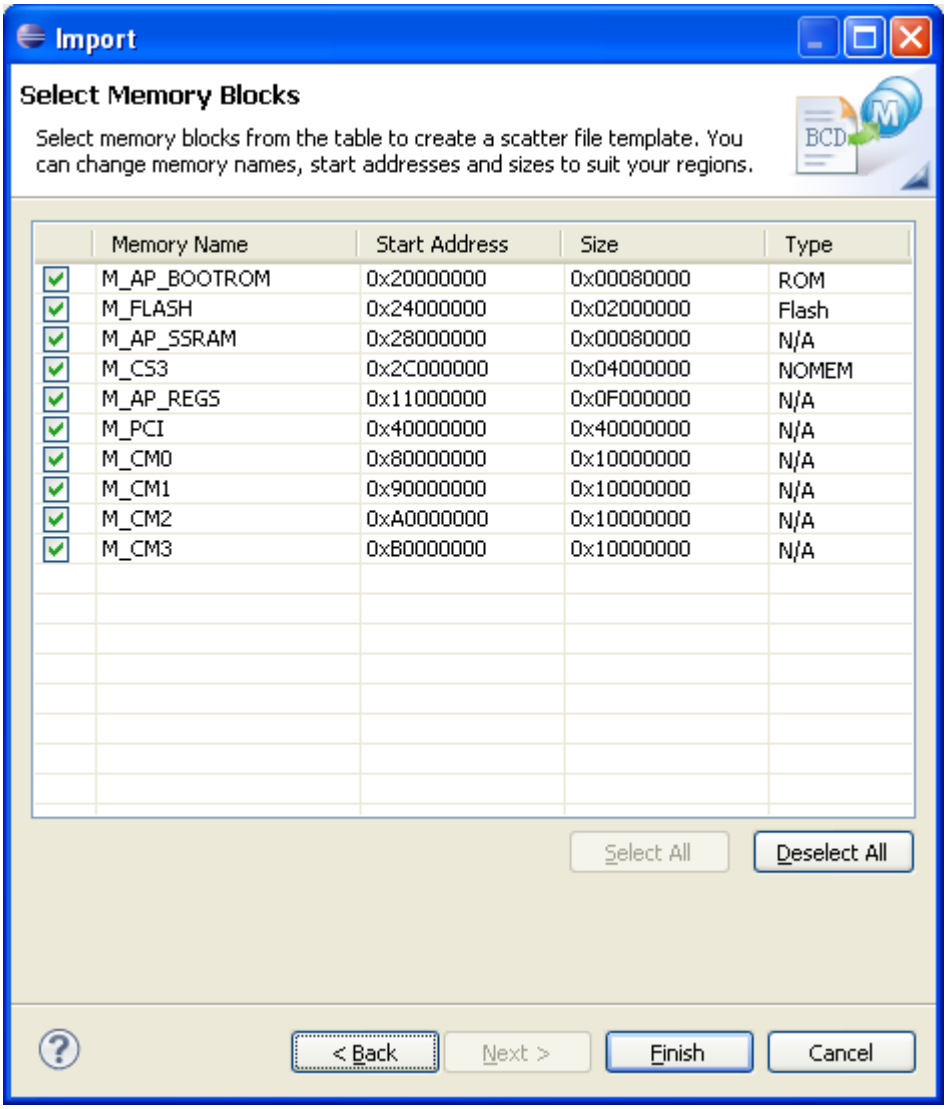

Figure 3-10 Memory block selection for the scatter file editor

8. Click **Finish** to complete the scatter file.

——— **Note** ———

Before you can use a scatter file you must add the *`--scatter=file`* option to the project within the **C/C++ Build** > **Settings** > **Tool settings** > **ARM Linker** > **Image Layout** panel of the **Properties** dialog box.

———————————

Chapter 4
Terminology, shortcuts and icons

This following topics describe some of the terminology used in Eclipse for DS-5 User Guide, useful keyboard shortcuts, and menu and toolbar icons.

It contains the following sections:

4.1 Terminology

A list of the terms used in this document.

Block
A small sub-division of a flash device that can be programmed.

Device
A component on a target containing the application that you want to debug.

Dialog box
A small page containing tabs, panels and editable fields prompting you to enter information.

Editor
A view that controls the visual aspects of source code for a specific file type.

Erase
A feature of a flash device where memory cells are reset to a known value.

Flash device
A set of flash memory that has a single command interface.

Panel
A small area in a dialog box or tab to group editable fields.

Perspective
A page within the Eclipse window containing a set of related views, editors, menus, and toolbars.

Program
A term used to describe the storing of data on a flash device.

Project
A group of related files and folders in Eclipse.

Resource
A generic term used to describe a project, file, folder, or a combination of these.

Send To
A term used to describe sending a file to a target.

Tab
A small overlay page containing panels and editable fields within a dialog box to group related information. Clicking on a tab brings it to the top.

Target
A development platform on a printed circuit board or a software model that emulates the expected behavior of ARM hardware.

View
A small page to display related information for a specific function.

Width
The smallest number of bits (8, 16 or 32) that can be natively accessed by a flash device.

Wizard
A group of dialog boxes to guide you through a common tasks, for example, creating new files and projects.

Workbench
A window containing perspectives, menus, and toolbars.

Workspace
An area designated on your file system to store files and folders related to your projects.

4.2 Keyboard shortcuts

A list of the most common keyboard shortcuts available for use with Eclipse.

F3
Click on an assembler label from a branch instruction or a C/C++ calling function and press F3 to move the editor focus to the position of the selected item.

F10
Use in conjunction with the arrow keys to access the main menu.

Alt+F4
Exit Eclipse.

Alt+Left arrow
Go back in navigation history.

Alt+Right arrow
Go forward in navigation history.

Ctrl+Semicolon
Provided with the ARM assembler editor to add comment markers to a selected block of code in the active file.

Ctrl+End
Moves the editor focus to the end of the code.

Ctrl+B
Build all projects in the workspace that have changed since the last build.

Ctrl+Home
Moves the editor focus to the beginning of the code.

Ctrl+F
Opens the Find or Find/Replace dialog box to search through the code in the active editor. Some editors are read-only and therefore disable this functionality.

Ctrl+F4
Close the active file in the editor view.

Ctrl+F6
Cycles through open files in the editor view.

Ctrl+F7
Cycles through available views.

Ctrl+F8
Cycles through available perspectives.

Ctrl+F10
Use in conjunction with the arrow keys to access the drop-down menu.

Ctrl+L
Move to a specified line in the active file.

Ctrl+Q
Move to the last edited position in the active file.

Ctrl+Space
Provides auto-completion on selected functions in editors.

Shift+F10
use in conjunction with the arrow keys to access the context menu.

Ctrl+Shift+F
Activates the code style settings in the Preferences dialog box and apply them to the active file.

Ctrl+Shift+L
Opens a small page with a list of all keyboard shortcuts.

Ctrl+Shift+R
Opens the Open resource dialog box.

Ctrl+Shift+T
Opens the Open Type dialog box.

Ctrl+Shift+/
Provided with the C/C++ editor to add comment markers to the start and end of a selected block of code in the active file.

4.3 Menu and toolbar icons

These are the most common menu and toolbar icons available for use with Eclipse.

For information on icons, markers, and buttons not listed in the following tables, see the standard *Workbench User Guide* or the *C/C++ Development User Guide* in the Help Contents.

If you leave the mouse pointer positioned on a toolbar icon for a few seconds without clicking, a tooltip appears informing you of the purpose of the icon.

Perspective icons

Table 4-1 Perspective icons

Icon	Description	Icon	Description
	Open new perspective		C/C++
	DS-5 Debugger		

View icons

Table 4-2 View icons

Icon	Description	Icon	Description
	Display drop-down menu		Synchronize view contents
	Minimize		Maximize
	Restore		Close

View markers

Table 4-3 View markers

Icon	Description	Icon	Description
	Bookmark		Information
	Task		Search result
	Error		Warning

Editor icons

Table 4-4 Editor icons

Icon	Description	Icon	Description
	Save the active file		Save all files
	Print the active file		Close view
	Create new configuration		Duplicate selected configuration
	Delete selected configuration		Collapse configuration tree

Configuration icons

Table 4-5 Configuration icons

Icon	Description	Icon	Description
	Open Run Configurations dialog box		Open Debug Configurations dialog box

Outline icons

Table 4-6 Outline icons

Icon	Description	Icon	Description
	Hide fields		Hide static members
	Hide non-public members		Sort alphabetically
	Class		Namespace
	Macro definition		Enum
	Enumerator		Variable
	Protected field		Private field
	Public field		Include
	Protected method		Private method
	Public method		Struct
	Type definition		Union
	Function		Target configuration file
	Include file		Memory
	Peripheral		Register
	Bitfield		Enumeration definition
	Map rules		Tools information

Miscellaneous icons

Table 4-7 Miscellaneous icons

Icon	Description	Icon	Description
	Open a new resource wizard		Open new project wizard
	Open new folder wizard		Open new file wizard
	open search dialog box		Display context-sensitive help
	Add load region		Add execution region
	Add section		Delete selected item
	Focus on active task		Displays product updates

Navigation icons

Table 4-8 Navigation icons

Icon	Description	Icon	Description
	Navigate back		Navigate forwards
	Navigate up one level		Print active page
	Open help instruction page		Synchronize TOC with active page
	Bookmark active page		

Help Contents icons

Table 4-9 Help Contents icons

Icon	Description	Icon	Description
	Display list of all documents		Display list of documents in last search
	Display list of context-sensitive help links to related topics		Display list of all bookmarks
	Maximize frame		Restore frame
	Synchronize TOC with active page		