

CodeWarrior Development Studio for Microcontrollers V10.x Getting Started Guide

Revised: June 9, 2010



Freescale, the Freescale logo, CodeWarrior and ColdFire are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Flexis and Processor Expert are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2010 Freescale Semiconductor, Inc. All rights reserved.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

How to Contact Us

Corporate Headquarters	Freescale Semiconductor, Inc. 6501 William Cannon Drive West Austin, Texas 78735 U.S.A.
World Wide Web	http://www.freescale.com/codewarrior
Technical Support	http://www.freescale.com/support

Table of Contents

1	Introduction	5
	Overview of this Manual.	5
	Additional Information Resources	6
2	Overview	13
	Introduction to Microcontrollers 10.0.	13
	Introduction to Eclipse IDE	13
	System Requirements	14
3	CodeWarrior Development Process	15
	CodeWarrior Development Process	15
	Project Files.	15
	Editing Code	16
	Compiling	16
	Linking	16
	Debugging.	16
4	Using Microcontrollers Workbench	19
	Workspace.	19
	Workbench	20
	Welcome Page.	22
	Help Documentation.	25
	Eclipse Help	25
	Cheat Sheets	26
	Perspective and View	28
	C/C++ Perspective.	28
	Debug Perspective.	30
	Working with Views	32
	Working with Perspectives	35
	Editor	37
	Opening Files for Editing	37
	Working with Markers in Editor Area.	38

Table of Contents

Toolbars	39
Workbench Toolbar	40
View Toolbar	43
Perspective Switcher	43
Trim Stack	44
Fast View Toolbar	45
5 Creating and Debugging Projects	47
Creating Projects	47
Creating New Microcontrollers Bareboard Project	47
Debugging Projects	55
Breakpoints	58
Watchpoints	60
Index	63

Introduction

This manual introduces you to the interface of CodeWarrior for Microcontrollers V10.x. It describes basic components of the Microcontrollers 10.x IDE and the CodeWarrior development process. This manual also describes how to work with projects in Microcontrollers 10.x.

This chapter consists of these topics.

- [Overview of this Manual](#) — Describes the contents of this manual
- [Additional Information Resources](#) — Describes supplementary CodeWarrior documentation, third-party documentation, and references to helpful code examples and web sites

Overview of this Manual

[Table 1.1](#) describes each chapter in this manual.

Table 1.1 Chapter Contents

Chapter	Description
Overview	Provides an overview of Microcontrollers 10.0 and Eclipse IDE. It also provides system requirements for installing Microcontrollers 10.x.
CodeWarrior Development Process	Describes stages involved in developing an application using CodeWarrior IDE.
Using Microcontrollers Workbench	Provides an overview of the Microcontrollers 10.x interface and its components, and describes how to work with the different components of the Eclipse interface.
Creating and Debugging Projects	Describes how to create and debug projects in Microcontrollers 10.x.

Additional Information Resources

- For Freescale documentation and resources, visit the Freescale web site:
<http://www.freescale.com>
- For additional electronic-design and embedded-system resources, visit the EG3 Communications, Inc. web site: <http://www.eg3.com>
- For monthly and weekly forum information about programming embedded systems (including source-code examples), visit the Embedded Systems Programming magazine web site: <http://www.embedded.com>
- For late-breaking information about new features, bug fixes, known problems, and incompatibilities, read the release notes in this folder:

`<CWInstallDir>\MCU\`

where *CWInstallDir* is the directory in which the CodeWarrior software is installed.

[Table 1.2](#) lists the additional documents you can refer to for more information about CodeWarrior for Microcontrollers V10.x. These documents are categorized according to the four different documentation types as Getting Started, User Guides, Application Notes and Supporting Information.

Table 1.2 Related Documentation

Documentation Type	Document	Description	Location
Getting Started	Microcontrollers V10.x Quick Start	Explains the steps to install Microcontrollers V10.x, and create and debug a project.	<code><CWInstallDir>\MCU</code>
Getting Started	CodeWarrior Project Importer Quick Start	Explains the steps to convert a classic CodeWarrior project into an Eclipse IDE project.	<code><CWInstallDir>\MCU</code>
Getting Started	Eclipse Quick Reference Card	Introduces you to the interface of CodeWarrior for Microcontrollers V10.0 Eclipse-based IDE and provides a quick reference to the key bindings.	<code><CWInstallDir>\MCU</code>

Table 1.2 Related Documentation (*continued*)

Documentation Type	Document	Description	Location
Getting Started	HCS08 Profiling and Analysis for Microcontrollers V10.x Quick Start	Explains how to collect trace and critical code data after creating, building, and running a project on the HCS08 MC9S08QE128 target in the CodeWarrior for Microcontrollers version 10.x debugger.	<CWInstallDir>\MCU
Getting Started	ColdFire Profiling and Analysis for Microcontrollers V10.x Quick Start	Explains how to collect trace and critical code data after creating, building, and running a project on the ColdFire V1 MCF51JM128 target in the CodeWarrior for Microcontrollers version 10.x debugger.	<CWInstallDir>\MCU
User Guide	Freescale Eclipse Extensions Guide	Explains extensions to the CodeWarrior Eclipse IDE across all CodeWarrior products.	<CWInstallDir>\MCU \Help\PDF
User Guide	Microcontrollers V10.x Targeting Manual	Explains how to use CodeWarrior Development Studio for Microcontrollers V10.x	<CWInstallDir>\MCU \Help\PDF
User Guide	Microcontrollers V10.x HC08 Build Tools Reference Manual	Describes the compiler used for the Freescale 8-bit Microcontroller Unit (MCU) chip series.	<CWInstallDir>\MCU \Help\PDF
User Guide	Microcontrollers V10.x RS08 Build Tools Reference Manual	Describes the ANSI-C/C++ Compiler used for the Freescale 8-bit Microcontroller Unit (MCU) chip series.	<CWInstallDir>\MCU \Help\PDF

Introduction

Additional Information Resources

Table 1.2 Related Documentation (*continued*)

Documentation Type	Document	Description	Location
User Guide	Microcontrollers V10.x ColdFire Build Tools Reference Manual	Describes the compiler used for the Freescale 8-bit Microcontroller Unit (MCU) chip series	<CWInstallDir>\MCU\Help\PDF
User Guide	Microcontrollers V10.x MISRA-C:2004 Compliance Exceptions for the HC(S)08, RS08 and ColdFire Libraries Reference Manual	Describes the MISRA-C:2004 compliance exceptions for the HC(S)08, RS08, and ColdFire libraries.	<CWInstallDir>\MCU\Help\PDF
User Guide	CodeWarrior Development Tools EWL C Reference	Describes the contents of the Embedded Warrior Library for C. This document is available only in ColdFire Architecture.	<CWInstallDir>\MCU\Help\PDF
User Guide	CodeWarrior Development Tools EWL C++ Reference	Describes the contents of the Embedded Warrior Library for C++. This document is available only in ColdFire Architecture.	<CWInstallDir>\MCU\Help\PDF
User Guide	Microcontrollers V10.x HC(S)08/RS08 Assembler Reference Manual	Explains how to use the HC(S)08/RS08 Macro Assembler	<CWInstallDir>\MCU\Help\PDF

Table 1.2 Related Documentation (*continued*)

Documentation Type	Document	Description	Location
User Guide	Microcontrollers V10.x ColdFire Assembler Reference Manual	Explains the assembly-language syntax and IDE settings for the ColdFire assemblers	<CWInstallDir>\MCU\Help\PDF
User Guide	Microcontrollers V10.x HC(S)08/RS08 Build Tools Utilities Manual	Describes the following five CodeWarrior IDE utilities: SmartLinker, Burner, Libmaker, Decoder, and Maker.	<CWInstallDir>\MCU\Help\PDF
User Guide	Microcontrollers V10.x Profiling and Analysis Users Guide	Explains the CodeWarrior Profiling and Analysis tools. These tools provide visibility into an application as it runs on the simulator and hardware. Developers can use these tools to understand how an application runs, as well as identify operational problems.	<CWInstallDir>\MCU\Help\PDF
User Guide	USB TAP Users Guide	Explains the steps to develop and debug a number of processors and microcontroller using CodeWarrior USB TAP probe.	<CWInstallDir>\MCU\Help\PDF
User Guide	Ethernet TAP Users Guide	Explains the steps to develop and debug a number of processors and microcontroller using CodeWarrior Ethernet TAP probe.	<CWInstallDir>\MCU\Help\PDF

Introduction

Additional Information Resources

Table 1.2 Related Documentation (*continued*)

Documentation Type	Document	Description	Location
User Guide	Open Source BDM-JM60 Users Guide	Describes an Open Source programming and debugging development tool designed to work with Freescale HCS08, RS08, Coldfire V1, V2, V3 and V4, and DSC56800E microcontrollers.	<CWInstallDir>\MCU\Help\PDF
User Guide	Processor Expert Users Manual	Provides information about Processor Expert plug-in, which generates code from the Embedded Beans.	<CWInstallDir>\MCU\Help\PDF
User Guide	Device Initialization Users Manual	Provides information about the user interface, creating a simple design, configuring a device, generating initialization code, and using it in your application.	<CWInstallDir>\MCU\Help\PDF
Application Note	AN3859 - Adding Device(s) to the CodeWarrior Flash Programmer for Microcontrollers V10.x	Explains how to use the Flash Tool Kit to support additional flash devices on the Flash Programmer for CodeWarrior Development Studio for Microcontrollers V10.x.	<CWInstallDir>\MCU\Help\PDF
Application Note	AN3967 - How to Write Flash Programming Applets	Provides information on creating Flash configuration files for the Flash Programming interface.	<CWInstallDir>\MCU\Help\PDF

Table 1.2 Related Documentation (*continued*)

Documentation Type	Document	Description	Location
Application Note	AN4104 - Converting Classic ColdFire Projects to Microcontrollers V10.x	Explains how to convert a ColdFire project created in CodeWarrior Development Studio for Microcontrollers V6.2 or CodeWarrior Development Studio for ColdFire Architectures V7.1 to CodeWarrior Development Studio for Microcontrollers V10.x	<CWInstallDir>\MCU\Help\PDF
Supporting Information	Microcontrollers V10.x FAQ Guide	Lists most frequently asked or anticipated questions and answers to CodeWarrior Development Studio for Microcontrollers V10.x.	<CWInstallDir>\MCU\Help\PDF

Introduction

Additional Information Resources

Overview

This chapter provides an overview of CodeWarrior for Microcontrollers V10.0 and Eclipse IDE. It also provides system requirements for installing Microcontrollers 10.0.

This chapter consists of these topics.

- [Introduction to Microcontrollers 10.0](#)
- [Introduction to Eclipse IDE](#)
- [System Requirements](#)

Introduction to Microcontrollers 10.0

If you are an experienced CodeWarrior user, note that the CodeWarrior for Microcontrollers V10.0 environment uses the Eclipse IDE, whose user interface is substantially different from the *classic* CodeWarrior IDE.

Introduction to Eclipse IDE

The Eclipse IDE (Integrated Development Environment) is an open-source development environment that lets you develop and debug your software. It controls project manager, source code editor, class browser, compilers and linkers, and debugger.

Those who are more familiar with command-line development tools may find the concept of CodeWarrior project manager new. The project manager organizes all files related to your project. This lets you see your project at a glance and eases the organization and navigation between source code files.

The Eclipse IDE has an extensible architecture that uses plug-in compilers and linkers to target various operating systems and microprocessors. The IDE is hosted on Microsoft Windows, Win32 Linux, and other platforms. There are many development tools available for the IDE, including C, C++, and Java compilers for desktop and embedded processors.

For more information about the Eclipse IDE, read the Eclipse documentation at:

<http://www.eclipse.org/documentation/>

System Requirements

Following table lists system requirements for installing Microcontrollers 10.0:

Table 2.1 System Requirements

Hardware	Windows OS: PC with 1 GHz Intel® Pentium® compatible processor Linux® OS: 1.8 GHz Intel Pentium class processor (or better) 512 MB of RAM (1 GB recommended) CD-ROM drive Depending on host-target connection: Parallel Port, 9-pin Serial Port, or USB Port
Operating System	Microsoft® Windows XP 64-bit (Professional Edition), Microsoft Windows Vista® 32-bit and 64-bit (Home Premium Edition and Business Edition), or Microsoft Windows 7 32-bit and 64-bit (Home Premium Edition and Professional Edition) Red Hat Enterprise Edition 5.2
Disk Space	2 GB total 400MB on Windows system disk

CodeWarrior Development Process

This chapter describes stages involved in developing an application using the CodeWarrior IDE.

CodeWarrior Development Process

While working with the CodeWarrior IDE, you will proceed through the development stages familiar to all programmers: writing code, compiling and linking, and debugging.

See the Freescale Eclipse Extension Guide for:

- Complete information on tasks, such as editing, compiling, and linking
- Basic information on debugging

The difference between the CodeWarrior environment and traditional command-line environments is how the software (in this case the IDE) helps you manage your work more effectively.

If you are unfamiliar with an integrated environment in general, or with the CodeWarrior IDE in particular, you may find the topics in this section helpful. Each topic explains how each component of the CodeWarrior tools relates to the traditional command-line environment.

Project Files

A CodeWarrior project is analogous to a set of make files, because a project can have multiple settings that are applied when building the project. For example, you can have one project that has both a debug version and a release version of your program. You can build one or the other, or both as you wish. The different settings used to launch your program within a single project are called *launch configurations*.

The IDE uses the CodeWarrior Projects view to list all the files in a project. The files listed in the CodeWarrior Projects view include source code files and libraries.

You can add or remove files easily. You can also assign files to one or more different build configurations within the project, therefore files common to multiple build configurations can be managed simply.

The IDE automatically manages all the interdependencies between files and tracks which files have changed since the last build. This speeds the build process because the IDE only compiles those files that have changed since the last build.

In addition, the IDE stores the settings for compiler and linker options for each build configuration. You can modify these settings using the IDE, or with `#pragma` statements in your code.

Editing Code

The Eclipse IDE has an integral text editor designed for programmers. It handles text files in MS-DOS/Windows® and UNIX® formats.

To edit a source code file or any other editable file in a project, double-click the filename in the CodeWarrior project view to open the file.

The navigational features of the editor window lets you switch between related files, locate a particular function, mark a location within a file, or go to a specific line of code.

Compiling

To compile a source code file, ensure that the file is a part of the current launch configuration. If the file is in the configuration, select it in the project window and select **Project > Build Project** from the IDE menu bar.

To automatically compile all the files in the current launch configuration after you modify them, select **Project > Build Automatically** from the IDE menu bar.

Linking

Select **Project > Build Project** from the IDE menu bar to link object code into a final binary file. The Build Project command brings the active project up-to-date, then links the resulting object code into a final output file.

You control the linker through the IDE. There is no need to specify a list of object files. The Workspace tracks all the object files automatically.

You can modify the build configuration settings to choose the name of the final output file.

Debugging

Select **Run > Debug Configurations** from the IDE menu bar to debug your project. This command downloads the current project's executable to the target board and starts a debug session.

NOTE You must have previously entered debugger settings for the launch configuration by selecting **Run > Debug Configurations**. The IDE uses the settings in the launch configuration to generate debugging information and initiate communications with the target board.

You can now use the debugger to step through the program's code, view and change the value of variables, set breakpoints, and much more. See the Freescale Eclipse Extensions Guide and the Creating and Debugging Projects chapter of this manual for more information about debugging.

Using Microcontrollers Workbench

This chapter familiarizes you with the CodeWarrior for Microcontrollers V10.0 IDE and enables you to work with its components.

This chapter consists of these topics.

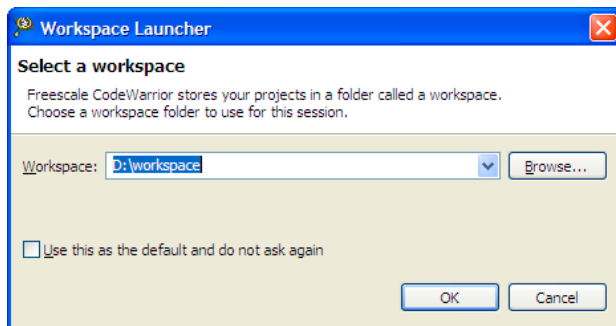
- [Workspace](#)
- [Workbench](#)
- [Welcome Page](#)
- [Help Documentation](#)
- [Perspective and View](#)
- [Editor](#)
- [Toolbars](#)

Workspace

To start workspace, select **Start > Programs > Freescale CodeWarrior > CodeWarrior for MCU v10.0 > CodeWarrior**.

When Microcontrollers V10.0 is launched first time, the **Workspace Launcher** dialog box appears (refer [Figure 4.1](#)), asking you to select a location for the workspace. The workspace is the directory that stores the source code and other files and settings related to your projects in Microcontrollers V10.0.

Figure 4.1 Workspace Launcher Dialog Box



Click the **OK** button to select the default location. If you want to specify a different location for the workspace, click the **Browse** button and select the required workspace directory. When you select the workspace location, make sure the location is not in your install directory.

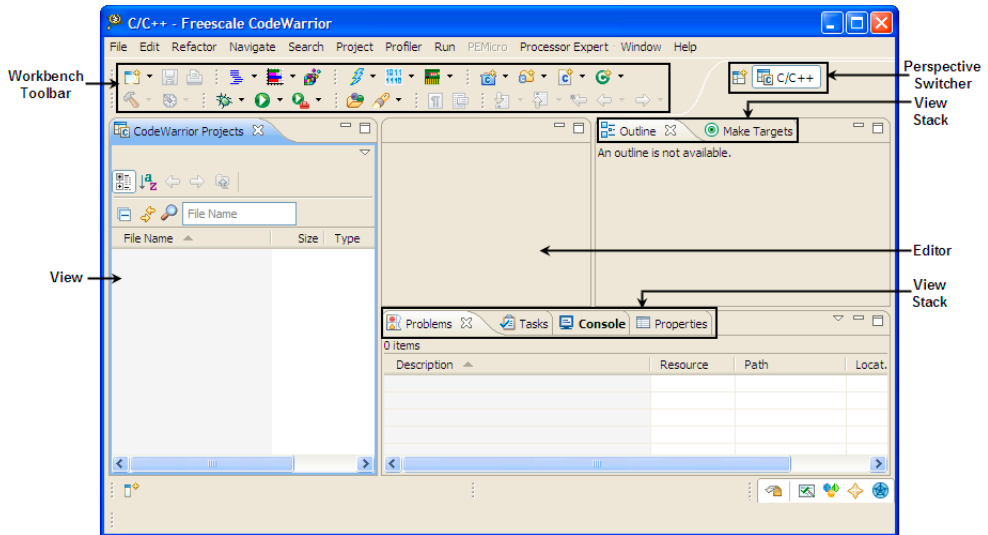
Workbench

After you select the workspace location, CodeWarrior for Microcontrollers V10.0 starts. If Microcontrollers V10.0 is started for the first time, the **Welcome** page is displayed ([Figure 4.3](#)), otherwise the Workbench window ([Figure 4.2](#)) is displayed.

To go to the Workbench window from the **Welcome** page, click **Go to Workbench** in the **Welcome** page.

NOTE For more information about the **Welcome** page, refer [Welcome Page](#).

Figure 4.2 Workbench Window



The Workbench window is the main CodeWarrior window in the IDE that facilitates the seamless integration of the CodeWarrior tools. All development happens within the Workbench window.

The Workbench window can have one or more perspectives associated with it. A perspective defines the layout of the views and editors in the Workbench window.

Using Microcontrollers Workbench

Welcome Page

[Table 4.1](#) describes the components of the Workbench window.

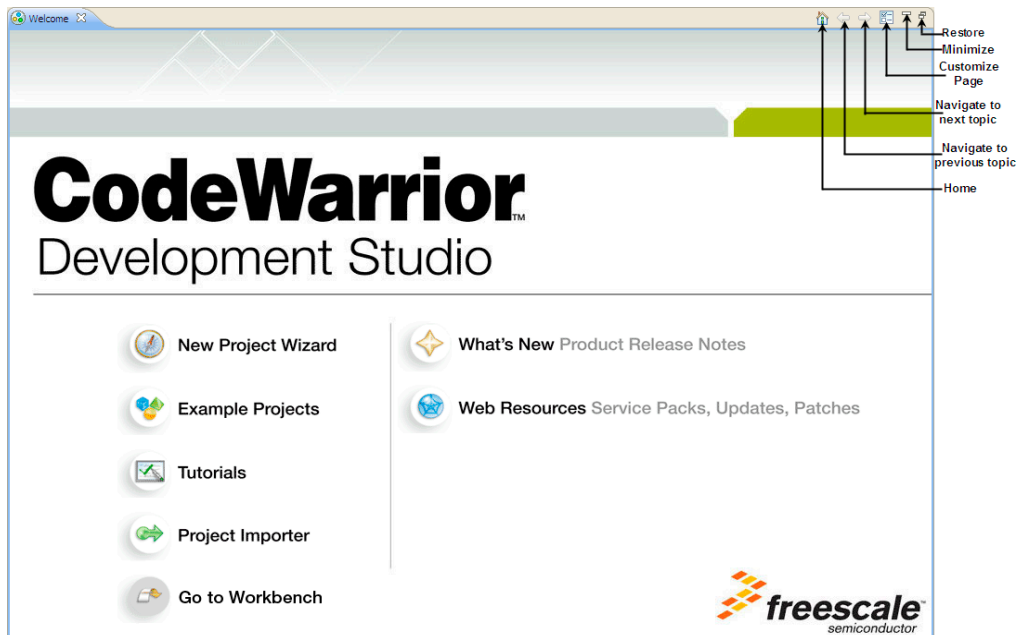
Table 4.1 Components of Workbench Window

Workbench Component	Description
View	<p>Enables you to navigate through the information in the Workbench, such as the resources in the Workbench or the properties of the active editor. A view might appear by itself, or tabbed with other views. The term ‘<i>View stack</i>’ refers to the tabbed views in the IDE.</p> <p>For more information about views, refer Perspective and View</p>
Editor	<p>Enables you to edit or browse through a resource. Tabs in the editor area indicate the names of resources that are currently open for editing.</p> <p>For more information about Editor, refer Editor</p>
Toolbars	<p>Provides you quick access to the actions to perform in the Workbench, such as create or debug a project.</p> <p>For more information about toolbars, refer Toolbars</p>

Welcome Page

The **Welcome** page ([Figure 4.3](#)) is the first page you see when you start Microcontrollers V10.0 or open a workspace for the first time.

Figure 4.3 Welcome Page



To return to the **Welcome** page or to access the **Welcome** page from the Workbench window, select **Help > Welcome** from the IDE menu bar.

Welcome page content typically includes links to several other pages that introduces you with the product and help you become more familiar with it. The following table lists the links available in the **Welcome** page:






Table 4.2 Welcome Page Content

Links	Icon	Description
New Project Wizard		Starts the New Bareboard Project wizard. For information about how to create a project using the New Bareboard Project wizard, refer Creating New Microcontrollers Bareboard Project .
Example Projects		Provides you to access to the sample projects available in the product.

Using Microcontrollers Workbench

Welcome Page

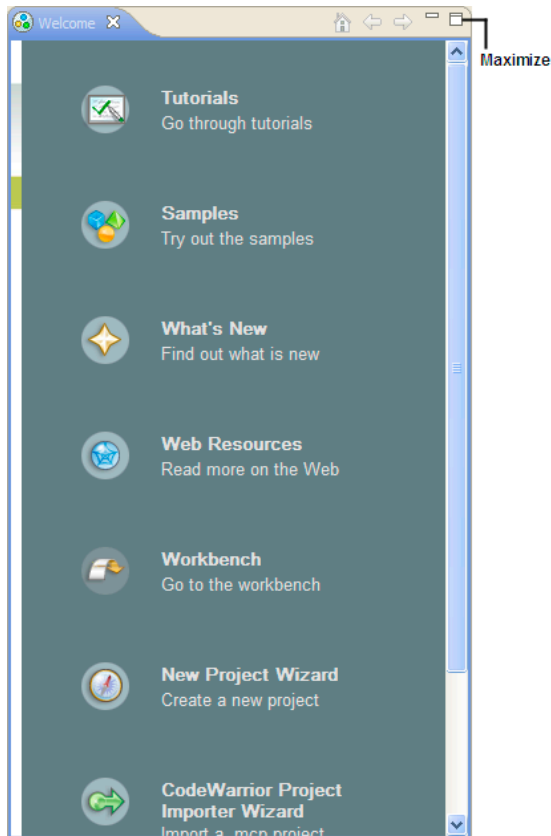
Table 4.2 Welcome Page Content

Links	Icon	Description
Tutorials		Takes you to the Tutorials page that provides links to the Cheat Sheets, Getting Started guide, and Help system.
Project Importer		Starts CodeWarrior Project Importer wizard. You can use the wizard to import a classic CodeWarrior project into Eclipse-based CodeWarrior IDE.
Go to Workbench		Takes you to the Workbench window.
What's New		Gives you information about the major new features in this release of the product.
Web Resources		Takes you to http://www.freescale.com/codewarrior

You can use the **Welcome** page toolbar in the top right corner of the page to go to home page if you are on a different page, navigate across the pages, customize the **Welcome** page, and minimize or restore a page.

The **Welcome** page can be displayed in two modes—full and stand-by. In full mode, the view is maximized across the whole window ([Figure 4.3](#)), whereas in stand-by mode it only shows up as a regular view ([Figure 4.4](#)). To view the **Welcome** page as a regular view, click the **Restore** button in the top right corner of the **Welcome** page. To view it in full mode, click the **Maximize** button in the top right corner of the **Welcome** view.

Figure 4.4 Welcome Page — Regular View



Help Documentation

You can use the Microcontrollers 10.0 help documentation to understand and work with Microcontrollers 10.0.

Eclipse Help

You can access the help system in the Workbench window using the Help view or a separate Help window.

- Help view — Provides help inside the Workbench window. To open the **Help** view, select **Help > Dynamic Help** or **Help > Search**. The view will display the Related

Topics or Search page, respectively. You can also open the Related Topic or context-sensitive help by pressing **F1**. The Related topic or context-sensitive help displays the help topics related to the GUI element selected in the CodeWarrior IDE. For example, if you select the **CodeWarrior Projects** view and press F1. The **Help** view will display the help topics related to the **CodeWarrior Projects** view

You can use links at the bottom of the **Help** view to browse through the help file contents.

- **Help window** — Provides the same content as the Help view, but in a separate window. To open the Help window, select **Help > Help Contents**. The Help window is displayed with the table of contents for the product documentation. Click on one of the links to expand the navigation tree for a set of documentation.

Cheat Sheets

Cheat sheets guide you through a series of steps required to perform a particular task, such as creating and debugging a project. You can launch the tools required to perform the task, such as the New Project wizard for creating new projects, from the cheat sheet or you can use the description provided in the cheat sheet to launch the tool.

To launch a cheat sheet in the Workbench window:









1. Select **Help > Cheat Sheets**. The **Cheat Sheet Selection** dialog box appears.
2. Expand the required category and select the cheat sheet you want to open.
3. Click **OK**.

The cheat sheet opens as a view. At any time, only one cheat sheet is open and active. When you launch a cheat sheet, any opened cheat sheet is closed before the new one is opened. The completion status of closed cheat sheet is saved.

WARNING! The cheat sheets might not appear if a view is maximized and the **Cheat Sheets** view is detached.
A workaround to this issue is to close the **Cheat Sheets** view, if open. Then, restore the maximized view and select **Help > Cheat Sheets** to access the desired cheat sheet.

[Table 4.3](#) describes the various controls available in a cheat sheet to enable you to work with the cheat sheet:

Table 4.3 Cheat Sheet Controls

Controls	Description
 Click to Begin	Click to start working with the cheat sheet. When you click this icon, the first step in the cheat sheet is expanded and highlighted.
 Click to perform	Click to launch the tool required to perform the task in the cheat sheet. If the tool is a dialog box, the cheat sheet opens to the right of the dialog box.
 Click when complete	Click to move to the next step. When you click this icon, the next step in the cheat sheet is expanded and highlighted.
For Step:  For Task:  Skip this task	Click to skip the current step or task. When you skip a step or task, the step or task will have the skip mark  in the left margin. If the step or task does not present this control, you must perform that step or task and you cannot skip it.
 Click to redo	Click to redo an already completed or skipped step. After redoing a step, the cheat sheet will continue from that step.
 Click to Restart	Click to restart a cheat sheet from the first step, anytime after starting it.

Following is the list of cheat sheets categories and the cheat sheets in each of the category available in your product build.

- CodeWarrior Core Features
 - Importing Archive Files
 - Importing Breakpoints
 - Importing Classic CodeWarrior Projects
 - Importing Existing Projects into Workspace
 - Importing Preferences
 - Importing Resources from Local File System

Using Microcontrollers Workbench

Perspective and View

- Importing Team Project Set
- Making C/C++ the IDE's Default Perspective
- Using the Flash Programmer
- CodeWarrior for Microcontrollers Features
 - Building Library (HCS08)
 - Changing P&E Connections Setting
 - Configuring Perspective
 - Creating New Project from Example Project
 - Creating, Building, and Debugging Project
 - Debugging Projects in ROM
 - Debugging Project Using Command Line
 - Examples: Porting Classic IDE Projects to Eclipse
 - Importing and Debugging Externally Built Executable File
 - Using Memory View
 - Using Microcontrollers Change Wizard
 - Using Registers View
 - Working with Build Configurations
- CodeWarrior Profiling and Analysis Features
 - Creating, Debugging, Collecting, and Viewing Data (For the ColdFire V1 Target)
 - Creating, Debugging, Collecting, and Viewing Data (For the HCS08 Target)

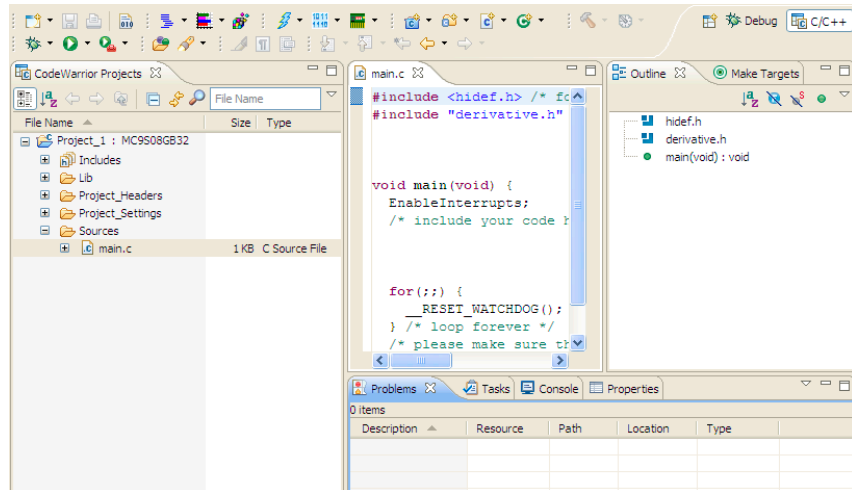
Perspective and View

Each perspective is a collection of views, which provides a set of functionality aimed at accomplishing a specific type of task. The most-commonly used perspectives in CodeWarrior IDE are C/C++ and Debug.

C/C++ Perspective

C/C++ perspective is tuned for working with the C/C++ projects. The views in the C++ perspective let you perform the tasks involved in creating the C/C++ programs. [Figure 4.5](#) displays the C++ perspective in the CodeWarrior Workbench window.

Figure 4.5 C/C++ Perspective



[Table 4.4](#) describes the commonly used views in the C/C++ perspective.

Table 4.4 C++ Perspective Views

View	Description
CodeWarrior Projects	Lets you to perform tasks related to your CodeWarrior projects, such as browse through the CodeWarrior project source files and open files. See <i>Freescale Eclipse Extension Guide</i> for more information about the CodeWarrior Projects view.
Console	Acts as a virtual console that shows the output of the execution of your program, and enables you to enter input for the program.
Properties	Displays property names and basic properties of a selected resource.
Problems	Displays the errors encountered during building the project.
Tasks	Displays all the tasks in the Workbench window. The view displays tasks associated with specific files and generic tasks that are not associated with any specific file.

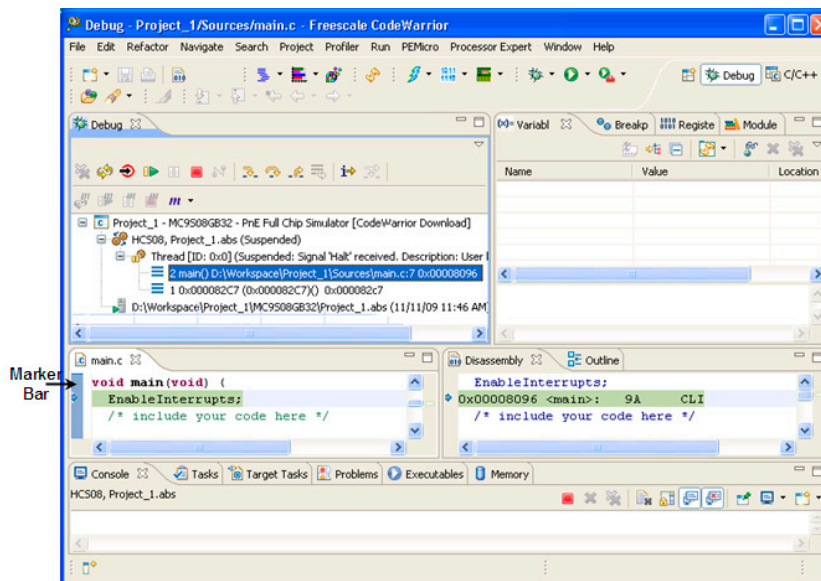
Table 4.4 C++ Perspective Views

View	Description
Make Targets	Enables you to select the make targets that you want to build in your workspace.
Outline	Displays an outline of a structured file that is currently open in the editor area, and lists structural elements. The contents of the Outline view are editor-specific.

Debug Perspective

The Debug perspective lets you manage the debugging or running of a program in the Workbench window. You can control the execution of your program by setting breakpoints, suspending launched programs, stepping through your code, and examining the contents of variables. [Figure 4.6](#) displays the Debug perspective in the CodeWarrior Workbench window.

Figure 4.6 Debug Perspective



[Table 4.5](#) describes the commonly used views in the Debug perspective.

Table 4.5 Debug Perspective Views

View	Description
Debug	Shows the target debugging information in a tree hierarchy. The Debug view shows stack crawl information for each thread running on the target.
Variables	Displays information about the variables in a selected stack frame. When value of a variable changes, the background color for the variable changes to yellow. The Variables view does not refresh as you run your executoo. A refresh occurs when execution stops.
Breakpoints	Lists the breakpoints set for your program. You can perform different actions on breakpoints using the context menu for the Breakpoints view.
Registers	Lists information about the registers in a selected stack frame. When the program stops, the changed values are highlighted in the Registers view.
Modules	Shows the application executable and all shared libraries loaded by the application during a debug session.
Memory	Lets you monitor and modify the process memory.
Debugger Shell	Enables you to execute commands in a command-line environment. The command-line debugger engine executes the commands that you enter in the debugger shell view, then displays the results.
Disassembly	Shows the loaded program as assembly language instructions mixed with source code for comparison.

Table 4.5 Debug Perspective Views

View	Description
Expressions	Enables you to add an expression. You can view the expression and its value in the Expressions view. When the execution of a program is suspended, all expressions are reevaluated.
Target Tasks	Enables you to run utilities, such as Hardware Diagnostics and Import/Export/Fill Memory on a target device.

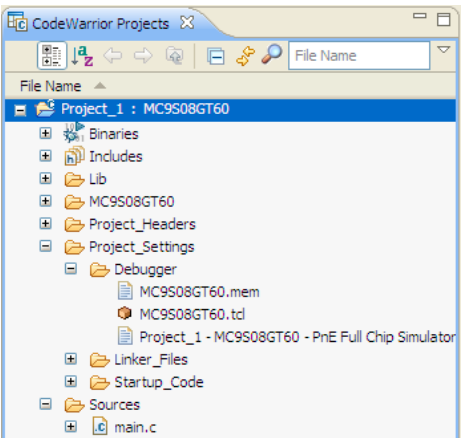
NOTE For information about adding a view to a perspective, refer to [Opening Views](#).

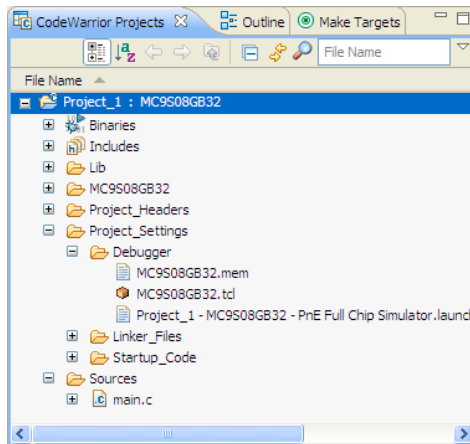
NOTE For more information about the Debug perspective and the Debug perspective views, see *Freescale Eclipse Extensions Guide*.

Working with Views

A view can be standalone or stacked with other views in a tabbed view. [Figure 4.7](#) shows the standalone and tabbed options for the CodeWarrior Projects view. To activate a view that is part of a tabbed view, click its tab.

Figure 4.7 CodeWarrior Projects View — Standalone and Tabbed



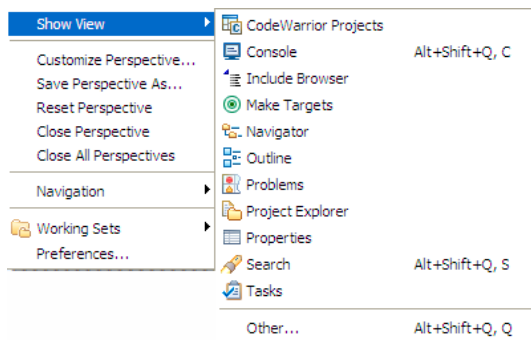


Opening Views

To open a view that is not included in the current perspective:

1. Select **Window > Show View**.

Figure 4.8 Show View Submenu



2. Perform one of the following:
 - Click the view that you want to add to the perspective, in the **Show View** submenu.
 - Click **Other** to view a list of all the available views. Select a view from the **Show View** dialog box and click the **OK** button.

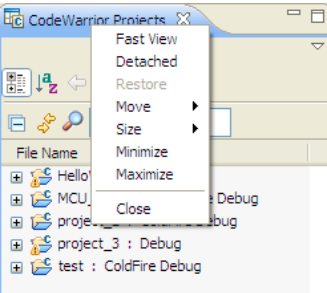
The selected view is added to the perspective.

Using View Menus

Views have two menus, view tab context menu and view pull-down menu.

You can access the view's tab context menu by right clicking on the view's tab (see [Figure 4.9](#)).

Figure 4.9 View Tab Context Menu




[Table 4.6](#) describes the commands included in the view tab context menu.

Table 4.6 View Tab Context Menu Commands

Command	Description
Fast View	Displays a view as a fast view. Fast views provide quick access to the views that you use often. Fast views are represented by the toolbar buttons on the Fast View Toolbar .
Detached	Displays a view as a detached view. Detached views are views that are displayed in a separate window. They work like other views except they are always shown in front of the Workbench window.
Restore	Restores a maximized or minimized view to its original size.
Move	Allows you to move the view or the view tab group to a different position in the Workbench window.
Size	Allows you to resize a view. You can resize a view in Top, Bottom, Right, or Left direction depending upon the current position of the view.
Minimize	Minimizes the view.

Table 4.6 View Tab Context Menu Commands

Command	Description
Maximize	Maximizes the view.
Close	Closes the view.


The second menu, called the view pull-down menu, is accessed by clicking  on the view. The view pull-down menu contains operations that apply to the entire contents of the view, and not to a specific item shown in the view. Operations for sorting and filtering are commonly found in the view pull-down menu.

Working with Perspectives

You can perform a set of tasks on the perspectives, such as you can change the way a perspective looks and then save the changes or restore the perspective's layout.

Opening Perspective and Switching Between Perspectives

To open a new perspective:

1. Click  on the perspective switcher or select **Window > Open Perspective**.
2. Select the perspective you want to open or select **Other** to view a list of all the available perspectives. The **Open Perspective** dialog box opens.
3. Select the perspective that you want to open and click the **OK** button.

When the perspective opens, the title bar of the Workbench window displays the name of the selected perspective.

Open perspectives are represented by icons on the perspective switcher. When you have more than one perspective open, you can switch between them by clicking the icons on the toolbar.

By default, a perspective opens in the same window. If you want to open the perspective in a new window, change the setting in the **Window > Preferences > General > Perspectives** preference page.

Configuring Menu and Toolbar Options in Perspective

In addition to configuring the layout of your perspective, you can also configure the command groups displayed on the toolbars and menus, and options available on the following submenus:

- **File > New**
- **Window > Open Perspective**
- **Window > Show View**

To configure menu and toolbar options in a perspective:

1. Open or switch to the perspective that you want to customize.
2. Select **Window > Customize Perspective**. The **Customize Perspective** dialog box opens.
3. To add or remove shortcuts to or from the submenus:
 - a. Click the **Shortcuts** tab.
 - b. Select a submenu from the **Submenus** list box.
 - c. Select a shortcut category from the **Shortcut Categories** list. The associated shortcuts are displayed in the **Shortcuts** list in the right pane of the **Shortcuts** tab page.
 - d. Check or clear the checkbox next to the shortcut category if you want to add or remove all the shortcuts associated with it, or check or clear the checkbox next to the desired shortcut in the **Shortcuts** list.
4. To add or remove command groups to or from the current perspective:
 - a. Click the **Commands** tab.
 - b. Select the command group to be added or removed from the **Available command groups** list. The **Menubar details** and **Toolbar details** columns display the menubar and toolbar options that will be added or removed from the current perspective for the selected command group.
 - c. Check or clear the command group to be added or removed.
5. Click the **OK** button.

Saving or Resetting a Perspective

After modifying a perspective by adding, deleting, or moving views, or by adding or removing menubar and toolbar options, you can save the changes for future use or restore the perspective to its original layout.

To save the changes to a perspective:

1. Open or switch to the perspective that you want to save.
2. Select **Window > Save Perspective As**. The **Save Perspective As** dialog box opens.
3. Type a new name for the perspective in the **Name** field.
4. Click the **OK** button.

The name of the new perspective is added to the **Window > Open Perspective** submenu.

To restore a perspective:

1. Open or switch to the perspective that you want to restore.
2. Select **Window > Reset Perspective**.

The perspective is displayed with its default layout.

Deleting a User-Defined Perspective

You can delete perspectives that you define yourself, but not those that are delivered with the Workbench window. To delete a user-defined perspective:

1. Select **Window > Preferences**. The **Preferences** dialog box opens.
2. Select **General > Perspectives**.
3. Select the perspective that you want to delete from the **Available perspectives** list, and click the **Delete** button.
4. Click the **OK** button.

Editor

An editor is a visual component within the Workbench window. It is typically used to edit or browse a resource. Typically, editors are launched by double-clicking on a resource in a view. Modifications made in an editor follow an open-save-close lifecycle model.

Opening Files for Editing

You can launch an editor for a given file in one of the following ways.

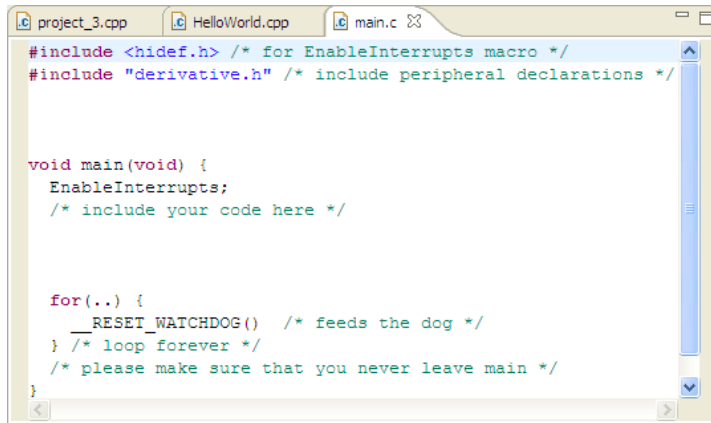
- By right-clicking the file in one of the views and then selecting **Open** from the context menu.
- By double-clicking the file in one of the views.
- By double-clicking a bookmark that is associated with the file, in the Bookmarks view.

- By double-clicking an error or warning, or task record that is associated with the file, in the Problems view.

All of the above alternatives open the file in the default editor for that type of file. To open it in a different editor, select **Open With** from the file's context menu.

Multiple files can be opened in different editor windows, which are then stacked together in the editor area as seen in [Figure 4.10](#).

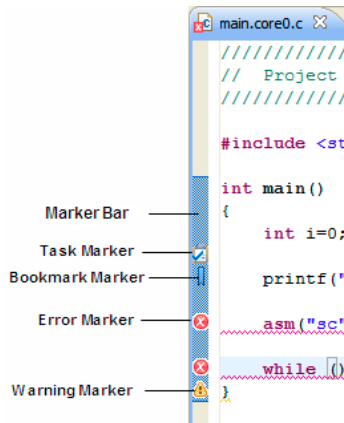
Figure 4.10 Editor Area



Working with Markers in Editor Area

Markers are icons that are associated with various Workbench kresources. You can view markers either in the marker views, such as Tasks, Bookmarks, and Problems views, or in the marker bar in the editor area (see [Figure 4.11](#)).

Figure 4.11 Marker Bar in Editor Area



[Table 4.7](#) describes different types of markers that can be used in the editor area.

Table 4.7 Markers in Editor Area

Marker	Description
Task	Represents a work item. Tasks can be of two types, automatically generated tasks associated with a resource, and user-specified tasks that may or may not be associated with a resource. Both of these type of tasks appear in the Tasks view.
Problems	<p>Represent invalid states and are categorized as follows:</p> <ul style="list-style-type: none"> • Error markers — Indicates the source location of the syntax or compilation errors. • Warning markers — Indicates the source location of warnings, such as compilation warnings. • Information markers — Indicates the source location of information only tasks. <p>Problems markers appear in the Problems view.</p>
Bookmark	Places an anchor either at a specific line in a resource or on the resource itself. Bookmark marker appear in the Bookmarks view.

Toolbars

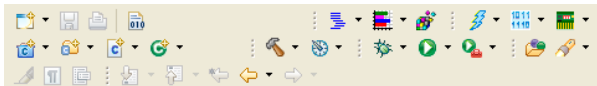
There are five kinds of toolbars in the Workbench window.

- [Workbench Toolbar](#)
- [View Toolbar](#)
- [Perspective Switcher](#)
- [Trim Stack](#)
- [Fast View Toolbar](#)

Workbench Toolbar

The Workbench toolbar ([Figure 4.12](#)) is displayed at the top of the Workbench window just below the menu bar. The toolbar displays different commands based on the active perspective. Items in the toolbar might be enabled or disabled based on the state of either the active view or editor.

Figure 4.12 Workbench Toolbar (For C/C++ Perspective)



[Table 4.8](#) describes the Workbench toolbar commands.

Table 4.8 Workbench Toolbar Commands






Command	Icon	Description
New		Creates a new project, group, folder, file, or class.
Save		Saves the content of the current editor. The Save button is disabled if there are no unsaved changes in the current editor.
Print		Prints the content of the current editor.
Build All		Builds all projects.
Trace		Enables simulator and the target hardware to generate the trace data.

Table 4.8 Workbench Toolbar Commands








Command	Icon	Description
Critical Code		Enables simulator and the target hardware to generate the critical code data.
All Results		Opens the Trace and Profiler Results view that can be used to view the data file created while running the application on the simulator and target board.
Flash Programmer		Displays a drop-down menu that contains following options: <ul style="list-style-type: none"> • Open Flash Programmer: Opens Target Tasks view using which you can create a Flash Programmer task. • Import Flash Task: Allows you to create a new Flash Programmer task using pre-defined flash task.
Hardware diagnostic		Displays a drop-down menu that contains following options: <ul style="list-style-type: none"> • Open HW Diagnostic: Opens Target Tasks view using which you can create a Hardware Diagnostic task. • Import HW Diagnostic Task: Allows you to create a new Hardware Diagnostic task using pre-defined HW Diagnostic task.
Import Export Memory		Displays a drop-down menu that contains following options: <ul style="list-style-type: none"> • Open Import Export Memory: Opens Target Tasks view using which you can create a Import/Export/Fill Memory task. • Import IEM Task: Allows you to create a new Import/Export/Fill Memory task using pre-defined IEM task.
New C/C++ Project		Creates a new C/C++ project.
New C/C++ Source Folder		Creates a folder within the current project.

Table 4.8 Workbench Toolbar Commands

















Command	Icon	Description
New C/C++ Source File		Creates a file within the current project.
New C++ Class		Creates a C++ class within the current project.
Build selected		Builds active configurations of the selected projects.
Manage configurations		Launches the Manage configurations dialog box, which allows you to manage configurations for the current project.
Debug		Launches the Debug As dialog box. In the Debug As dialog box, you can specify settings for debugging the project, such as the debugger to use to debug a project.
Run		Launches the Run As dialog box. In the Run As dialog box, you can specify settings to run a project, such as the program arguments to use.
External Tools		Launches the External Tools dialog box, which allows you to create, manage, and run launch configurations.
Open Element		Brings up the Open Element window, used to open up the declaration of C/C++ classes, structures, unions, typedefs, enumerations, namespaces, functions, methods, and variables.
Search		Launches the Search dialog box with the C/C++ Search page at front.
Show Whitespace Characters		Displays whitespace characters in the current file in the editor area.
Show Source of Selected Element Only		Displays source of the element selected in the current file in the editor area.

Table 4.8 Workbench Toolbar Commands

Command	Icon	Description
Next Annotation		Brings the cursor to the next annotation in the current file in the editor area.
Previous Annotation		Brings the cursor to the previous annotation in the current file in the editor area.
Last Edit Location		Brings the cursor to the last edited location in the file. If the file that was edited last has been closed, it is re-opened.
Back		Navigates back through open files in the editor area.
Forward		Navigates forward through open files in the editor area.

View Toolbar

The view toolbars appear in the title bar of a view. Actions in a view's toolbar varies with the view and apply only to the view in which they occur. Some view toolbars include a Menu button, shown as an inverted triangle, that opens the view pull-down menu containing actions for that view. [Figure 4.13](#) displays the CodeWarrior Projects view toolbar.

Figure 4.13 CodeWarrior Projects View Toolbar

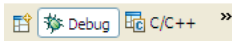


Perspective Switcher

Perspective switcher allows quick access to the perspectives that are currently open. It also contains a button that can open new perspectives. The perspective switcher is by default located in the top-right, next to the main toolbar. However, it is also possible to position it below the main toolbar (top-left), or vertically on the left-hand side of the Workbench window (left).





By default, the perspective switcher displays the name of the perspectives, but it is possible to hide the text and show only the icons. To reposition the perspective or hide the text, right-click on it and select the appropriate item from the context menu. [Figure 4.14](#) displays the perspective switcher.

Figure 4.14 Perspective Switcher



[Table 4.9](#) lists the perspective switcher commands.

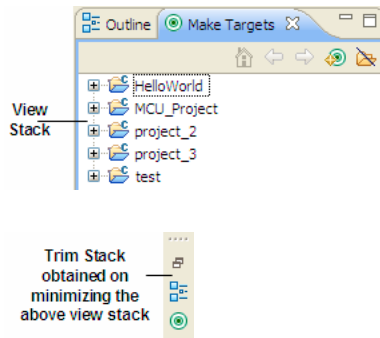
Table 4.9 Perspective Switcher Commands


Command	Icon	Description
Open Perspective		Click to open a new perspective or switch to a different perspective.
Perspectives	 	Click the icon of a perspective to switch to that perspective. For example, click the Debug icon to switch to the Debug perspective. Perspective switcher displays icon for only those perspectives that are already opened.
Show List		Click to access the list of the already opened perspectives.

Trim Stack

Minimizing a view stack produces a toolbar in the trim at the outer edge of the Workbench window called Trim Stack (see [Figure 4.15](#)).

Figure 4.15 Trim Stack



Trim stack contains an icon for each of the views in the stack. Clicking on one of these icons displays the view as an overlay on the Workbench window. Click  on Trim Stack to restore the views in the stack to their original position and size.

Fast View Toolbar

The Fast View toolbar contains icons representing the current set of the fast views. Fast views are hidden views that can be quickly opened and closed. They work like other views except they do not take up space in your Workbench window.

The Fast View toolbar appears in the bottom left corner of the Workbench window by default. However, it is possible to position it on the left or right as well. [Figure 4.16](#) displays the Fast View toolbar.

Figure 4.16 Fast View Toolbar



When you click the toolbar button for a fast view, that view opens temporarily in the current perspective overlaying it. As soon as you click outside that view or the view loses focus, it is hidden again.

To create a new fast view:

1. Click the left-most button on the Fast View toolbar. A menu containing a list of views opens up.
2. Select a view from this list or from the dialog box that appears if you select **Other**. The selected view is added to the Fast View toolbar.

NOTE You can also create a new fast view by dragging any open view to the Fast View toolbar or by selecting **Fast View** from the view's tab context menu.

To convert a fast view back into a normal view, select **Fast View** from the view tab's context menu or drag it back to the Workbench window.

Creating and Debugging Projects

This chapter takes you through the CodeWarrior for Microcontrollers V10.x IDE programming environment. This tutorial is not intended to teach you how to program. It instead teaches you how to use the CodeWarrior IDE to write and debug applications for a target platform.

This chapter consists of these topics:

- [Creating Projects](#)
- [Debugging Projects](#)

Creating Projects

This topic describes how to create a project using the Microcontrollers New Bareboard Project wizard.

NOTE For information about the various pages that the wizard displays as it assists you in creating a CodeWarrior project, refer *Microcontrollers V10.x Targeting Manual*.

Creating New Microcontrollers Bareboard Project

1. Select **Start > Programs > Freescale CodeWarrior > CodeWarrior for MCU v10.0 > CodeWarrior**.

CodeWarrior for Microcontrollers V10.0 launches. The **Workspace Launcher** dialog box appears, prompting you to select a workspace to use.

2. Click the **OK** button to accept the default workspace. To use a workspace different from the default, click the **Browse** button and specify the desired workspace. The CodeWarrior IDE launches.

Creating and Debugging Projects

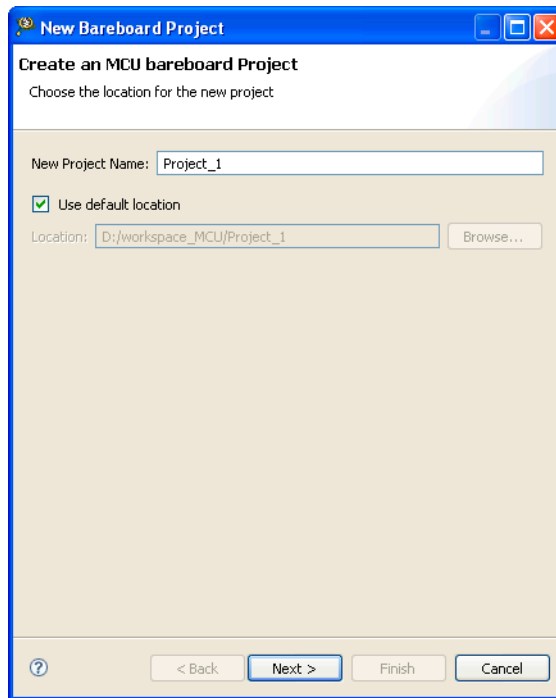
Creating Projects

3. Select **File > New > Bareboard Project** from the IDE menu bar.

The **New Bareboard Project** wizard starts. The **Create an MCU bareboard Project** page appears.

4. Specify a name for the new project. For example, enter the project name as `Project_1`.

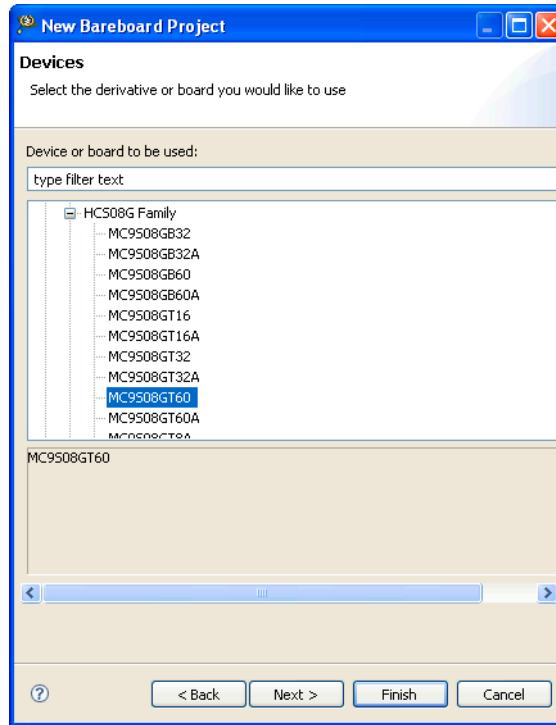
Figure 5.1 New Bareboard Project Wizard — Create an MCU Bareboard Project Page



5. Click **Next**.

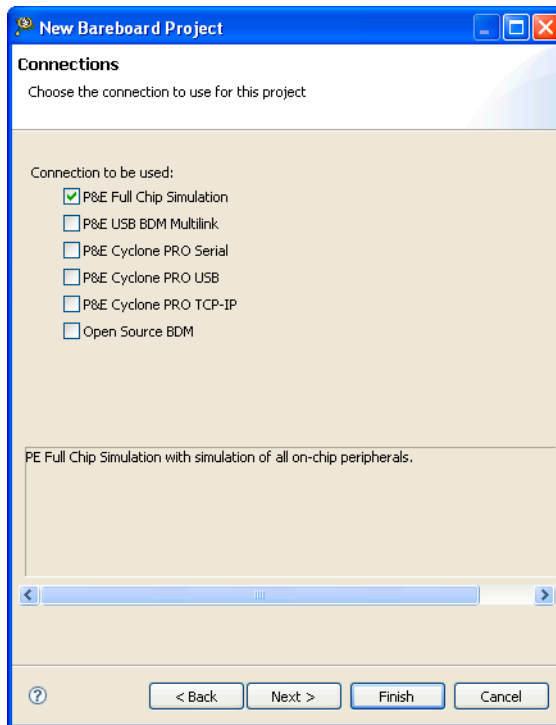
The **Devices** page appears.

Figure 5.2 New Bareboard Project Wizard — Devices Page



6. Expand the tree control and select the derivative or board you would like to use. For example, select **HCS08 > HCS08G Family > MC9S08GT60**.
7. Click **Next**.
The **Connections** page appears.

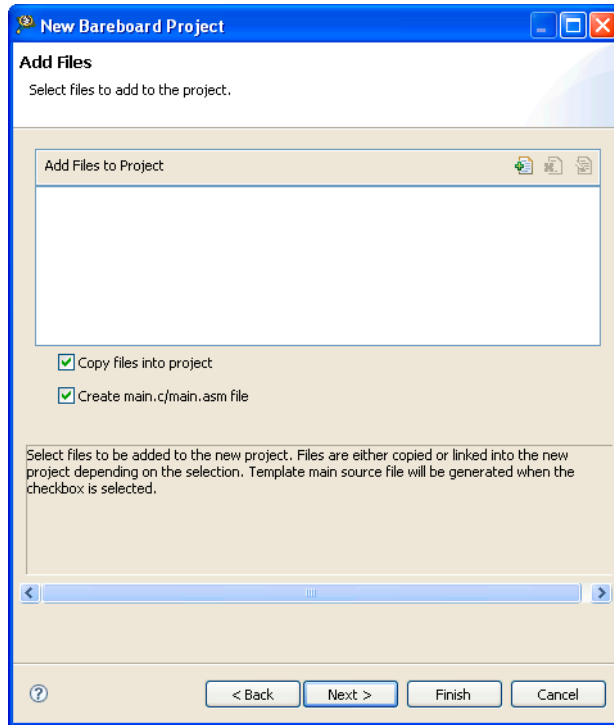
Figure 5.3 New Bareboard Project Wizard — Connections Page



8. Select the appropriate connection.
9. Click **Next**.

The **Add Files** page appears.

Figure 5.4 New Bareboard Project Wizard — Add Files Page




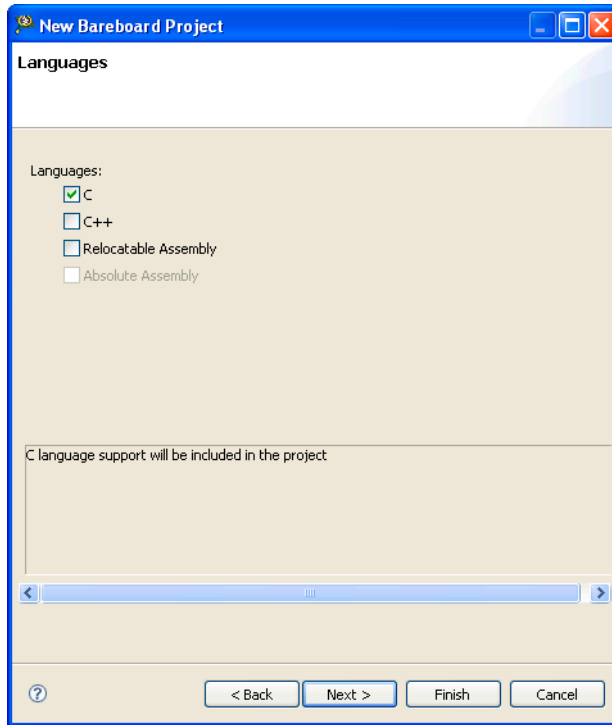
10. If you want to add a file to the project, click .
The **Add File Path** dialog box appears.
 - a. Type the path of the file you want to add to the project or browse to the file by clicking the **File system** button.
 - b. Click **OK** to close the **Add File Path** dialog box.
 - c. Check the **Copy files into project** checkbox if you want to add the selected file to the project. If you clear the **Copy files into project** checkbox, the file is linked into the project and not copied.
 - d. Clear the **Create main.c/main.asm file** checkbox if you do not want to create the main source file in the project.
11. Click **Next**.
The **Languages** page appears.

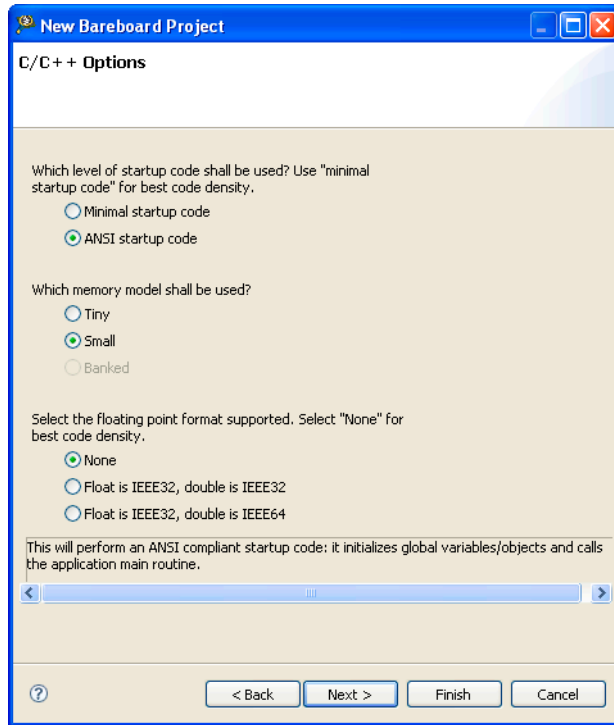
Figure 5.5 New Bareboard Project Wizard — Language Page



12. Select the programming language you want to use. For example, check the C checkbox.
13. Click **Next**.

The **C/C++ Options** page appears.

Figure 5.6 New Bareboard Project Wizard — C/C++ Options Page



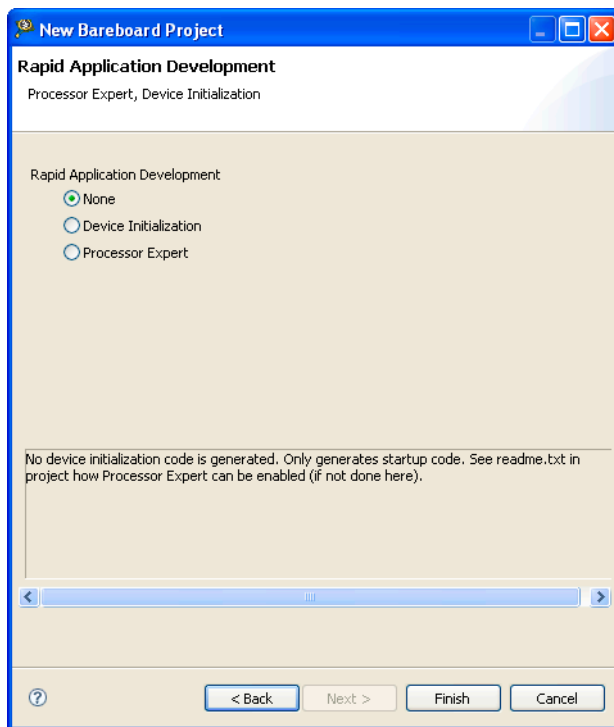
14. Select the appropriate level of startup code, memory model, and floating point format.
15. Click **Next**.

The **Rapid Application Development** page appears.

Creating and Debugging Projects

Creating Projects

Figure 5.7 New Bareboard Project Wizard — Rapid Applications Development Page



16. Select the appropriate rapid application development tool.

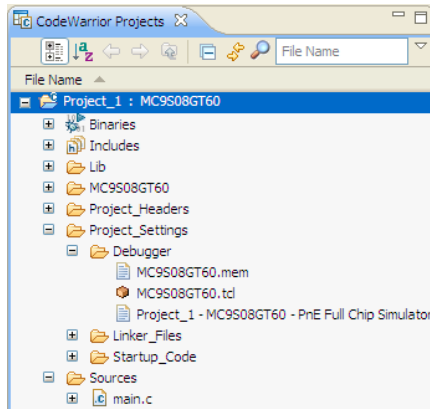
NOTE If you select **Processor Expert** in the **Rapid Application Development** page, click **Next** to display the **Processor Expert MCU Pin Variants and Configuration** page. Select the required MCU pin variant and configuration and click **Finish**.

17. Click **Finish**.

The wizard creates a project according to your specifications. You can access the project from the **CodeWarrior Projects** view in the Workbench window ([Figure 5.8](#)).

18. Right-click the project and select **Build Project** from the context menu to build the project.

Figure 5.8 CodeWarrior Projects View



The new Microcontrollers project is ready for use. You can now customize it by adding your own source code files, changing debugger settings, or adding libraries.

NOTE The contents of the project directory vary depending upon the options selected while creating the project.

Debugging Projects

When you use the **New Bareboard Project** wizard to create a new project, the wizard sets the debugger settings of the project's launch configurations to default values. You can change these default values based on your requirements.

To modify the debugger settings and debug the project:

1. From the main menu bar of the IDE, select **Run > Debug Configurations**.

The **Debug Configurations** dialog box appears. The left side of this dialog box has a list of debug configurations that apply to the current application.

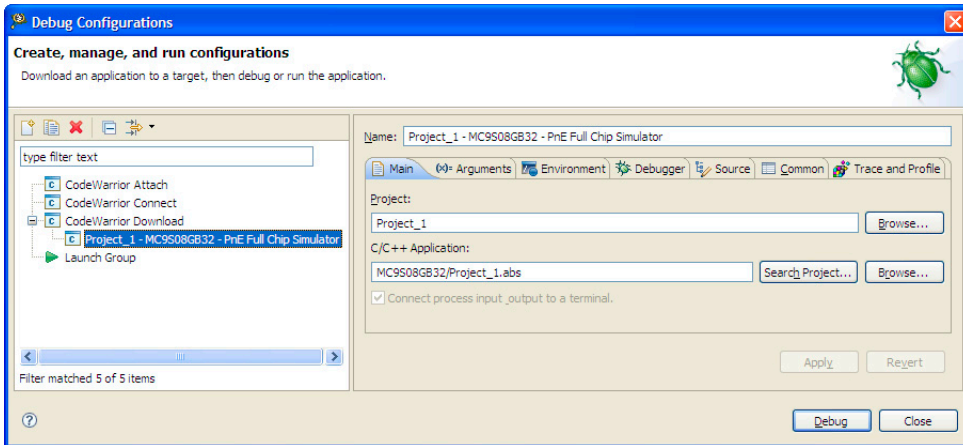
2. Expand the **CodeWarrior Download** configuration.
3. From the expanded list, select the debug configuration that you want to modify.

[Figure 5.9](#) displays the **Debug Configurations** dialog box with the settings for the debug configuration you selected.

Creating and Debugging Projects

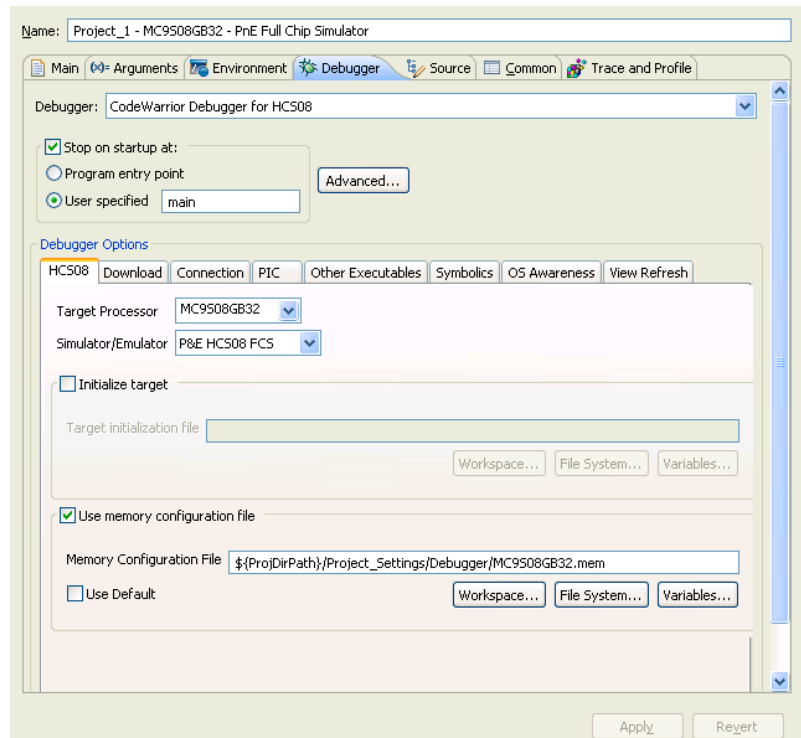
Debugging Projects

Figure 5.9 Debug Configurations Dialog Box



4. Click the **Debugger** tab. The **Debugger** page appears in the area beneath the tabs.

Figure 5.10 Debug Configurations Dialog Box — Debugger Page



5. Change the settings on this page as per your requirements. For example, select the required target processor and simulator/emulator.
6. Click the **Apply** button to save the new settings.
7. Click the **Debug** button to start the debugging session.

You just finished starting a debugging session and attaching the debugger to a process.

NOTE You can click the **Revert** button to undo any of the unsaved changes. The IDE restores the last set of saved settings to all pages of the **Debug Configurations** dialog box. Also, the IDE disables the **Revert** button until you make new pending changes.

NOTE For information about debugger features and various targets and associated connections, refer *Microcontrollers V10.x Targeting Manual*.

Breakpoints

The different types of breakpoints that can be set on an executable line of source code are:

- Regular breakpoints — suspend the execution of a thread before a line of code or method is executed. Regular breakpoints include:
 - Line Breakpoint — suspends thread execution when the line of code it applies to, is executed.
 - Method Breakpoint — suspends execution when the method that it applies to, is entered.
- Conditional breakpoints — halt program execution when the condition you specify is met.
- Special breakpoints — halt program execution and then remove the breakpoint that caused the halt. A special breakpoint can be one of these types:
 - Software — The debugger sets a software breakpoint into target memory. When program execution reaches the breakpoint, the processor stops and activates the debugger. The breakpoint remains in the target memory until the user removes it. The breakpoint can only be set in writable memory like SRAM or DDR. You cannot use this type of breakpoints in ROM.
 - Hardware — Selecting the Hardware menu option causes the debugger to use the internal processor breakpoints. The number of hardware breakpoints available varies by processor type and can be used with all types of memories (ROM/RAM) because they are implemented by using processor registers.

Setting Regular Breakpoint

To set a line breakpoint at a line of source code:

1. Open the source code file in the editor and put the cursor in the line on which you want to set the breakpoint.
2. Right-click the marker bar next to the source code line and select **Toggle Line Breakpoint**. You can also double-click on the marker bar to set the breakpoint.

A breakpoint marker appears on the marker bar, directly to the left of the line where you added the breakpoint.

While the breakpoint is enabled, thread execution suspends before that line of code is executed. The debugger selects the suspended thread and displays its stack frames.

To set a method breakpoint:

1. Open the source code file in the editor.
2. Select **Window > Show View > Outline** from the IDE menu bar to open the **Outline** view.

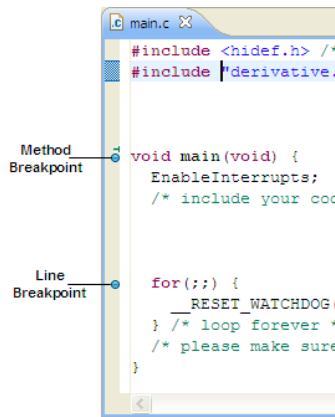
3. Select the method where you want to add the breakpoint.
4. Select **Run > Toggle Method Breakpoint** from the IDE menu bar. You can also select **Toggle Breakpoint** from the method's context menu.

A breakpoint appears on the marker bar next to the selected method.

While the breakpoint is enabled, thread execution suspends before the method is entered or exited.

[Figure 5.11](#) displays the line and method breakpoints in the editor area.

Figure 5.11 Regular Breakpoints



Setting Special Breakpoints

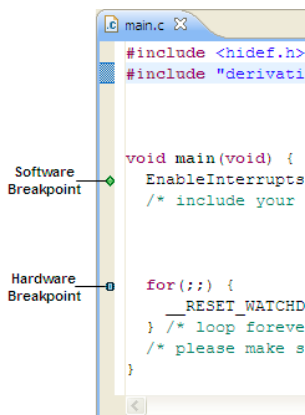
A special breakpoint is not a regular breakpoint and therefore, cannot be set by double clicking.

To set a special breakpoint:

1. Open the source code file in the editor and put the cursor in the line on which you want to set the special breakpoint.
2. Right-click the marker bar next to the source code line and select **Set Special Breakpoint > Software** or **Hardware**.

[Figure 5.12](#) displays the software and hardware breakpoints in the editor area.

Figure 5.12 Special Breakpoints



NOTE For more information about Breakpoints, such as viewing breakpoints in Breakpoints view and breakpoints actions, see Freescale Eclipse Extensions Guide.

Watchpoints

You use watchpoints (sometimes referred to as access breakpoints or memory breakpoints) to halt program execution when your program reads or writes to a specific memory location. You can then examine the call chain, check register and variable values, and step through your code. You can also change variable values and alter the flow of normal program execution.

Setting Watchpoints

You can set a watchpoint from the:

- Add Watchpoint dialog box
- Breakpoints view
- Memory view
- Variables view

NOTE Not all targets support setting a watchpoint on a memory range. For example, if a target has only one or two debug watch registers, you cannot set a watchpoint on 50 bytes.

To set a watchpoint:

1. Open the Debug perspective.
2. Click one of these tabs:
 - Breakpoints
 - Memory
 - Variables

The selected view comes forward.

NOTE If the desired view does not appear in the Debug perspective, select **Window > Show View** and select the view to display it in the Debug perspective.

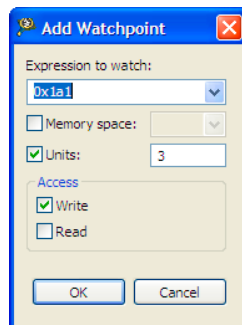
3. Right-click within the selected view.

The part of the view in which to right-click varies depending upon the type of the view:

- Breakpoints — an empty area inside the view.
 - Memory — the addressable unit or range of units on which you want to set the watchpoint.
 - Variables — a global variable.
4. Select **Add Watchpoint (C/C++)** from the context menu that appears.

The **Add Watchpoint** dialog box appears.

Figure 5.13 Add Watchpoint Dialog Box



5. Make the required settings in the **Add Watchpoint** dialog box and click **OK**.

The **Breakpoints** view shows information about the newly set watchpoint and the number of addressable units that the watchpoint monitors.

The **Problems** view shows error messages if the debugger fails to set a watchpoint.

Creating and Debugging Projects

Debugging Projects

NOTE For information about the options in the **Add Watchpoint** dialog box, refer to the *FreescalEclipse Extensions Guide*.

Index

A

- Add Files Page 51
- Add or remove command groups from perspective 36
- Add or remove shortcuts from submenus 36
- Add Watchpoint dialog box 61
- Apply 57

B

- board 49
- Breakpoints 58

C

- C/C++ perspective 28
 - views
 - CodeWarrior Projects 29
 - Console 29
 - Make Targets 30
 - Outline 30
 - Problems 29
 - Properties 29
 - Tasks 29
- Cheat Sheets
 - Click to Begin 27
 - Click to perform 27
 - Click to redo 27
 - Click to Restart 27
 - Click to skip 27
 - Click when complete 27
 - Skip this task 27
- CodeWarrior Development Process
 - Compiling 16
 - Debugging 16
 - Editing Code 16
 - Linking 16
 - Project Files 15
- CodeWarrior Project View 55
- CodeWarrior Projects 54
- Conditional Breakpoints 58
- Connections Page 50
- Create an MCU Bareboard Project Page 48

- Creating New Bareboard Project 47
- Creating New MCU Project 47
- Creating Projects 47
- CWInstallDir 6

D

- Debug 57
- Debug Dialog Box 56
 - Debugger Page 57
- Debug perspective 30
 - views
 - Breakpoints 31
 - Debug 31
 - Debugger Shell 31
 - Disassembly 31
 - Expressions 32
 - Memory 31
 - Modules 31
 - Registers 31
 - Target Tasks 32
 - Variables 31
- Debugger Page 57
- Debugger tab 56
- Debugging Projects 55
- default workspace 47
- deleting a perspective 37
- derivative 49
- Device and Connection Page 49
- Disk Space 14

E

- Eclipse IDE Introduction 13
- editor 37

H

- Hardware Breakpoint 58
- Hardware Requirements 14
- Help 25
- Help Documentation
 - Cheat Sheets 26
 - Eclipse Help 25

how to

- create a fast view 45
- delete perspective 37
- open a new perspective 35
- open files for editing 37
- open views 33
- restore perspective 37
- save perspective 36
- set a hardware breakpoint 59
- set a line breakpoint 58
- set a method breakpoint 58
- set a software breakpoint 59
- set a special breakpoint 59
- set a watchpoint 60
- switch between perspectives 35

L

- launch a cheat sheet 26
- Line Breakpoint 58

M

markers

- Bookmark 39
- Problems
 - Error 39
 - Information 39
 - Warning 39
- Task 39

Method Breakpoint 58

Microcontrollers New Bareboard Project wizard 47

N

New Bareboard Project Wizard

- Add Files Page 51
- Connections Page 50
- Create an MCU Bareboard Project Page 48
- Device and Connection Page 49
- Processor Expert MCU Pin Variants and Configuration page 54

New Project Wizard 48

O

- opening files for editing 37
- opening or switching between perspective 35
- opening views 33
- Operating System 14

P

- Perspective and View 28
- perspective switcher commands
 - Open Perspective 44
 - Perspectives 44
 - Show List 44
- Processor Expert MCU Pin Variants and Configuration Page 54

R

- Regular Breakpoints 58
 - Line Breakpoint 58
 - Method Breakpoint 58
- release notes 6
- restoring a perspective 37
- Revert unsaved changes in Debug dialog box 57

S

- saving a perspective 36
- setting a hardware breakpoint 59
- setting a line breakpoint 58
- setting a method breakpoint 58
- setting a software breakpoint 59
- setting a special breakpoint 59
- setting a watchpoint 60
- Software Breakpoint 58
- Special Breakpoints 58
- standalone and tabbed views 32
- System Requirements 14

T

- toolbars in Workbench 39
 - Fast View 45
 - perspective switcher 43
 - trim stack 44
 - view toolbar 43
 - Workbench toolbar 40

U

- using view menus
 - view pull-down menu 34
 - view tab context menu 34

V

- view pull-down menu 35
- View Stack 22
- view tab context menu commands
 - Close 35
 - Detached 34
 - Fast View 34
 - Maximize 35
 - Minimize 34
 - Move 34
 - Restore 34
 - Size 34
- view's tab context menu 34

W

- Watchpoints 60
- Welcome Page 22
 - Example Projects 23
 - Go to Workbench 24
 - New Project Wizard 23
 - Project Importer 24
 - Tutorials 24
 - Web Resources 24
 - What's New 24
- Workbench 20
- Workbench Components
 - Editor 22
 - Toolbars 22
 - View 22
- Workbench toolbar commands
 - All Results 41
 - Back 43
 - Build All 40
 - Build Selected 42
 - Critical Code 41
 - Debug 42
 - External Tools 42
 - Forward 43

- Last Edit Location 43
- Manage configurations 42
- New 40
- New C++ Class 42
- New C/C++ Project 41
- New C/C++ Source File 42
- New C/C++ Source Folder 41
- Next Annotation 43
- Open Element 42
- Previous Annotation 43
- Print 40
- Run 42
- Save 40
- Search 42
- Show Source of Selected Element Only 42
- Show Whitespace Characters 42
- Trace 40
- working with markers 38
- working with perspectives
 - deleting a perspective 37
 - opening or switching between perspective 35
 - restoring a perspective 37
 - saving a perspective 36
- working with views 32
 - opening views 33
 - using view menus 34
- Workspace 19

