

# CodeWarrior®

## IDE User Guide



*Because of last-minute changes to CodeWarrior, some parts of this manual may be out of date. Please read all the Release Notes files that come with CodeWarrior to get important last minute information.*

# Copyright

Metrowerks CodeWarrior Copyright ©1993-1996 by Metrowerks Inc. and its Licensors. All rights reserved.

Documentation stored on the compact disk(s) may be printed by licensee for personal use. Except for the foregoing, no part of this documentation may be reproduced or transmitted in any form by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from Metrowerks Inc.

Metrowerks, the Metrowerks logo, CodeWarrior, and Software at Work are registered trademarks of Metrowerks Inc. PowerPlant and PowerPlant Constructor are trademarks of Metrowerks Inc.

All other trademarks and registered trademarks are the property of their respective owners.

ALL SOFTWARE AND DOCUMENTATION ON THE COMPACT DISK(S) ARE SUBJECT TO THE LICENSE AGREEMENT IN THE CD BOOKLET.

## How to Contact Metrowerks

<b>U. S. A. and international:</b>	Metrowerks Corporation 2201 Donley Drive, suite 310, Austin, TX 78758 U. S. A.
<b>Canada:</b>	Metrowerks Inc. 1500 du College, suite 300, Ville St-Laurent, QC Canada H4L 5G6
<b>Metrowerks Mail Order:</b>	voice: 800 377-5416 fax: 512 873-4901
<b>World Wide Web:</b>	<a href="http://www.metrowerks.com">http://www.metrowerks.com</a>
<b>Registration information:</b>	<a href="mailto:register@metrowerks.com">register@metrowerks.com</a>
<b>Technical support:</b>	<a href="mailto:support@metrowerks.com">support@metrowerks.com</a>
<b>Sales, marketing, &amp; licensing:</b>	<a href="mailto:sales@metrowerks.com">sales@metrowerks.com</a>
<b>AppleLink:</b>	METROWERKS
<b>America OnLine:</b>	keyword: Metrowerks
<b>Compuserve:</b>	goto Metrowerks

# Table of Contents

---

<b>1 Introduction . . . . .</b>	<b>17</b>
Introduction . . . . .	17
Read the Release Notes! . . . . .	19
IDE User Guide Overview . . . . .	19
About the CodeWarrior IDE . . . . .	20
Where to Go From Here . . . . .	20
QuickStart and Tutorial Resources . . . . .	21
Operating System Targeting Documentation . . . . .	21
<b>2 Getting Started . . . . .</b>	<b>23</b>
Getting Started Overview . . . . .	23
System Requirements . . . . .	23
CodeWarrior IDE Installation. . . . .	24
CodeWarrior IDE Guided Tour . . . . .	24
Initial View of the IDE . . . . .	24
IDE Menus . . . . .	25
File Menu. . . . .	26
Edit Menu . . . . .	26
Search Menu . . . . .	27
Project Menu . . . . .	28
Tools Menu . . . . .	29
Window Menu . . . . .	30
Help Menu . . . . .	31
IDE Toolbar . . . . .	31
IDE Toolbar Overview . . . . .	31
Accessing Additional Commands . . . . .	32
Other IDE Components . . . . .	33
<b>3 Working with Projects . . . . .</b>	<b>35</b>
Projects Overview . . . . .	35
Creating a Project. . . . .	36
About Project Stationery . . . . .	36
Using Stationery Projects . . . . .	37
Choosing the New Project Stationery File . . . . .	37

---

Naming Your New Project . . . . .	38
Modifying Your New Project . . . . .	40
Building Your New Project . . . . .	41
About the Project Stationery Folder . . . . .	41
Creating Your Own Project Stationery . . . . .	41
Opening an Existing Project . . . . .	43
Using the Open Command . . . . .	44
Using the Project Switch List . . . . .	44
Saving a Project . . . . .	45
Items Saved with Your Project . . . . .	46
Saving a Copy of Your Project . . . . .	46
Closing a Project . . . . .	46
Guided Tour of the Project Window . . . . .	47
Navigating the Project Window . . . . .	47
Project Window User Interface Items . . . . .	47
Group Organization . . . . .	48
File Column . . . . .	49
Code Column . . . . .	49
Data Column . . . . .	49
Debug Column . . . . .	50
Touch Column . . . . .	50
Interfaces File Pop-up . . . . .	50
Group File Pop-up . . . . .	51
Managing Files in a Project . . . . .	51
About Groups and Segments . . . . .	51
Selecting Files and Groups . . . . .	52
Selection by mouse-clicking . . . . .	53
Selection by keyboard . . . . .	54
Expanding and Collapsing Groups . . . . .	54
Adding Files . . . . .	55
Where Files Appear . . . . .	56
Using the Add Files Command . . . . .	56
Using the Add Window Command . . . . .	57
Moving Files and Groups . . . . .	58
Creating Groups . . . . .	59
Removing Files and Groups . . . . .	60

---

Renaming Groups . . . . .	61
Touching and Untouching Files . . . . .	62
Synchronizing modification dates . . . . .	63
Moving a Project . . . . .	64
Resource.frk and Your Project . . . . .	64
Controlling Debugging in a Project . . . . .	64
Activating Debugging for a Project . . . . .	65
Activating Debugging for a File . . . . .	65
Debug Info Marker for Groups . . . . .	66
Adding Preprocessor Symbols to a Project . . . . .	66
<b>4 Working with Files . . . . .</b>	<b>69</b>
Working with Files Overview . . . . .	69
Creating a New File . . . . .	69
Opening an Existing File. . . . .	70
Opening Files with the File Menu . . . . .	70
Project File . . . . .	70
Text File . . . . .	71
Opening Files from the Project Window . . . . .	71
File Column. . . . .	72
Group File Pop-up Menu . . . . .	72
Interfaces Files Pop-up Menu . . . . .	73
Opening Files from an Editor Window . . . . .	74
Saving a File . . . . .	75
Saving One File . . . . .	76
Saving All Files . . . . .	76
Saving Files Automatically . . . . .	76
Renaming and Saving a File . . . . .	77
Backing Up Files. . . . .	78
Saving as a UNIX or DOS text file . . . . .	79
Closing a File . . . . .	79
Closing One File . . . . .	80
Closing All Files . . . . .	80
Printing a File . . . . .	81
Setting Print Options . . . . .	81
Printing a Window . . . . .	82

---

Reverting to a Previously-Saved File . . . . .	83
--	----

<b>5 Editing Source Code . . . . .</b>	<b>85</b>
--	-----------

Source Code Editor Overview . . . . .	85
Guided Tour of the Editor Window . . . . .	85
Text Editing Area. . . . .	87
Interface Pop-Up Menu . . . . .	87
Routine Pop-Up Menu . . . . .	88
Marker Pop-Up Menu . . . . .	90
Options Pop-Up Menu . . . . .	90
File Path Caption. . . . .	91
File Privileges Icons . . . . .	92
Line Number Button . . . . .	92
Pane Splitter Controls. . . . .	92
Pop-Up Menu Disclosure Button . . . . .	93
Editor Window Configuration . . . . .	93
Setting Text Size and Font . . . . .	93
Seeing Window Controls . . . . .	93
Splitting the Window into Panes . . . . .	95
Creating a new pane . . . . .	96
Resizing a pane . . . . .	97
Removing a pane . . . . .	97
Saving Window Settings . . . . .	97
Basic Text Editing. . . . .	98
Basic Editor Window Navigation. . . . .	98
Scroll bar navigation . . . . .	98
Keyboard navigation . . . . .	99
Adding Text . . . . .	100
Deleting Text . . . . .	100
Selecting Text . . . . .	101
Moving Text (drag and drop) . . . . .	102
Figure 5.13 shows the final result after dragging the text selection down to a new line. . . . .	105
Using Cut, Copy, Paste, and Clear . . . . .	105
Balancing Punctuation . . . . .	106
Using automatic balancing . . . . .	106

---

Shifting Text Left and Right . . . . .	106
Undoing Changes . . . . .	107
Undoing the last edit . . . . .	107
Undoing and redoing multiple edits . . . . .	107
Reverting to the last saved version of a file . . . . .	108
Controlling Color . . . . .	108
Navigating in Text . . . . .	108
Finding a Routine . . . . .	109
Adding, Removing, and Selecting a Marker . . . . .	109
Adding a Marker . . . . .	109
Removing a Marker . . . . .	112
Jumping to a Marker . . . . .	112
Opening a Related File . . . . .	113
Going to a Particular Line . . . . .	114
Using Go Back and Go Forward . . . . .	114

## **6 Searching and Replacing Text . . . . . 115**

Searching and Replacing Text Overview . . . . .	115
Guided Tour of the Find Window . . . . .	115
Search and Replace Section . . . . .	116
Find Text Box . . . . .	117
Replace Text Box . . . . .	117
Recent Strings Pop-Up Menu . . . . .	117
Find Button . . . . .	118
Replace Button . . . . .	118
Replace & Find Button . . . . .	119
Replace All Button . . . . .	119
Batch Check Box . . . . .	119
Wrap Check Box . . . . .	119
Ignore Case Check Box . . . . .	119
Entire Word Check Box . . . . .	120
Regex Check Box . . . . .	120
Multi-File Search Disclosure Triangle . . . . .	120
Multi-File Search Button . . . . .	120
Multi-File Search Section . . . . .	121
File Sets Pop-Up Menu . . . . .	122

---

File Sets List. . . . .	123
Stop at End of File Check Box . . . . .	123
Sources Check Box . . . . .	124
System Headers Check Box . . . . .	124
Project Headers Check Box . . . . .	124
Others Button . . . . .	124
Searching for Selected Text. . . . .	125
Finding text in the active Editor window . . . . .	125
Finding text in another window . . . . .	126
Searching and Replacing Text in a Single File . . . . .	126
Finding Search Text. . . . .	127
Controlling Search Range . . . . .	129
Controlling Search Parameters . . . . .	129
Ignore Case Check Box . . . . .	129
Entire Word Check Box . . . . .	130
Replacing Found Text. . . . .	130
Replace All . . . . .	130
Selective Replace. . . . .	130
Using Batch Searches . . . . .	132
Searching and Replacing Text in Multiple Files . . . . .	133
Activating Multi-File Search . . . . .	133
Choosing Files to be Searched . . . . .	134
Adding project source files . . . . .	135
Adding project header files . . . . .	135
Adding system header files . . . . .	135
Adding and removing arbitrary files . . . . .	135
Choosing a file set . . . . .	137
Saving a File Set . . . . .	137
Removing a File Set. . . . .	138
Controlling Search Range . . . . .	139
Using Regular Expressions (grep). . . . .	140
Matching simple expressions . . . . .	140
Matching any character. . . . .	141
Repeating expressions . . . . .	141
Grouping expressions . . . . .	141
Choosing one character from many. . . . .	142



---

Matching the beginning or end of a line. . . . .	143
Using the Find string in the Replace string . . . . .	143
Remembering sub-expressions. . . . .	143
<b>7 Browsing Source Code . . . . .</b>	<b>145</b>
Browser Overview . . . . .	145
Activating the Browser . . . . .	146
Understanding the Browser Strategy . . . . .	146
Catalog View . . . . .	147
Browser View . . . . .	148
Hierarchy View . . . . .	149
Guided Tour of the Browser . . . . .	150
Catalog Window . . . . .	151
Category pop-up menu . . . . .	152
Symbols pane . . . . .	153
Multi-Class Browser Window . . . . .	153
Orientation button . . . . .	155
List button . . . . .	156
File button . . . . .	156
Classes pane . . . . .	156
Member Functions pane . . . . .	157
Data Members pane . . . . .	157
Source pane . . . . .	158
Resize bar. . . . .	158
Identifier icon . . . . .	158
Single-Class Browser Window . . . . .	159
Bases text field. . . . .	160
Show check boxes . . . . .	160
Show Declaration button . . . . .	160
Show Hierarchy button . . . . .	160
Multi-Class Hierarchy Window . . . . .	161
Line button . . . . .	162
Hierarchy expansion triangle . . . . .	162
Ancestor Class pop-up menu . . . . .	163
Single-Class Hierarchy Window . . . . .	163
Symbol Window . . . . .	164

---

Function Symbols pane . . . . .	166
Context Pop-Up Menu . . . . .	166
Using the Browser . . . . .	168
Setting Browser Options . . . . .	168
Navigating Code in the Browser . . . . .	169
Using the Context Pop-Up Menu. . . . .	169
Go Back and Go Forward . . . . .	169
Opening a Source File. . . . .	171
Seeing a Declaration . . . . .	171
Seeing a Function Definition . . . . .	171
Editing Code in the Browser . . . . .	172
Analyzing Inheritance . . . . .	172
Finding Functions That Are Overrides . . . . .	173
Seeing MFC Classes . . . . .	173
Saving a Default Browser . . . . .	174

## **8 Configuring IDE Options . . . . . 175**

Configuring IDE Options Overview. . . . .	175
Option Dialogs Guided Tour . . . . .	176
Options Panels. . . . .	176
Property Sheet Dialog Buttons . . . . .	179
Apply button . . . . .	179
Cancel button . . . . .	180
OK button . . . . .	180
Choosing Preferences . . . . .	180
Editor Settings Panel . . . . .	180
Dynamic Scroll . . . . .	181
Balance While Typing . . . . .	181
Save All Before “Update” . . . . .	182
Use Multiple Undo. . . . .	182
Flashing Delay . . . . .	182
Context Popup Delay. . . . .	182
Drag and Drop Editing . . . . .	183
Sort Function Popup . . . . .	183
Font Preferences . . . . .	183
Window Position and Size . . . . .	183

---

Selection Position . . . . .	183
Projector Aware . . . . .	183
Main Text Color . . . . .	184
Background Color . . . . .	184
Fonts and Tabs Panel . . . . .	184
Syntax Coloring Panel . . . . .	185
Changing syntax highlighting colors . . . . .	187
Controlling syntax highlighting within a window. . . . .	188
Using color for custom keywords . . . . .	188
Importing or exporting custom keywords . . . . .	189
Browser Coloring Panel . . . . .	190
Choosing Project Settings . . . . .	191
Target Panel . . . . .	192
Target . . . . .	193
Post Linker . . . . .	193
File Name List. . . . .	193
Extension . . . . .	194
Compiler . . . . .	194
Precompiled . . . . .	194
Launchable . . . . .	194
Resource File . . . . .	194
Ignored by Make. . . . .	195
Access Paths Panel . . . . .	195
Treat #include <...> as #include "...". . . . .	196
User Include Path Pane . . . . .	196
System Include Path Pane. . . . .	196
Add Default. . . . .	197
Add . . . . .	197
Change. . . . .	198
Build Extras . . . . .	199
Use Modification Date Caching . . . . .	200
Activate Browser . . . . .	200
Generate Make Map File . . . . .	200
Store Analysis Results . . . . .	200
Custom Keywords Panel . . . . .	200
C/C++ Compiler Panel . . . . .	201

---

---

C/C++ Warnings Panel . . . . .	202
Pascal Compiler . . . . .	203
Pascal Warnings . . . . .	204
x86 Project Panel . . . . .	205
x86 CodeGen Panel . . . . .	206
x86 Linker Panel . . . . .	207
IR Optimizer Panel . . . . .	208
WinRC Compiler . . . . .	209

## **9 Compiling and Linking . . . . . 211**

Compiling and Linking Projects Overview . . . . .	211
Choosing a compiler . . . . .	212
Understanding Plugin Compilers. . . . .	212
Setting a File Extension . . . . .	212
Compiling and Linking a Project . . . . .	213
Touching and Untouching Files . . . . .	213
Compiling Files . . . . .	213
Compiling One File . . . . .	214
Compiling Selected Files . . . . .	214
Recompiling Files . . . . .	215
Updating a Project . . . . .	215
Making a Project . . . . .	215
Enabling Debugging . . . . .	216
Running a Project . . . . .	216
Debugging a Project . . . . .	217
Generating a Link Map . . . . .	218
Synchronizing Modification Dates . . . . .	218
Removing Binaries . . . . .	218
Removing Object Code . . . . .	219
Remove Binaries & Compact command . . . . .	219
Advanced Compile Options . . . . .	219
Alerting Yourself After a Build. . . . .	220
Speeding Up a Build by Avoiding Date Checks . . . . .	220
Using Precompiled or Preprocessed Headers . . . . .	220
Creating Precompiled Headers. . . . .	221
Precompile command . . . . .	222

---

Automatic updating . . . . .	222
Defining Symbols For C/C++ . . . . .	224
Defining Symbols For Pascal. . . . .	225
Preprocessing Source Code (C/C++ only) . . . . .	226
Disassembling Source Code . . . . .	228
Guided Tour of the Message Window . . . . .	229
Error Button . . . . .	231
Warning Button . . . . .	232
Project Information Caption . . . . .	232
Stepping Buttons. . . . .	232
Message List Pane . . . . .	232
Source Code Disclosure Triangle . . . . .	232
Source Code Pane . . . . .	233
Pane Resize Bar . . . . .	233
Pop-Up Menu Disclosure Button . . . . .	233
Interface Pop-Up Menu . . . . .	233
Routine Pop-Up Menu . . . . .	233
File Path Caption. . . . .	233
Line Number Button . . . . .	234
Using the Message Window . . . . .	234
Seeing Errors and Warnings . . . . .	235
Stepping Through Messages . . . . .	236
Correcting Compiler Errors and Warnings . . . . .	238
Correcting Errors in the Source Code Pane . . . . .	238
Opening the File for the Corresponding Message. . . . .	239
Correcting Linker Errors . . . . .	239
Correcting Pascal Circular References . . . . .	241
Saving and Printing the Message Window . . . . .	242
Locating Errors in Modified Files . . . . .	243
<b>10 IDE Menu Reference . . . . .</b>	<b>245</b>
IDE Menu Reference Overview . . . . .	245
File Menu . . . . .	245
New . . . . .	245
New Project. . . . .	246
Open. . . . .	246

---

---

Open Selection . . . . .	246
Open File . . . . .	246
Close . . . . .	246
Close All . . . . .	246
Switch to Debugger . . . . .	247
Save . . . . .	247
Save All . . . . .	247
Save As. . . . .	247
Save A Copy As . . . . .	247
Revert . . . . .	247
Print Setup . . . . .	248
Print . . . . .	248
Exit . . . . .	248
Edit Menu . . . . .	248
Undo. . . . .	248
Redo . . . . .	249
Multiple Undo. . . . .	249
Cut . . . . .	249
Copy . . . . .	250
Paste . . . . .	250
Clear . . . . .	250
Select All . . . . .	250
Balance . . . . .	250
Shift Left . . . . .	251
Shift Right . . . . .	251
Insert Reference Template. . . . .	251
Preferences . . . . .	251
Project Settings . . . . .	251
Search Menu . . . . .	251
Find . . . . .	252
Find Next. . . . .	252
Find Previous . . . . .	252
Find in Next File. . . . .	252
Find in Previous File . . . . .	252
Enter 'Find' String . . . . .	253
Enter 'Replace' String. . . . .	253

---

Find Selection . . . . .	253
Find Previous Selection . . . . .	253
Replace . . . . .	254
Replace & Find Next . . . . .	254
Replace & Find Previous . . . . .	254
Replace All . . . . .	254
Find Definition & Reference . . . . .	255
Find Reference . . . . .	255
Find Definition . . . . .	255
Go Back . . . . .	255
Go Forward . . . . .	256
Go To Line . . . . .	256
Project Menu . . . . .	256
Add Window . . . . .	256
Add Files . . . . .	256
Remove Files . . . . .	257
Reset File Paths . . . . .	257
Synchronize Modification Dates . . . . .	257
Check Syntax . . . . .	257
Preprocess . . . . .	258
Precompile . . . . .	258
Compile . . . . .	258
Disassemble. . . . .	258
Remove Binaries. . . . .	259
Remove Binaries & Compact . . . . .	259
Bring Up To Date . . . . .	259
Make. . . . .	259
Enable Debugger . . . . .	260
Disable Debugger . . . . .	260
Run . . . . .	260
Debug . . . . .	260
Tools Menu . . . . .	261
Show Toolbar . . . . .	261
Hide Toolbar . . . . .	261
Anchor Toolbar . . . . .	261
Unanchor Toolbar . . . . .	261

---

---

Reset Toolbar . . . . .	262
Clear Toolbar . . . . .	262
Window Menu . . . . .	262
Stack . . . . .	262
Tile . . . . .	262
Tile Vertical . . . . .	262
Zoom Window . . . . .	263
Save Default Window . . . . .	263
Show Catalog Window . . . . .	263
Show Hierarchy Window . . . . .	263
New Class Browser . . . . .	263
Project Switch List . . . . .	264
Errors & Warnings Window . . . . .	264
Other Window Menu Items . . . . .	264
Help Menu . . . . .	265
Contents . . . . .	265
Keys . . . . .	265
How to Use Help . . . . .	265
About Metrowerks . . . . .	265
<b>Index . . . . .</b>	<b>267</b>





# Introduction

---

This manual describes the CodeWarrior Integrated Development Environment (IDE) in detail. The IDE is used to develop code for various operating systems, using various programming languages.

## Introduction

This section introduces the CodeWarrior IDE. The topics in this chapter are:

- Read the Release Notes!
- IDE User Guide Overview
- About the CodeWarrior IDE
- Where to Go From Here

The IDE is a collection of development tools for creating and generating code for the systems listed in Table 1.1.

**Table 1.1 CodeWarrior IDE Targets**

Target	Description
68K Macintosh	Any Mac OS computer that uses the Motorola 68000 family of microprocessors
Power Macintosh	Any Mac OS computer that uses a PowerPC microprocessor
Win32/x86	The Win32 model for Windows NT 4.0 and Windows 95 on 80x86-class processors
Java	The Java virtual machine
Magic Cap	The Magic Cap operating system
BeOS	The Be operating system

## Introduction

### Introduction

---

Target	Description
PlayStation	The PlayStation game console
PowerTV OS	The PowerTV digital set-top box operating system
Palm OS	The operating system for the Pilot connected organizer



**NOTE:** *Your version of CodeWarrior does not contain compilers for all possible targets listed in Table 1.1. Check your product description for targets available to you.*

---

You use the same IDE when developing code for all these systems. You add new compilers and linkers by dragging specially-designed plug-ins into your Plugins folder. The Plugins folder contains separate subdirectories for Compilers, Linkers, Post Linkers and Preference Panels.

You can develop applications using these languages:

- Object Pascal, a compiler for ANS Pascal and Object Pascal, which supports Turbo Pascal Input/Output routines, conditional compilation and macros, extended debugging features, and inline assembly code.
- C and C++ , a compiler that implements templates, exception handling, run-time type information (RTTI), and inline assembly code.
- Java, for Java virtual machines.



**NOTE:** *Some languages (such as Java) can only be used for certain targets.*

---

## Read the Release Notes!

By now, you probably have read the CodeWarrior release notes. If you haven't, please do so. They contain important information about new features, bug fixes, and incompatibilities that may not have made it into the documentation due to release deadlines. You can find them on both CodeWarrior CDs, in the Release Notes folder.

## IDE User Guide Overview

There are several chapters in the User Guide. Each chapter begins with an overview of the topics discussed in that chapter. The chapter overviews are:

- Introduction — (this chapter) contains an overview of the CodeWarrior IDE languages, platforms, and documentation
- Getting Started Overview— system requirements, installation, guided tour of the user interface
- Projects Overview— introduces the CodeWarrior Project Window, shows how to setup, configure, and work with projects
- Working with Files Overview— introduces the concepts behind working with files in CodeWarrior
- Source Code Editor Overview— explains how to use the CodeWarrior text editor
- Searching and Replacing Text Overview— explains how to use the CodeWarrior facilities to search and replace text in files
- Browser Overview— describes Code Warrior's class browser, a tool you use to examine your project source code from various perspectives
- Configuring IDE Options Overview— discusses the many options available in CodeWarrior's Preferences and Project Settings dialogs
- Compiling and Linking Projects Overview— discusses how to control compilation, linking, and running a CodeWarrior project

## Introduction

*About the CodeWarrior IDE*

---

- IDE Menu Reference Overview— describes each command on each CodeWarrior IDE menu

## About the CodeWarrior IDE

The CodeWarrior IDE is a collection of tools that allows you to develop computer code for many different platforms using different programming languages. Using the IDE, you can develop a program, plug-in, library, or other executable code to run on a computer system.

The CodeWarrior IDE permits a code developer to quickly assemble a variety of source code files (for example, a file written in the C++ computer language), resource files, and library files into a project, without writing a complicated build script (or “Makefile”) for the project. Source code files may be added or deleted from the IDE’s project using simple mouse and keyboard operations instead of editing a build script. Tools such as a debugger, compilers, linkers, a code browser, and source code editor are included with CodeWarrior. These tools allow you to edit your code, navigate and browse your code, compile it, link it, and debug it until you have a running application. Options for code generation, debugging, and navigation of your project are all configurable in the IDE.

The CodeWarrior product comes with everything you need to develop code!

## Where to Go From Here

There are a few different options for where to start reading more about developing with CodeWarrior.

When you are ready to debug, be sure to read the *CodeWarrior Debugger Manual* on the CodeWarrior Reference CD.

When you are trying to get started quickly with a new platform or if you are new to CodeWarrior, see “QuickStart and Tutorial Resources” on page 21.

To get started quickly with a new target operating system, see “Operating System Targeting Documentation” on page 21.

## **QuickStart and Tutorial Resources**

You will find all the manuals mentioned in this section in the CodeWarrior Documentation folder on the CodeWarrior CD. For some products this will be on the CodeWarrior Reference CD.

If you are new to CodeWarrior, check out these resources:

- The *CodeWarrior QuickStart Guide* for an overview of CodeWarrior, descriptions of some CodeWarrior windows and shortcuts, and pointers to the references available to you. *Quick Start* is on your CD, so you can use it regardless of whether you purchased any printed documentation.
- “CodeWarrior IDE Guided Tour” on page 24 provides a quick overview of the CodeWarrior user interface.
- The CW Tutorials Code folder on the CodeWarrior CD contains some sample projects that will help you become productive quickly with CodeWarrior.
- The instructions for using the viewers Metrowerks provides for on-line documentation in the CodeWarrior Documentation folder on the CodeWarrior CD.

## **Operating System Targeting Documentation**

When you are developing code with CodeWarrior, you “target” a specific operating system. This means that you select a target operating system on which you want your finished code to run. Many different targets are available for your CodeWarrior product.

To target a specific operating system for your code, consult the guides described in Table 1.2.

## Introduction

*Where to Go From Here*

---

**Table 1.2    Targeting Guides for Various CodeWarrior Targets**

Targets	Targeting Manual
Mac OS	<i>Targeting Mac OS</i>
Win32/x86	<i>Targeting Win32</i>
Java	<i>Targeting Java</i>
PlayStation OS	<i>Targeting PlayStation OS</i>
General Magic's Magic Cap	<i>Targeting Magic Cap</i>
BeOS	<i>CodeWarrior BeIDE User's Guide, and the manuals supplied by Be, Inc.</i>
PowerTV	<i>Targeting PowerTV</i>
Palm OS	<i>Targeting Palm OS</i>



# Getting Started

---

This chapter gives you the information you need to get started with CodeWarrior. You'll find the system requirements and information about installing the software, as well as a guided overview of the CodeWarrior IDE user interface.

## Getting Started Overview

The sections in this chapter are:

- System Requirements
- CodeWarrior IDE Installation
- CodeWarrior IDE Guided Tour



---

**TIP:** For a quick look at the CodeWarrior IDE user interface, go to “CodeWarrior IDE Guided Tour” on page 24. The tour gives you your first glimpse of the CW IDE interface, particularly the IDE Toolbar.

---

## System Requirements

The Windows-hosted version of CodeWarrior requires a 80486DX or Pentium™-class processor, 16 megabytes of RAM, Microsoft Windows 95 or Windows NT 4.0 operating system, and a CD-ROM drive to install the software.

For optimum performance, we recommend that you use a computer equipped with a Pentium™-class processor with at least 24 megabytes of RAM.

## CodeWarrior IDE Installation

To learn how to install the CodeWarrior product, read the Quick-Start guide on the CodeWarrior CD. When you install CodeWarrior, you automatically install the CodeWarrior Integrated Development Environment (IDE), including the CodeWarrior tools, languages, and debugger.

All compilers and targets available in CodeWarrior use the same IDE. This manual provides information for all items in the IDE, noting platform-specific items when applicable.

## CodeWarrior IDE Guided Tour

This section gives an overview of the CodeWarrior IDE user interface. In this section you'll find:

- Initial View of the IDE
- IDE Menus
- IDE Toolbar
- Other IDE Components

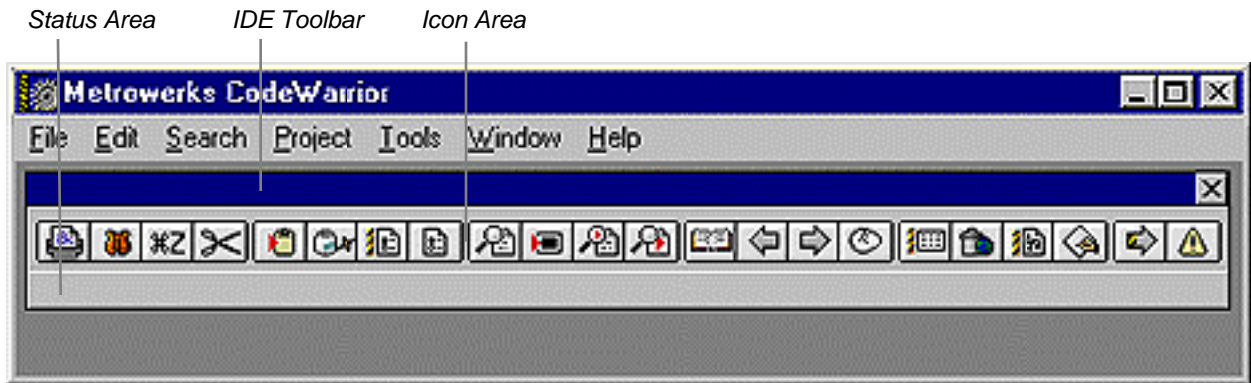
### Initial View of the IDE

When you launch CodeWarrior, you see the menu bar at the top of the CodeWarrior multiple-document interface (MDI) window, with the IDE Toolbar window below it, as shown in Figure 2.1.

You use the menus and icons on the IDE Toolbar to access CodeWarrior tools and commands.



**Figure 2.1 The CodeWarrior Window**



Because the menu commands are so extensive, this chapter describes each menu only briefly. Information about the various menu options can be found throughout this manual. If you'd like to look up what a specific menu item does, you can refer to "IDE Menu Reference Overview" on page 245.

The IDE Toolbar section in this chapter describes each icon on the Toolbar and tells you how to use it.

## IDE Menus

The CodeWarrior IDE menus allow you to perform a large number of various tasks to accomplish your work.



---

**For Beginners:** *Depending on your operating system and preference choices, some items on some menus may be disabled. These commands are not available for use with your current procedure or with your current target.*

---

The following sections describe each menu briefly. You'll find a detailed reference for every command on each menu starting at "IDE Menu Reference Overview" on page 245.

## Getting Started

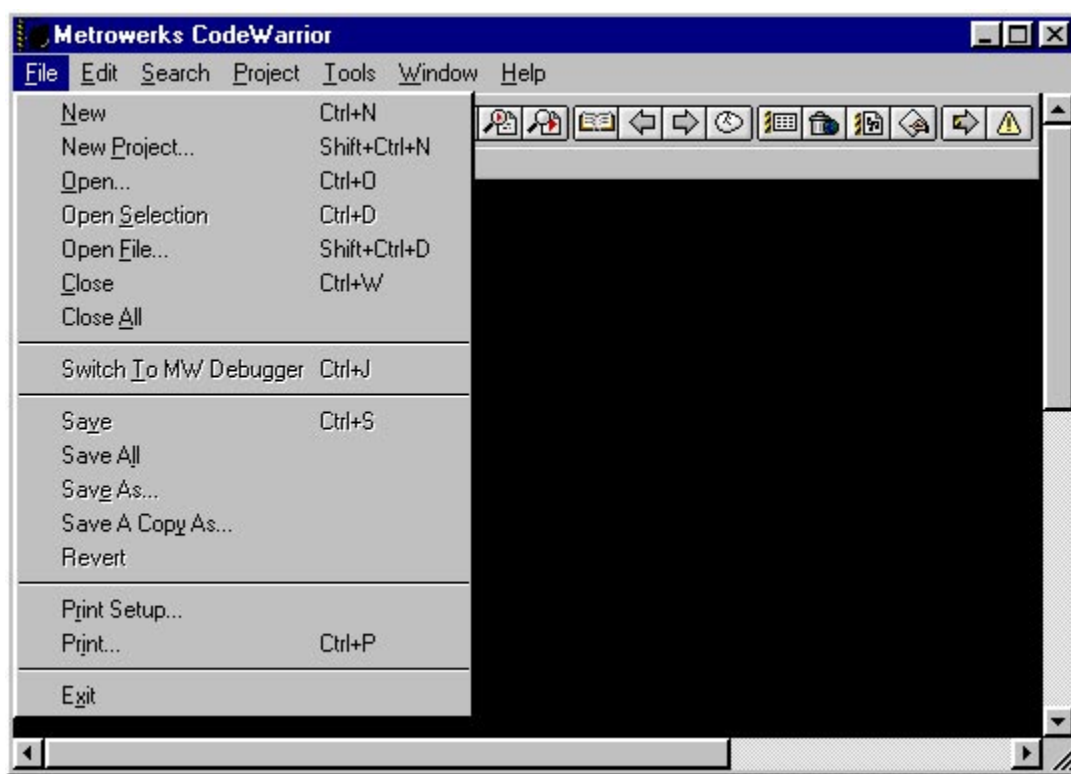
### CodeWarrior IDE Guided Tour

---

#### File Menu

The File Menu contains all the necessary commands used to create new and open existing source code files and projects. The File Menu also provides a few additional methods for saving edited files, as well as commands for switching to the debugger and printing files.

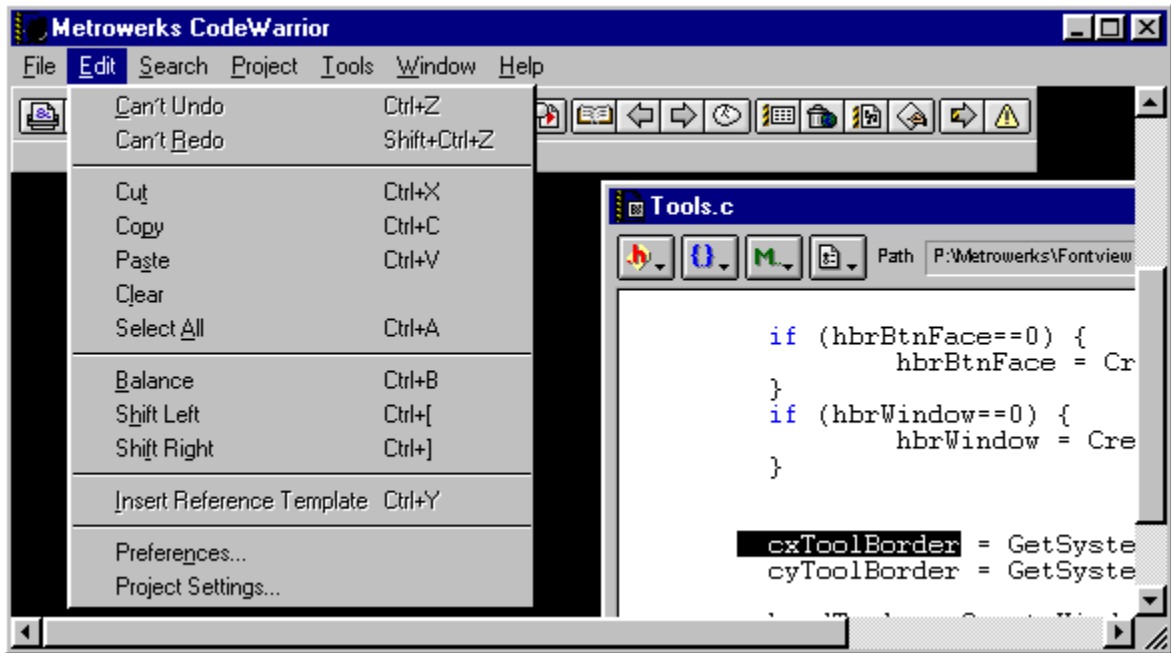
**Figure 2.2** The File menu



#### Edit Menu

The Edit Menu contains commands to help you customize the CodeWarrior IDE to your needs. Undo, cut, copy, paste, and selection operations are on this menu. Text formatting features are also present on this menu. “Configuring IDE Options Overview” on page 175 describes the Preferences and Project Settings possibilities.

**Figure 2.3 The Edit menu**



### Search Menu

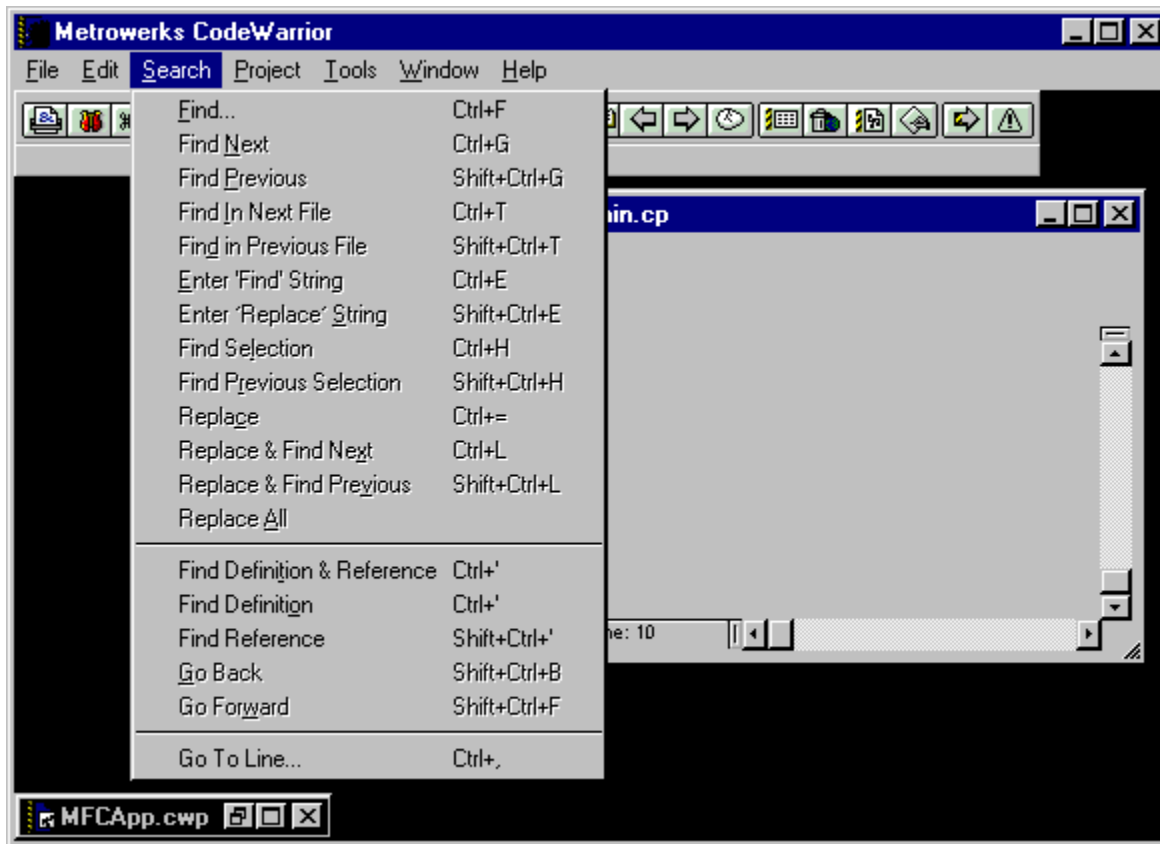
The Search Menu contains all the necessary commands used to find text, replace text, and to find the definitions of routines in your source code.

## Getting Started

### CodeWarrior IDE Guided Tour

---

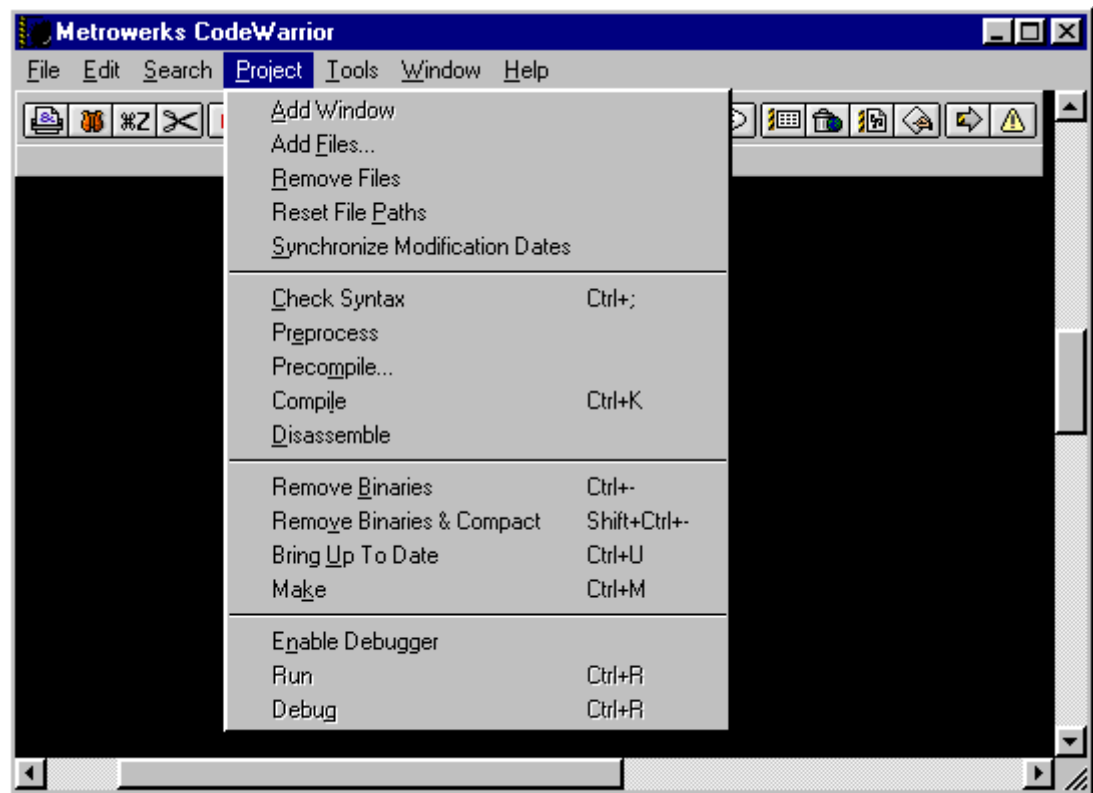
**Figure 2.4** The Search menu



### Project Menu

Use the commands on the Project Menu to add and remove source code files and libraries for your project, and to compile, build, link, and run your project. Commands for syntax checking, preprocessing, and disassembling source files are also on this menu. In addition, the Debugger may be enabled using this menu.

**Figure 2.5 The Project menu**



### **Tools Menu**

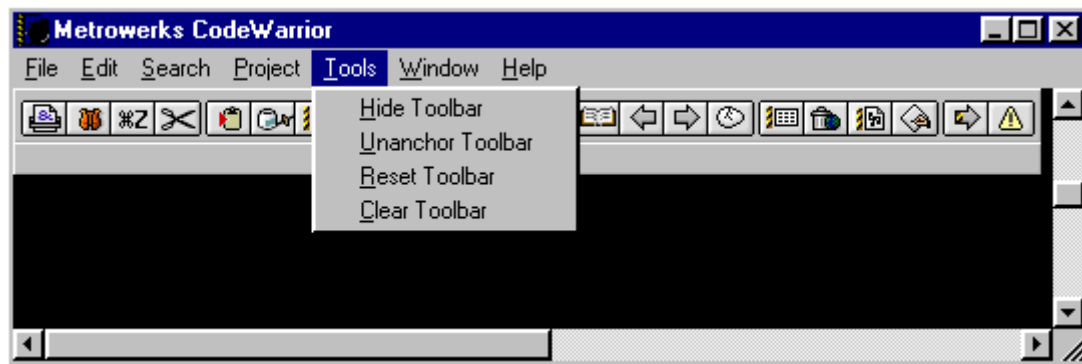
The Tools Menu contains all the commands used to customize the IDE Toolbar.

## Getting Started

### CodeWarrior IDE Guided Tour

---

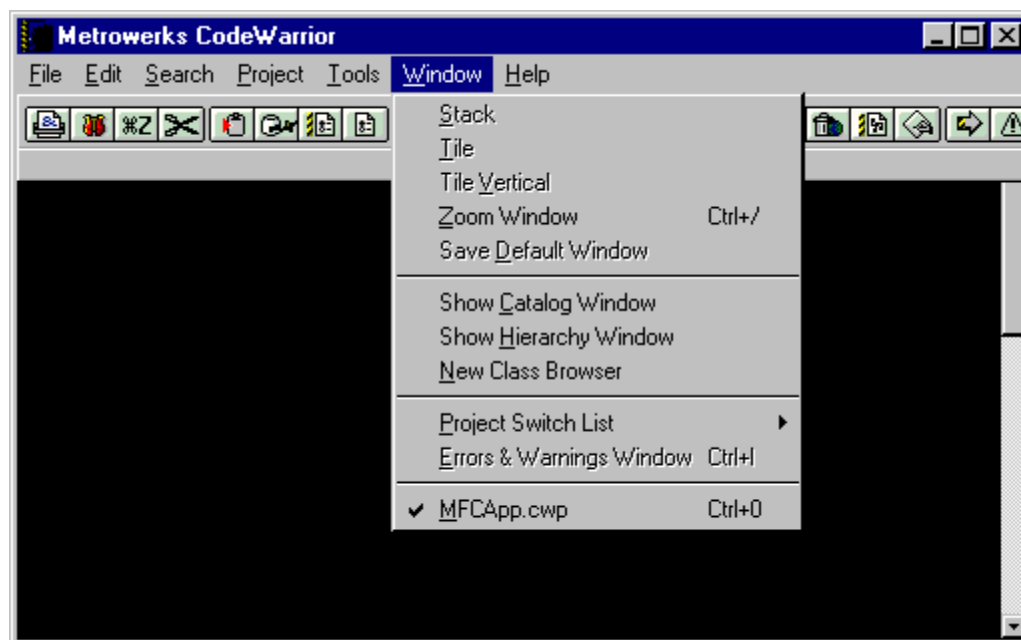
**Figure 2.6 The Tools menu**



### Window Menu

The Window Menu includes commands that arrange open editor windows, switch between windows, and switch to previously opened projects. Operations to bring up source code browser views are also accessible in this menu.

**Figure 2.7 The Window menu**



## Help Menu

The Help Menu includes commands that allow you to search for information on how to perform tasks and understand terminology in CodeWarrior. This menu also has a command for viewing the version of the CodeWarrior IDE you are working with.

**Figure 2.8** Help Menu



## IDE Toolbar

The IDE Toolbar is a palette of icons (Figure 2.9) that execute a corresponding menu command. While some commands are executing, such as compile, make, or check syntax, the IDE Toolbar displays status information in the area just below the row of icons.

The topics in this section are:

- IDE Toolbar Overview
- Accessing Additional Commands

### IDE Toolbar Overview

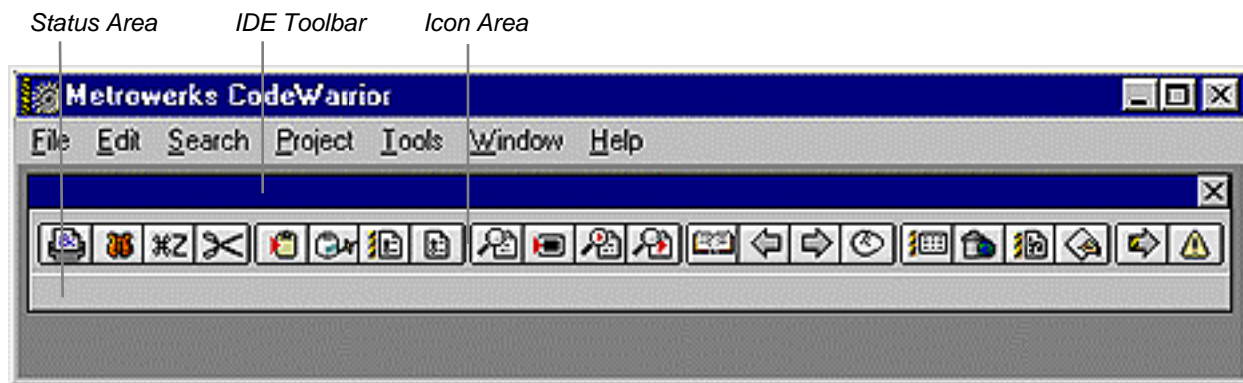
The IDE Toolbar has two sections called the Icon Area and the Status Area.

## Getting Started

### CodeWarrior IDE Guided Tour

---

**Figure 2.9 The default IDE Toolbar**



The icon area of the IDE Toolbar contains a row of icons that represent commands also listed in the CodeWarrior menus, such as Add File, Make, or Run. Some of these icons toggle to a different command when you press the Shift key and move the mouse pointer over the icon.

The message area of the IDE Toolbar sometimes contains a message that displays two possible types of information. Status information for commands such as searching, compiling, linking, or adding files are displayed at certain times. Otherwise, the name of the icon indicated by the location of the mouse pointer is displayed. When you move the pointer over a IDE Toolbar icon, you see the name of its corresponding menu command in the message area. If the command is disabled and not available with your current configuration, the information in the status area displays in italic characters.

### Accessing Additional Commands

For example, to execute the Find Previous command from the IDE Toolbar place the mouse pointer over the Find Next icon, then press the Shift key. The Find Next command displayed in the IDE Toolbar message area changes to Find Previous. Click the icon to perform the Find Previous command.



## **Other IDE Components**

This chapter discussed the CodeWarrior IDE menus and IDE Tool-bar. There are a few more components in the IDE that have Guided Tours.

For a tour of the Project Window, see “Guided Tour of the Project Window” on page 47.

For a tour of the Editor Window, see “Guided Tour of the Editor Window” on page 85.

For a tour of the Browser Windows, see “Guided Tour of the Browser” on page 150.

## **Getting Started**

*CodeWarrior IDE Guided Tour*

---



# Working with Projects

---

This chapter introduces the CodeWarrior project window, and shows how to set up, configure, and work with projects.

## Projects Overview

A project is a collection of files that the compiler and linker use to create executable computer code. Some examples of executable code include an application, library, or shared library.

Project files also contain options that affect the project you're working with. There are a wide variety of options that control many aspects of the IDE, such as code optimization, the browser, compiler warnings, and much more.

This chapter discusses many of the basic tasks involving projects, such as creating, opening, adding files to projects, and saving projects. It also describes operations such as moving files in the project window, marking files for debugging, and dividing the project window into segments or groups of files.

The topics in this chapter are:

- Creating a Project
- Opening an Existing Project
- Saving a Project
- Closing a Project
- Guided Tour of the Project Window
- Managing Files in a Project
- Moving a Project

- Controlling Debugging in a Project

## Creating a Project

This section discusses how to create a project. It includes an explanation of project stationery that you can use to speed the project creation process.

The topics in this section include:

- About Project Stationery
- Using Stationery Projects
- About the Project Stationery Folder
- Creating Your Own Project Stationery

### About Project Stationery

A project stationery file is an exact copy of a minimal “starter” project file. Think of it as a template, or blank slate, that is used to quickly create a new project. When you create a new project or open a project stationery file, CodeWarrior creates a new project and a new project folder, and copies all the stationery information to the new folder.

Stationery information includes the following:

- All option settings for the project
- All files included in the stationery project (libraries, source code files, and resource files)
- All segmentation and grouping information, including segment loader settings (Mac OS 68K project only) and names.

If you use a stationary file to create your project, all the necessary files can be put into a new folder with the same name as your project. After creating your new project from stationery, you can open it and begin writing code in CodeWarrior.

## Using Stationery Projects

There are a few short steps involved in using a stationery project file to create a new project:

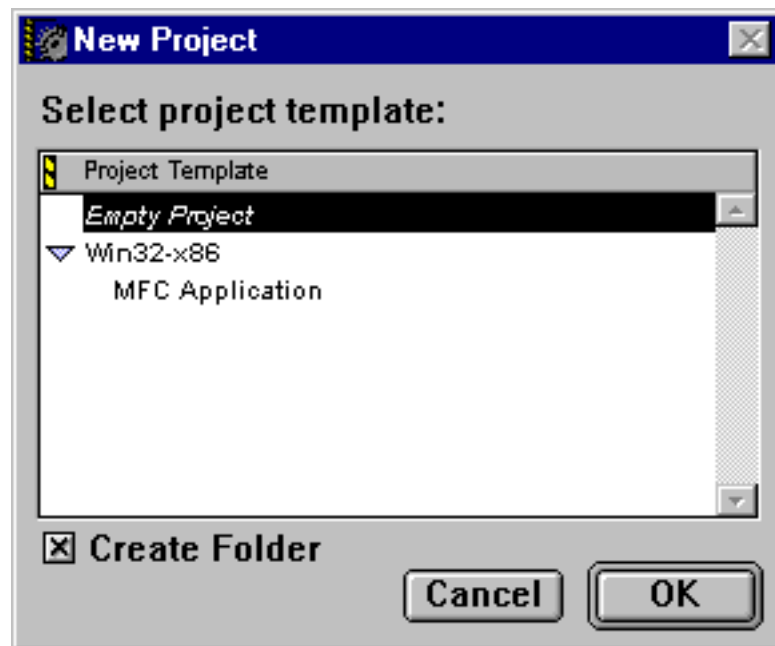
- Choosing the New Project Stationery File
- Naming Your New Project
- Modifying Your New Project
- Building Your New Project

### Choosing the New Project Stationery File

To create a new project, select the New Project command from the File Menu.

CodeWarrior displays the New Project window, shown in Figure 3.1, from which you choose your project stationery.

**Figure 3.1** New Project window



Click on a disclosure triangle in the window for a target you are interested in to see the project stationery available for that target. Fig-

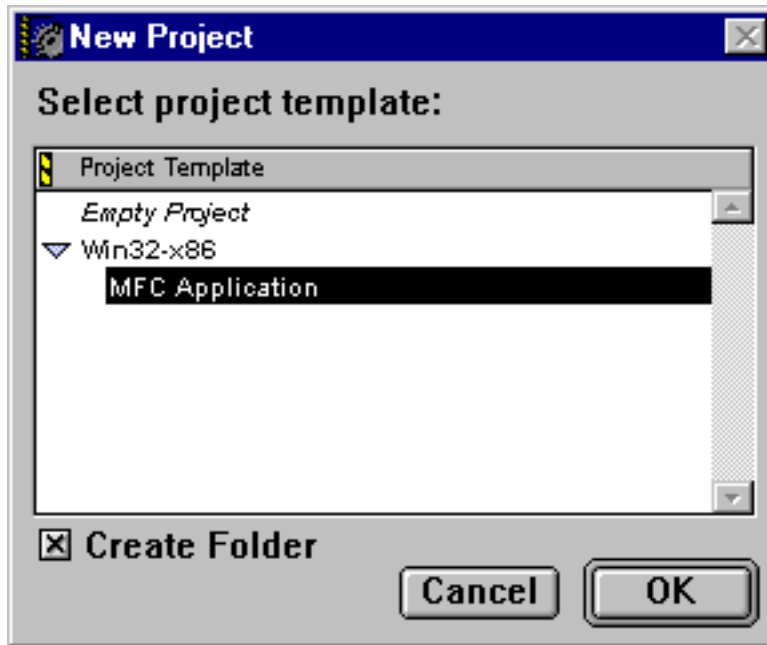
## Working with Projects

### *Creating a Project*

---

Figure 3.2 shows several levels expanded. It may be necessary for you to click on another disclosure icon to show the stationery of interest to you.

**Figure 3.2** Choosing project stationery



Click one of the project stationery items that corresponds to your intended target. If you want to create a new folder containing all the new project information, make sure the Create Folder checkbox is selected.

### **Naming Your New Project**

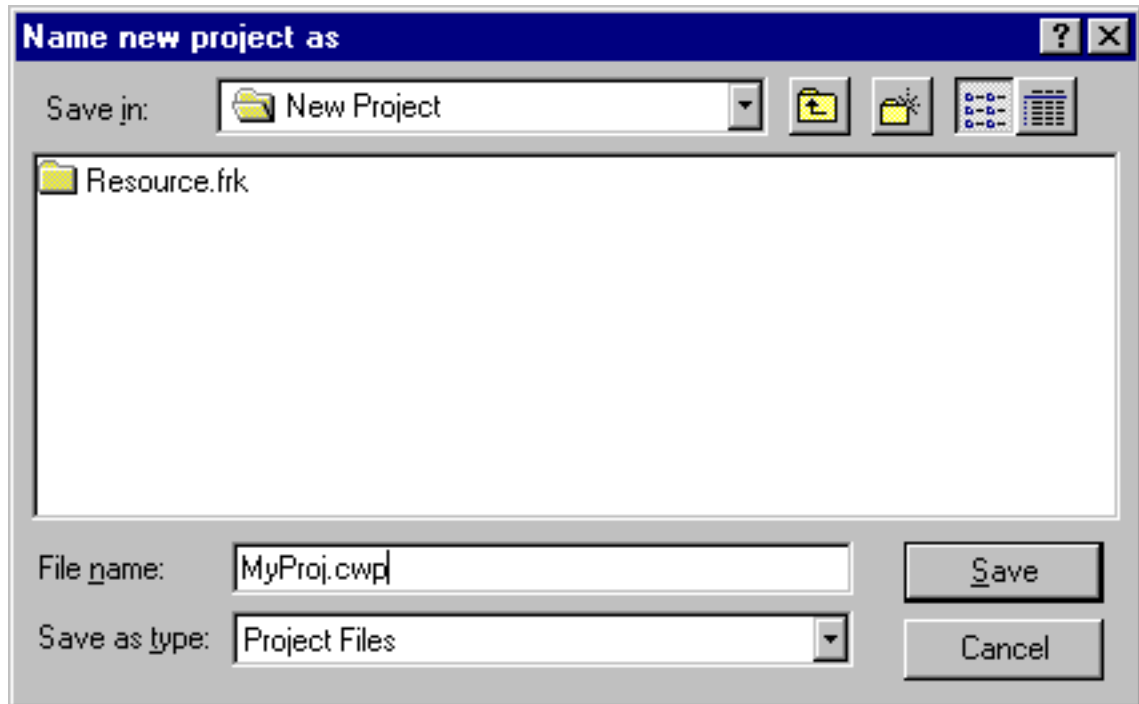
Click the OK button and the a dialog box appears to allow you to name your new project, as shown in Figure 3.3. Enter a name for your new project, and use the dialog controls to navigate to a location on your hard disk where you want to save the project. Then click Save to save the new project information.



**TIP:** We suggest naming your project with a `.cwp` extension. This makes your project easier to visually identify on your hard disk.

---

Figure 3.3 Naming a new project



**WARNING!** If you selected *Create Folder* when choosing stationery, and you already have a project with the same name in the same location on the hard disk, you will get an error message. Be sure to use a unique name for your new project.

---

When you choose **Save**, CodeWarrior automatically sets up your project, including:

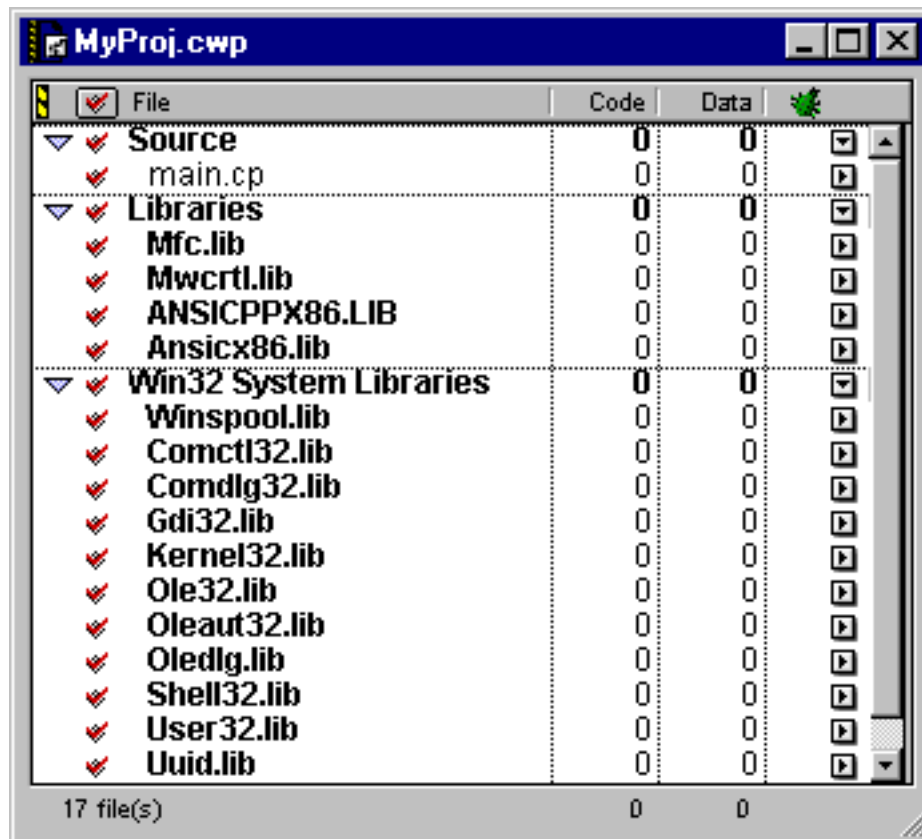
- Creating a project folder with the same name as your project (if you selected the **Create Folder** checkbox as discussed in “Choosing project stationery” on page 38), minus the file extension. The new folder contains your new project file and the files and libraries designated by your choice of stationery.

## Working with Projects

### Creating a Project

- Setting project settings and preferences to the settings in the chosen stationery.
- Opening the project window. The new project contains libraries, source code placeholders, and resource file placeholders (see Figure 3.4)

Figure 3.4 A project window



### Modifying Your New Project

Most new projects created from stationery contain source files that are basically empty placeholders. In Figure 3.4, the file named “main.cp” serves that function. You probably want to delete this file and replace with sources of your own. See the section “Managing Files in a Project” on page 51 for more information about manipulating these files.



## **Building Your New Project**

After you have your project created and files added to it, you will want to build it to produce your target application, library, or whatever you are making. To build a project see the discussion in “Compiling and Linking Projects Overview” on page 211.

## **About the Project Stationery Folder**

CodeWarrior provides project stationery for many different kinds of projects. Project stationery for common types of projects are inside folders nested in the (Stationery) folder, inside the CodeWarrior folder on your hard disk.

When you create a new project, you can see all the available stationery by clicking the disclosure triangles at the left side of the New Project window (Figure 3.2).

The following files can be included in the (Stationery) folder and are recognized by CodeWarrior as stationery projects.

- Normally saved projects
- Aliases to projects
- Project stationery files

See the ReadMe file in the (Project Stationery) folder for more information about the stationery.

## **Creating Your Own Project Stationery**

All project stationery files you see in Figure 3.2 are stored in the folder named (Stationery) in the CodeWarrior folder.

To create your own project stationery file, create or open a project. The stationery project file you create can include the files and options you want to have for your starter project.

Before opening a new project window, you can configure options so that you can save your favorite settings for creating projects.

## Working with Projects

### Creating a Project

---

Briefly, the procedure for doing this is to launch the CodeWarrior IDE, then choose the Preferences command under the Edit Menu. You see a dialog as shown in Figure 8.3 on page 179. To learn how to change these settings, refer to “Choosing Preferences” on page 180. You can also set project settings in the same way. See “Choosing Project Settings” on page 191.

After changing your settings, you can create a new project, using the New Project command under the Edit Menu. A dialog window as shown in Figure 3.5 appears asking you to name your new project, and select a location for the project file on your hard disk.

Next, add the appropriate files and set the Project Settings (on the Edit Menu) as desired for your stationery project. For information about adding or changing files in the project, see “Managing Files in a Project” on page 51. For information about changing any of the project options, see “Configuring IDE Options Overview” on page 175.

Now choose the Save A Copy As command from the File Menu. Select Project Files from the Save As Type pull-down menu in the dialog that appears, as shown in Figure 3.5. If you want more information on this process, refer to “Backing Up Files” on page 78 for more information about saving a copy of the project under a different name. Make sure to select a suitable location for your project stationery file. The file should be saved somewhere in the (Stationery) folder inside the CodeWarrior folder. Give the project a name and click Save.

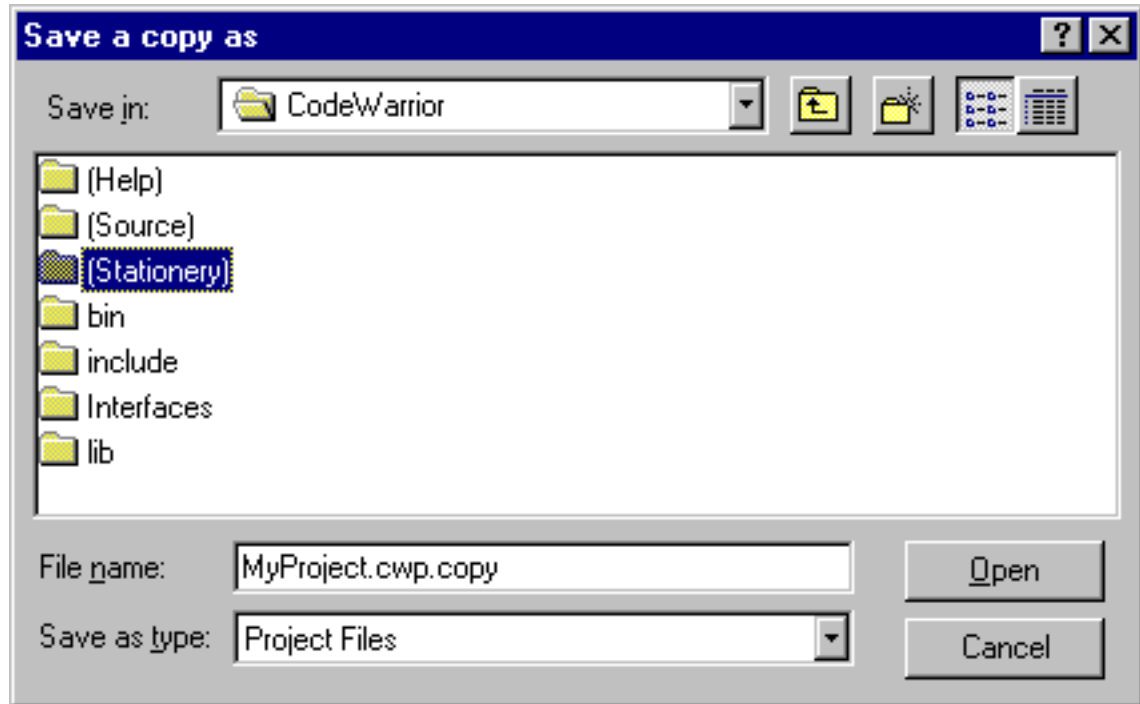


---

**TIP:** We suggest naming your project with a `.cwp`. This makes your project easier to identify on your hard disk and obeys the conventional naming that CodeWarrior expects from a project file.

---

**Figure 3.5** Saving a Project as Stationery



The reason you want to save the project before doing a lot of work with it is because you will have a “starter” or stationery project file on your hard disk. You can make copies of the “starter” project to get new projects quickly started.

New projects started with stationery will have all the settings you initially configured for your stationery project.

If at any time you decide that you want to use different project settings, you just create a new stationery project. Just make sure that you don't have a project window open, then configure your options, then open a new project, then save your new stationery project.

## Opening an Existing Project

There are two ways you can open a project file from within CodeWarrior. This section tells you how to open your projects so you can work on them.

## **Working with Projects**

### *Opening an Existing Project*

---

The topics in this section are:

- Using the Open Command
- Using the Project Switch List

### **Using the Open Command**

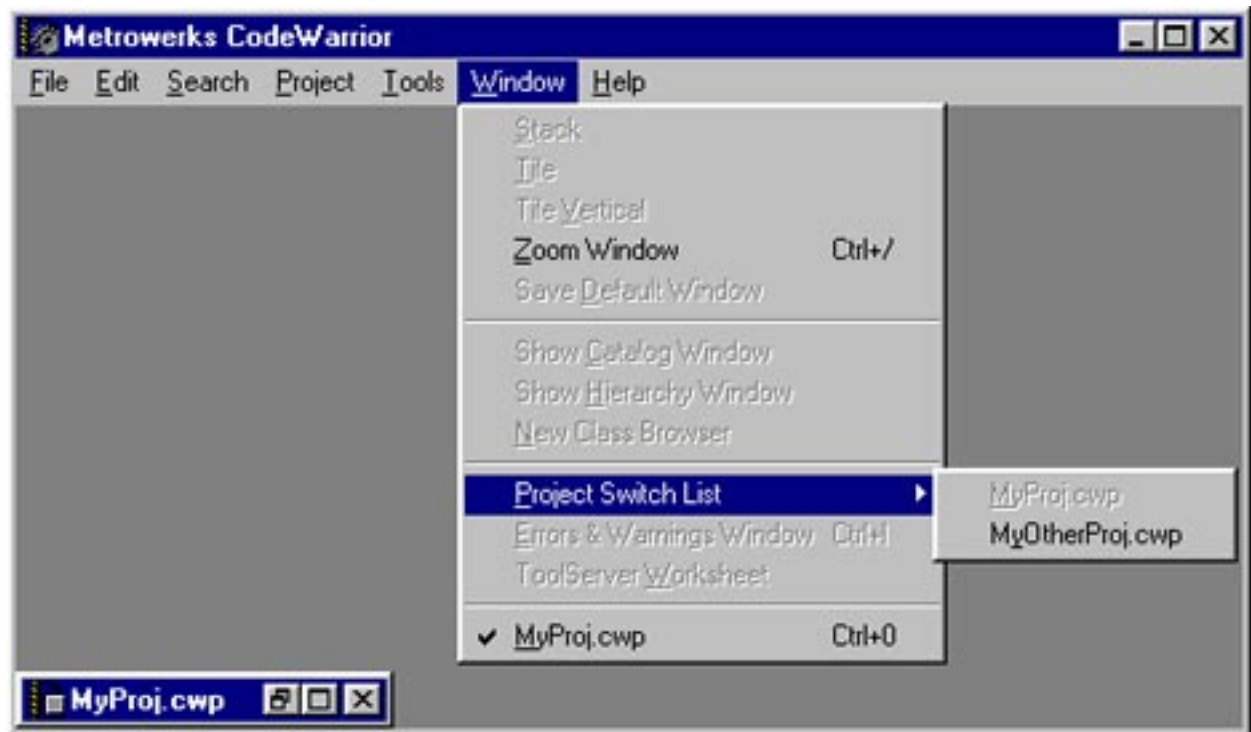
Information on how to open a project file is discussed in “Opening Files with the File Menu” on page 70.

### **Using the Project Switch List**

You can have only one project open at a time in CodeWarrior. Opening or creating an additional project closes the first one, saving all of the closed project’s information and settings.

The last few projects you have opened and closed are available from the Project Switch List submenu in the Window Menu (Figure 3.6). To switch back to a previously opened project, just choose a project from this list.

Figure 3.6 The project switch list



Choosing an item from this submenu closes the open project and opens the newly selected project.

To reopen a previously opened project, just choose a different project from the Project Switch List.

CodeWarrior adds projects to this list when you close them. When you quit the CodeWarrior session altogether, the Project Switch List is cleared.

## Saving a Project

CodeWarrior automatically updates and saves your project when you do certain actions. This section discusses these actions that cause the project file to get saved.

## Working with Projects

### *Closing a Project*

---

Your project settings get saved when you:

- Close the project
- Open another project
- Change preferences or settings for the project
- Compile any file in the project
- Quit CodeWarrior

You never have to manually save your project unless you want to create a copy of it. See “Backing Up Files” on page 78 for a description of how to create a backup copy of a project.

### Items Saved with Your Project

When CodeWarrior automatically saves your project, it saves the following information:

- The names of the files added to your project and their locations
- All configuration options
- The object code of any compiled source code files

To learn more about how to copy your project’s saved information when you copy your project files, refer to “Moving a Project” on page 64.

### Saving a Copy of Your Project

CodeWarrior lets you save a copy of the project in a format you specify. To read about using this feature, see the discussion in “Backing Up Files” on page 78.

## Closing a Project

After you have been working with your project for awhile, you may want to close it to work on another project, or to quit the CodeWarrior application to work on something else. To read about how to close your project, see “Closing One File” on page 80.

## Guided Tour of the Project Window

The project window displays the project you are working on, shows the project's files, segmentation, and whether or not debugging information is to be generated for each file. It also shows which files need to be compiled the next time the project is built, and object code and data sizes. See Figure 3.7 on page 48 for a good look at the project window.

For more on debugging information, see “Controlling Debugging in a Project” on page 64.

### Navigating the Project Window

To navigate the project window, use the scroll bars on the right side of the window, or the Up and Down Arrow keys on your keyboard. If your Project window contains many files, use the Home and End keys to jump from the first file in the first segment (Home Key) to the last file in the last segment (End Key).

Use the Page Up and Page Down keys to scroll your project window one page up or one page down.

### Project Window User Interface Items

This section will take you on a tour of the various user interface items that you can use to alter your project view or project settings. The project window contains many pop-up menus and columns that indicate segmentation/grouping, debugging, access path, and header information. This section briefly describes these pop-up menus and columns.

The topics in this section are:

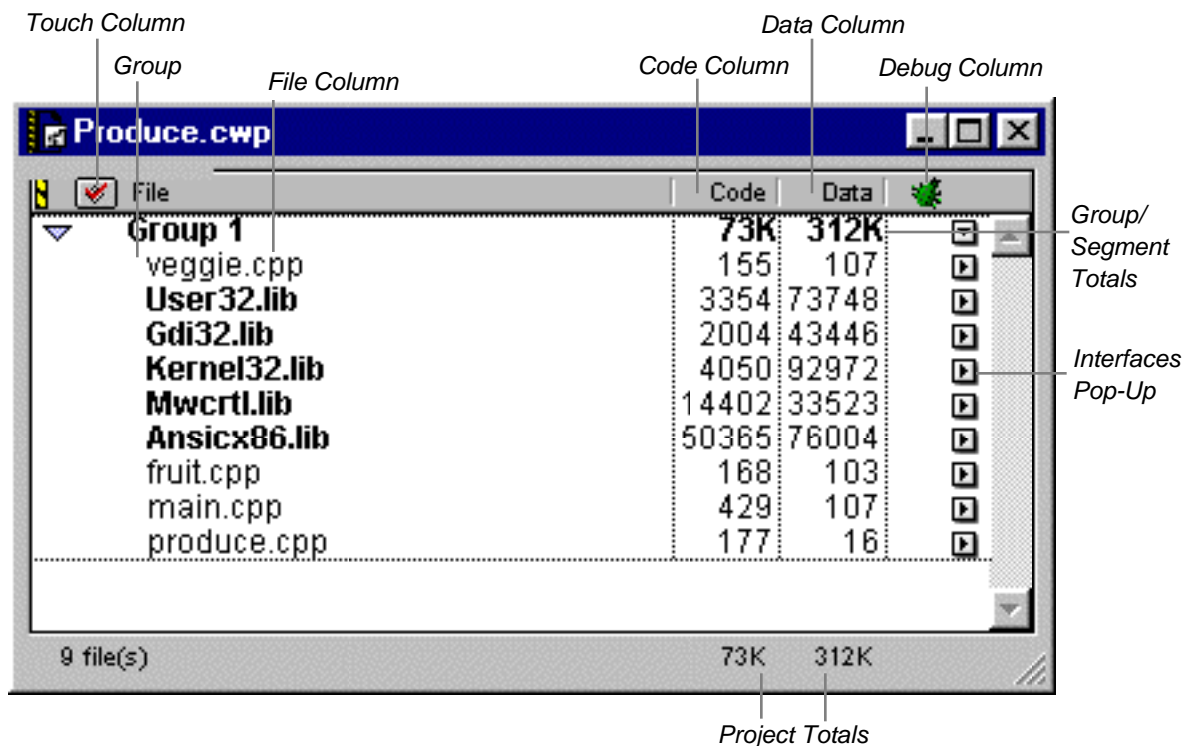
- Group Organization
- File Column
- Code Column
- Data Column
- Debug Column

## Working with Projects

### Guided Tour of the Project Window

- Touch Column
- Interfaces File Pop-up
- Group File Pop-up

**Figure 3.7** The Project window



### Group Organization

Each CodeWarrior project is visually divided into different groups of files. Files are moved between groups, groups can be collapsed and other groups expanded using the disclosure triangle, and entire groups can be moved around in the window. Figure 3.7 shows an example of file groups in the project window.

While this file grouping capability is useful from an organizational standpoint, it also has significance for some particular targets you may be building. For example, some targets need the group facility to produce segments at link time required by the target architecture.



For more discussion about groups see “Managing Files in a Project” on page 51.

### **File Column**

The File Column of the project window, shown in Figure 3.7, shows which files are in your project and the names of the groups or segments which contain those files. Files are added, moved around, opened, or removed.

Double-clicking on a file name in the File Column opens the file in an editor window. The exception to this is binary files (such as library files) which cannot be opened into an editor window. If the file was created with an application besides CodeWarrior the file will be opened in that application. For information on opening files from the File Column, see “Opening Files from the Project Window” on page 71.

For information on adding, moving, or removing files, see “Managing Files in a Project” on page 51.

### **Code Column**

The Code Column of the project window, shown in Figure 3.7 on page 48, shows the size, in bytes or kilobytes, of the compiled executable object code for a corresponding file. If zero is displayed, it means that your file has not yet been compiled.

The Code values do not necessarily reflect the amount of object code that will be added to the final binary file. The linker may not use all a project file’s object code. Instead, the linker only uses the parts that are referenced by other files in the project.

For more information on how the linker works, see “Compiling and Linking a Project” on page 213.

### **Data Column**

The Data Column of the project window, shown in Figure 3.7 on page 48, shows the size, in bytes or kilobytes, of the non-executable data area for a corresponding file. If zero is displayed, it means that

## Working with Projects

### *Guided Tour of the Project Window*

---

your file has not yet been compiled, or that the file does not contain a data section in its source code.

The Data values do not necessarily reflect the amount of data that will be added to the final binary file. The linker does not use all a project file's data. Instead, the linker only uses the parts necessary to completely link the project.

For more information on how the linker works, see “Compiling and Linking a Project” on page 213.

### **Debug Column**

The Debug Column of the project window, shown in Figure 3.7 on page 48, indicates whether debugging information is being generated for a file. It's easy to recognize the Debug Column because there is an icon of a small bug at the top of the column.

For more information about debugging information for a file, see “Activating Debugging for a File” on page 65.

### **Touch Column**

The Touch Column of the project window, shown in Figure 3.7 on page 48, indicates whether a file needs to be compiled the next time a project is built. The Touch Icon at the top of the column causes all files in the project to be touched.

For more information about touching and untouching files, see “Touching and Untouching Files” on page 213.

### **Interfaces File Pop-up**

The Interfaces File Pop-up in the project window, shown in Figure 3.7 on page 48, allows you to see and open interfaces and header files for your project source files.

The Interfaces File Pop-up also allows you to toggle the touched or untouched status for a given file.

For more information about opening interfaces and header files, see “Interfaces Files Pop-up Menu” on page 73.

For more information about touching and untouching files, see “Touching and Untouching Files” on page 213.

### **Group File Pop-up**

The Group File Pop-up of the project window, shown in Figure 3.7 on page 48, allows you to easily see which files are in a group without fully expanding the group. You can also open a file from this pop-up. This pop-up distinguishes itself from an Interfaces File Pop-up since it is in the same row as the name of a Group.

For more information about opening files this way, see “Opening Files from the Project Window” on page 71.

## **Managing Files in a Project**

This section discusses aspects of adding, moving, naming, organizing, viewing, marking for compilation, and removing files from your project. The topics are:

- About Groups and Segments
- Selecting Files and Groups
- Expanding and Collapsing Groups
- Adding Files
- Moving Files and Groups
- Creating Groups
- Removing Files and Groups
- Renaming Groups
- Touching and Untouching Files

### **About Groups and Segments**

Groups allow you to organize your source code files into logical categories to help you keep track of where you put your files. On some targets, groups serve the additional purpose of keeping code in segments that the linker will create.

## Working with Projects

### *Managing Files in a Project*

---

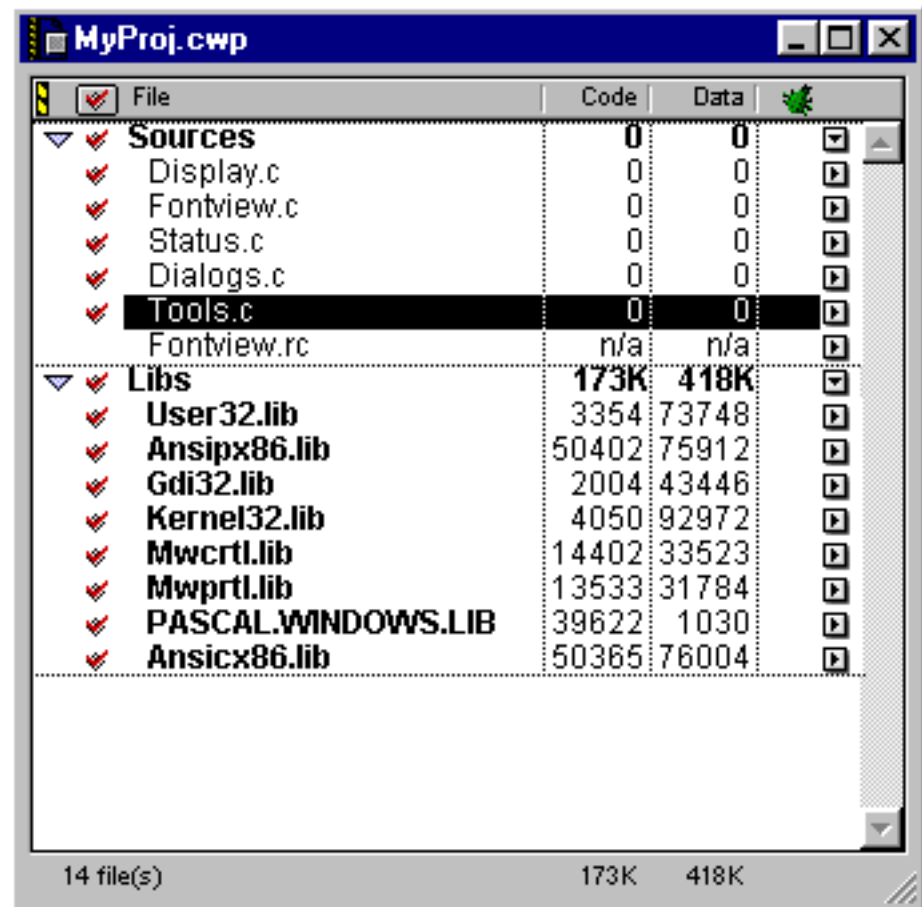
Groups of files in a CodeWarrior project are a convenient way to organize files sharing a common purpose. For example, all files that involve playing sound could be put in one group named “Sound” in your project.

## Selecting Files and Groups

From the project window you can select one or several files and groups to open, compile, check syntax, remove from the project, or move to a different group.

When a group is selected, all of the files within the group are selected, regardless of whether or not one of its source code files are included in the selection. For example, the project in Figure 3.8 has one of its source code files selected. If you executed the Check Syntax command, the operation is performed on all of the selected files.

Figure 3.8 Selecting a file in a project window



### Selection by mouse-clicking

To select a single file or group in the project window, click its name.

To deselect a single file or group in the project window, Shift-click its name.

To select a consecutive list of files or groups, select the first file or group in the list by clicking its filename, then Shift-click the last file or group to be selected. If the first entry in the selection is a file, only files will be selected. If the first entry in the selection is a group, only groups are selected.

## Working with Projects

### Managing Files in a Project

---

To select a non-consecutive group of files or groups, Shift-click each filename or group.

#### Selection by keyboard

Type the first few characters of the file name you want to select. As you type, the characters appear in the IDE Toolbar message area, and CodeWarrior selects the file as soon as the characters identify the file closest to your entry.

Use the Backspace key if you make a typo.

Press Enter to open the file.



---

**NOTE:** *Only visible files can be selected this way. Files in collapsed groups will not be matched with your keystrokes.*

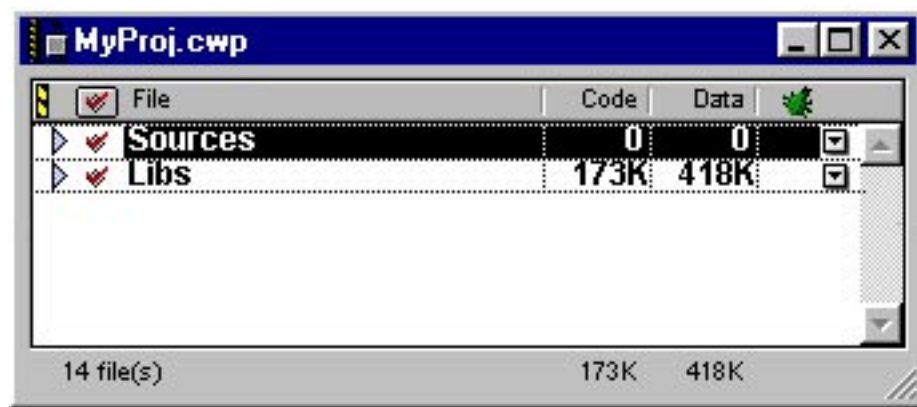
---

## Expanding and Collapsing Groups

Groups display files in collapsible lists, in much the same way the Windows tree control displays files and folders hierarchically.

To expand a group and view its files, click on the triangle to the left of the desired group. To close a group and view only its name, click the triangle again (Figure 3.9).

**Figure 3.9** Expanding and collapsing groups



To expand all groups in a project, Alt-Click the triangle of a collapsed group.

To collapse all groups in a project, Alt-Click the triangle of an expanded group.

## Adding Files

This section tells how to add files to your CodeWarrior project.

When adding a file to a project, Access Paths to the file or files automatically get set in the project. The Message Window informs you whenever a new access path is added.

For more information about Access Paths see "Access Paths Panel" on page 195.

For more information about the Message Window, see "Guided Tour of the Message Window" on page 229.

Here are the topics you will learn about in this section:

- Where Files Appear—where files go when they are added to your project
- Using the Add Files Command—add one or more files
- Using the Add Window Command—add one file

## Working with Projects

### Managing Files in a Project

---



---

**NOTE:** *If you are using a Metrowerks Pascal compiler, the source code file containing the main Pascal “Program” must be the first file in the project window.*

---

### Where Files Appear

Files are always added after the currently selected item in the project window, or at the bottom of the project window if there is nothing selected. To put a new file or files in a particular location, always select the file or group above that location before performing the Add Files or Add Window command.

If a group is selected, regardless of whether or not the group is expanded or collapsed, the files to be added are placed at the end of the selected group. To learn about how to select a file or group of files, see “Selecting Files and Groups” on page 52.



---

**NOTE:** *If a group or a file within a group is not selected prior to adding files, a new group is created and appended to your project. The files added are placed in this new group.*

---

Of course, you can always move a file or group of files to a new location after adding to the project. To see how to move files and groups around in the project window, see “Moving Files and Groups” on page 58.

### Using the Add Files Command

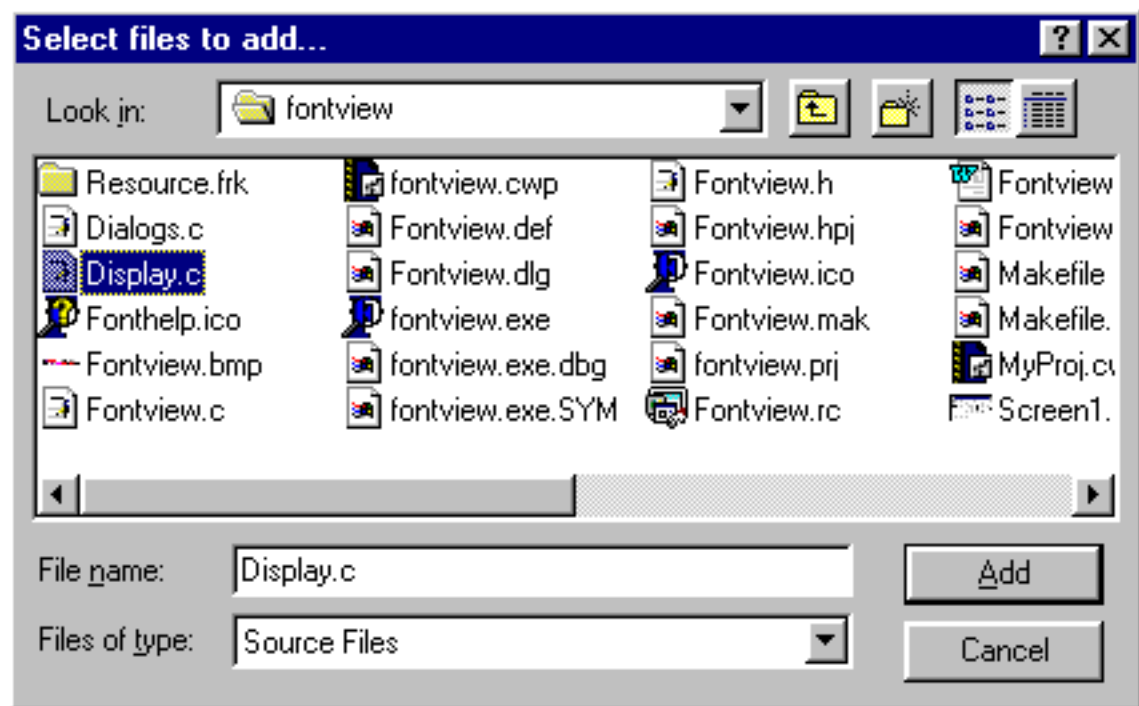
The Add Files command on the Project Menu opens a dialog box you can use to add files to your project from many locations. Use this command to add source code files, libraries, resource files, or a shared library.

This command opens a dialog box that can be used to add files to your project from a directory (Figure 3.10). In order for files to appear in this Add Files dialog box, the files must have a recognized extension (such as .c or .p) in the filename. To examine and config-



ure possible extensions for file names, refer to “Setting a File Extension” on page 212.

**Figure 3.10** Adding multiple files to a project



Select the files you want to add. The names of the selected files appear in the File name field. To add the files, click the Add button.



**NOTE:** You can add large numbers of files this way, but there will be a delay while CodeWarrior locates the files and adds them to the project.

### Using the Add Window Command

The Add Window command adds the file associated with the active window to the project. You typically do this when you’ve opened a new file for editing, then decided that you would like to add it to the currently-open project window.

## Working with Projects

### Managing Files in a Project

---

To use the Add Window command, select a location in the project window. Then, open a file, make sure its window is active, and select the Add Window command from the Project Menu. The Save As dialog comes up prompting you to select a location and name for the file. After you save the file, the file is added to the open project.



---

**NOTE:** *The Add Window command is enabled when the active window is a text file, the file is not yet in the project, and it has a permissible filename extension (see “Target Panel” on page 192). The Add Window command is dimmed otherwise.*

---

## Moving Files and Groups

To move one or more files or groups, select the files or groups to be moved. Selecting a group selects all of its files regardless of whether or not its files are visually selected in the project window. If you need help selecting files and groups, see “Selecting Files and Groups” on page 52.

Next, drag the selected files or groups to their new location in the project.

A focus bar (an underline) indicates where the selected files will be moved to when the mouse is released.

Whether you are moving files or groups depends on your selection. For example, if your selection consists of files then the focus bar is shown on each line, under both groups and files.

If your selection includes at least one group, then the underline is shown only under other groups as you move the mouse in the project window. This enables you to rearrange groups.

Finally, release the mouse button when the underline is positioned after the desired file or group position.

When a file is underlined and the mouse is released, the selected files are placed in the same group, following this file. When a group is underlined, the selected files are placed at the end of this group.

## Creating Groups

To create a new group, select the file(s) you want to put in the new group, then drag to right just below the division line of the last segment in the project. A new group is automatically created.

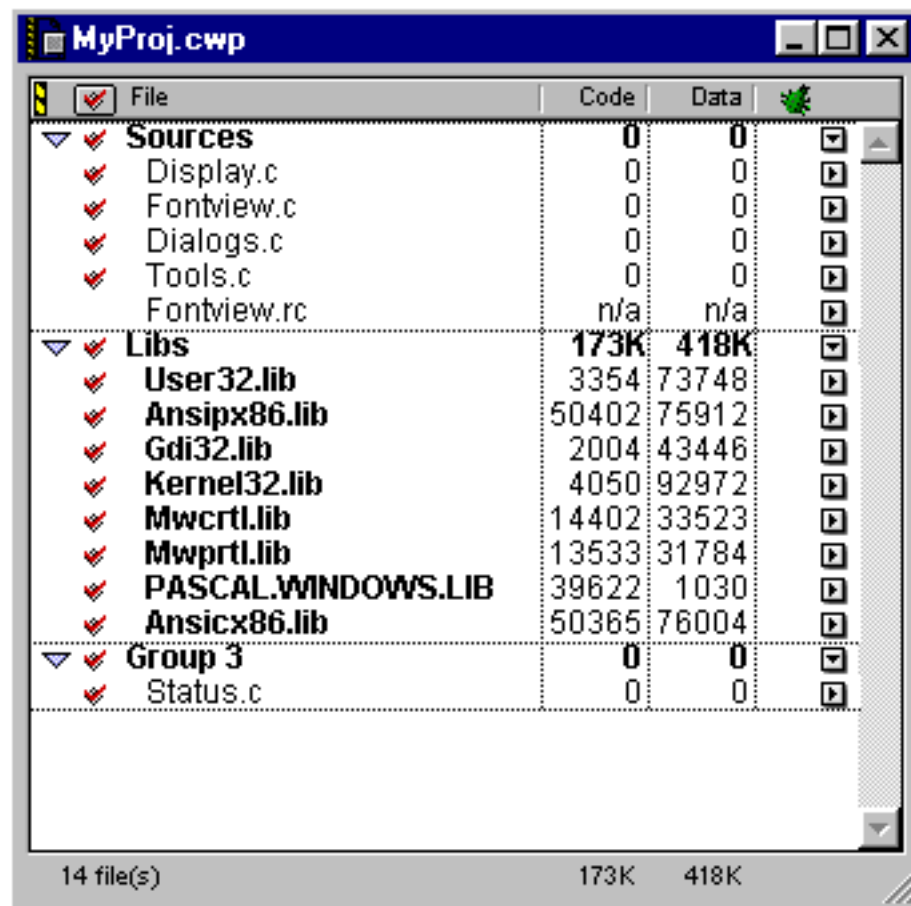
Another way to do this is select the file that will be the first file in the new segment then press Control-Return. CodeWarrior creates a new group with the selected file and those below it, stopping at the next group.

After creation, the project's groups are renumbered to include the new group as shown in Figure 3.11. The new group will have a name like "Group 2" or some other number. For information on how to change this name, see "Renaming Groups" on page 61.

## Working with Projects

### Managing Files in a Project

Figure 3.11 The project window after creating a new group



## Removing Files and Groups

To remove one or more files or groups, first select the files or groups to be removed. Note that selecting a group selects all of its files regardless of whether or not its files are visually selected in the project window.

To learn how to select files, refer to “Selecting Files and Groups” on page 52.



**WARNING!** *This command can't be undone. If you mistakenly remove a group, you must re-add its files using either the Add Window or Add Files commands under the Project Menu.*

---

Then select the Remove Files command from the Project Menu or press Alt-Delete. The selected files and groups will be removed from your project.

When you remove a group from your project, all of the files that are in that group will still be in the project. If there is more than one group in the project, the files will be added to the group that is above the group you are deleting. If there is only one group in your project, you won't be able to delete the group, since CodeWarrior requires at least one group in a project.

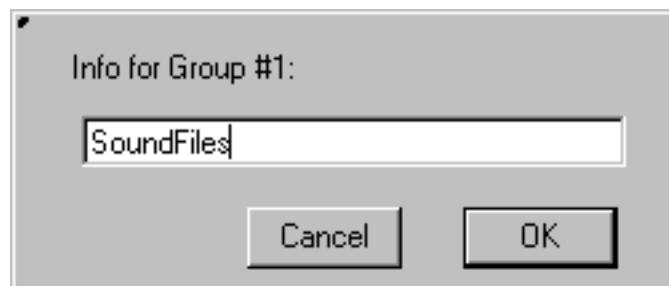
To remove a group from your project, select the group to remove and press Shift-Control-Return. CodeWarrior moves the files in this group up to the previous group and the group numbers are updated.

## Renaming Groups

To rename a group, select the group to be renamed by clicking on it then press the Enter key. You may also use the arrow keys to navigate to the group, then hit the Enter key.

When you have done this, a dialog box appears containing a standard name dialog (Figure 3.12).

**Figure 3.12** Changing a group name



## Working with Projects

### Managing Files in a Project

---

Type the new name in the name box and click OK. The name of the group is changed in the project window.

If you have selected more than one group, the same dialog opens for the next group selected, enabling you to change its name as well.

Note that when you attempt to change the group's name again, "Group" (or "Segment") and its number are displayed in the dialog's title, not the name that you have given it. This shows the group's order in your project.

## Touching and Untouching Files

Use the Touch Column shown in Figure 3.13 to flag files that need compilation. CodeWarrior doesn't always recognize file changes and may not automatically recompile all files in certain cases, which is why the Touch Column feature is useful.

There are two possible ways to make sure changed files get compiled. One way is to click in the Touch Column beside the filename in the project window. A check icon should appear in the Touch Column next to the file name.

The other way is to select the Touch command from the Interfaces File Pop-up menu. The Touch command may appear at the top of the Interfaces File Pop-up and the Group File Pop-up.



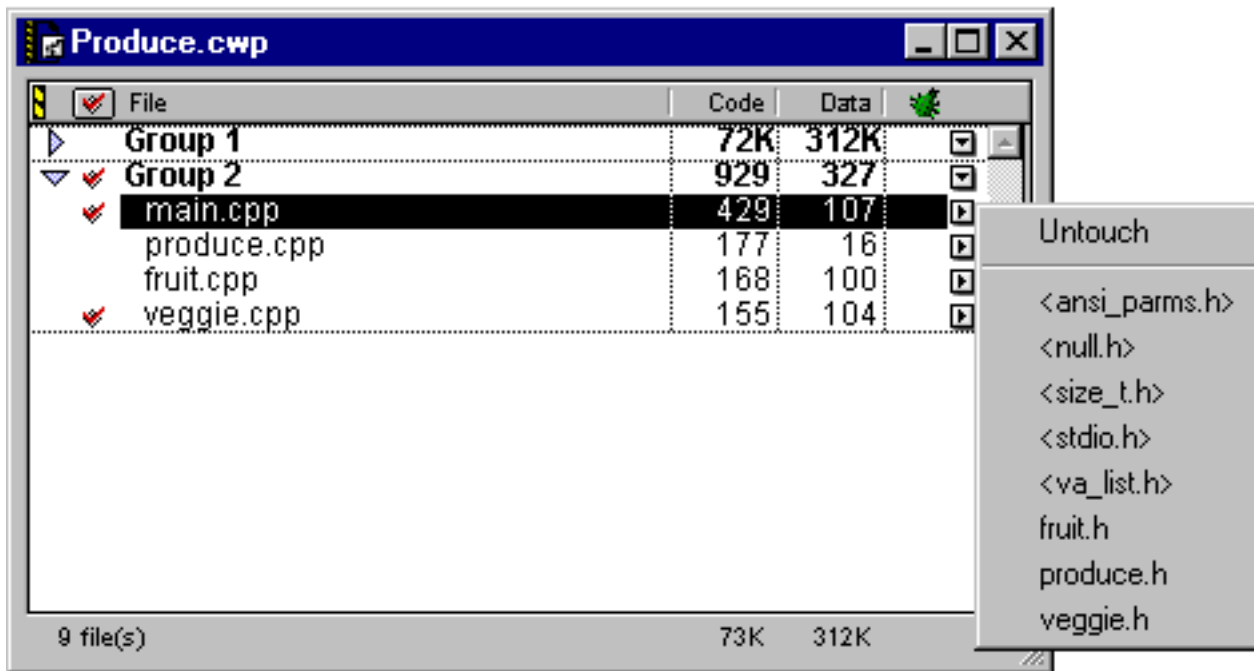
---

**TIP:** *If the file hasn't been changed since it was last compiled, the first command in either pop-up is Touch. When you choose Touch, CodeWarrior marks your file as dirty and compiles the file the next time it makes your project. If the file has been changed since it was last compiled, the Untouch command is shown.*

---

Refer to Figure 3.13 to see this illustrated in a Project window. You'll see a checkmark in the Touch Column next to the filename, and the Touch command in the Interfaces File Pop-up for the file you touched becomes Untouch.

Figure 3.13 Marking files for compilation



To unflag files you flagged for compilation, click again in the Touch Column, or choose Untouch from the Interfaces File Pop-up.

Note that the “check” icon at the top of the Touch Column may be used to touch all the files in the entire project.

To learn more about Touch and Untouch, see “Touching and Untouching Files” on page 213.

### Synchronizing modification dates

To update the modification dates stored in your project file, click the checkmark icon above the Touch Column, next to the barberpole icon.

Alternatively, choose the Synchronize Modification Dates command in the Project Menu.

CodeWarrior updates the modification dates stored in the project file. It checks the modification date for each file in the project, and if

## Working with Projects

### *Moving a Project*

---

the file has been modified since it was last compiled, CodeWarrior marks it for recompilation.

This command is useful if you edit your files with a third-party editor that doesn't notify CodeWarrior when it modifies a file.

## Moving a Project

Because of the way that CodeWarrior stores the options settings information for your project files, some special attention is required when moving your project files around on your hard disk.

### **Resource.frk and Your Project**

You change the options for your project by choosing the Project Settings command on the Edit Menu and making changes. When you do this, information is written into the Resource.frk folder that resides in the same folder as your project file.

If you decide to move your project file to another place on your hard disk, you will need to copy the Resource.frk folder also in order to preserve the settings of your project.

To learn more about options settings, refer to “Configuring IDE Options Overview” on page 175.

## Controlling Debugging in a Project

Your program will probably not run correctly the first time you build it. In order to debug it, you need to enable debug information for your project and the files you are concerned with. This section tells you how to do this.

The topics in this section are:

- Activating Debugging for a Project
- Activating Debugging for a File



## Activating Debugging for a Project

To enable debugging for a project, you need to set certain options in the Project Settings. Refer to “x86 Linker Panel” on page 207 and “Choosing Project Settings” on page 191 for a discussion of how to do this. If you select the Enable Debugger command from the Project Menu all the necessary configuration will be done for you automatically.

## Activating Debugging for a File

To generate debugging information for a source code file, click in the Debug Column and a Debug Info Marker appears in the column (a small star). When you add files to your project, CodeWarrior automatically sets this marker unless you have deselected the appropriate option in the x86 Linker Panel.

The Debug Info Marker indicates that debugging information will be generated for this file when the project is built. Clicking the marker removes it and causes the source code file to not have debugging information.

Libraries cannot have debugging information generated since they are built outside of your project.

When a Debug Info Marker is selected for the first time, the file is also marked for compilation the next time you build your project.

To generate Debug Info for all files in the project, hold down the Alt key while clicking any Debug Column marker. Generating debugging information for all files can be deselected by Alt-clicking again.



---

**NOTE:** *Marking a source code file for debug inclusion does not mean that a debug file is created during linking. The x86 Linker Panel contains the option that enables CodeWarrior to create a debugging file.*

---

## Working with Projects

### *Adding Preprocessor Symbols to a Project*

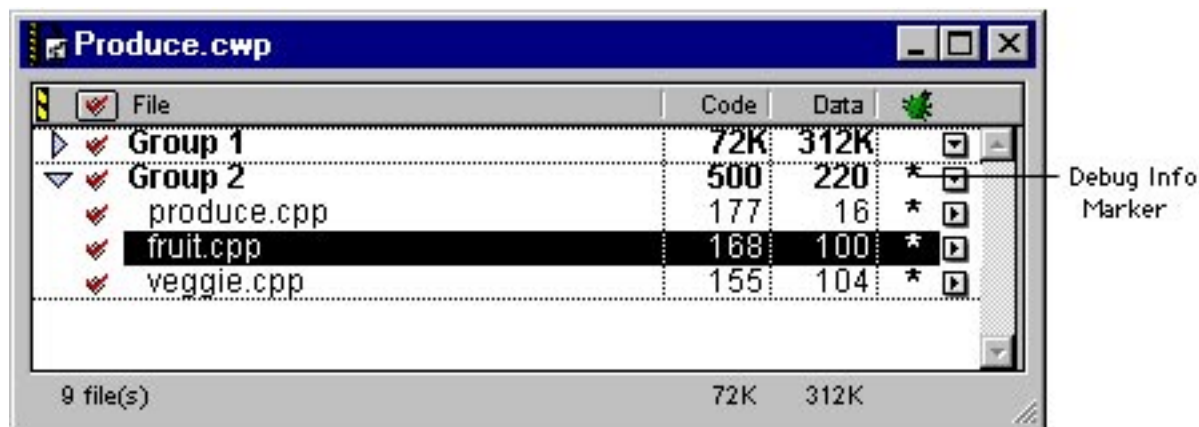
---

#### Debug Info Marker for Groups

The Debug Info marker also appears on group entries (Figure 3.14), and can be toggled on and off by clicking. The Debug Column, for groups, may be in either one of these states:

- **Black star:** all files in the group generate debugging information
- **No marker:** means no debugging info for all files in group.

**Figure 3.14** The Debug Column



## Adding Preprocessor Symbols to a Project

Sometimes you may want to add your own symbol definitions to your project so that they are automatically included at the beginning of each source code file when you build your project.

An example of this using the C or C++ language would be:

```
#define GLOBAL_DEBUG
```

Maybe you want to define this symbol when building development versions of your code, but want to undefine it before shipping your final product.

To do this, you would create a precompiled header and insert this symbol definition into the header. See "Using Precompiled or Pre-

processed Headers” on page 220 for more information on how to do this.

Also see the *CodeWarrior C,C++, and Assembly Language Reference* manual, and the *CodeWarrior Pascal Language Reference* for more information about this topic.

## **Working with Projects**

### *Adding Preprocessor Symbols to a Project*

---



# Working with Files

---

This chapter introduces the concepts behind working with files in CodeWarrior.

## Working with Files Overview

In this chapter we discuss opening, creating, saving, closing, and printing files in the CodeWarrior environment.

The topics in this chapter are:

- Creating a New File
- Opening an Existing File
- Saving a File
- Closing a File
- Printing a File
- Reverting to a Previously-Saved File

## Creating a New File

To create a new untitled window where source code may be entered, choose the New command from the File Menu.

After the new window appears, an insertion point is placed on the first line of the window. When you begin typing, CodeWarrior places text at this insertion point.

To learn more about text editing in the window you have just created, see “Source Code Editor Overview” on page 85.

## Opening an Existing File

There are several ways to open a file with the CodeWarrior IDE. The methods discussed here are:

- Opening Files with the File Menu
- Opening Files from the Project Window
- Opening Files from an Editor Window



---

**NOTE:** *You cannot open libraries or shared libraries with the CodeWarrior Editor, because of their binary format.*

---

### Opening Files with the File Menu

You can open two types of files with the CodeWarrior Editor:

- Project File—a file containing information on building a CodeWarrior project
- Text File—a source code, interface, or other text file

#### Project File

To open a Project File, choose the Open command from the File Menu. The CodeWarrior Editor displays an Open dialog, as shown in Figure 4.1. Click on the triangle on the right end of the Files of Type drop-down menu to cause the menu to appear, and select Project Files from it. The list of files changes to show project files that are eligible for you to open. You can navigate to a different directory to look for files to open by using the Look In drop-down menu at the top of the dialog.

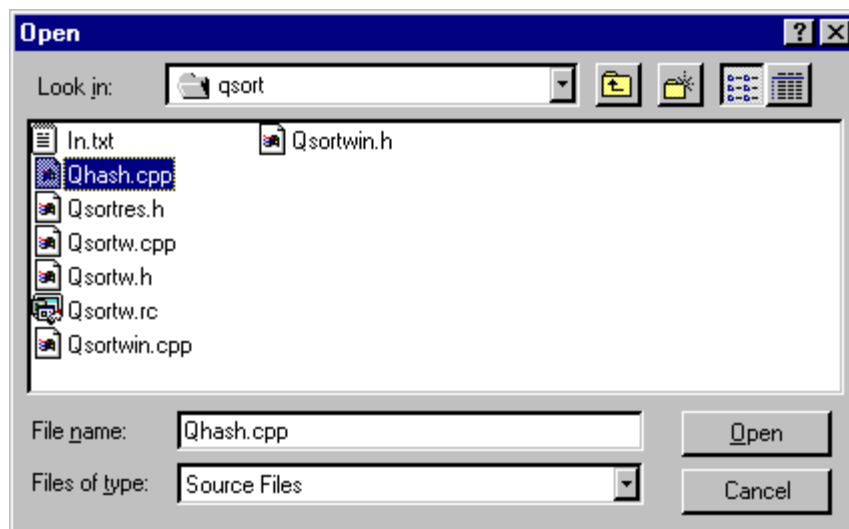
Click on the project file you would like to open, then click the Open button. CodeWarrior then opens the project, closing any other project you were working with.

For more information about working with CodeWarrior project files, see “Projects Overview” on page 35.

## Text File

To open a Text File, choose the Open command from the File Menu. The CodeWarrior Editor displays an Open dialog, as shown in Figure 4.1. Click on the triangle on the right side of the Files of Type drop-down menu to cause the menu to appear, and select Source Files from it. The list of files changes to show files that are eligible for you to open. You can navigate to a different directory to look for files to open by using the Look In drop-down menu.

**Figure 4.1** Open dialog



Then click on the file you would like to open to select it, then click the Open button. CodeWarrior then opens the file in an Editor window.

For more information about editing source code, see “Source Code Editor Overview” on page 85.

## Opening Files from the Project Window

There are three different ways to open files from within the Project window, depending on the type of the file you wish to see. The three different mechanisms include:

## Working with Files

### *Opening an Existing File*

---

- File Column—opening a file that is in the project
- Group File Pop-up Menu—opening a text source file from within a collapsed group
- Interfaces Files Pop-up Menu—opening an interface file included by a project's source file

#### **File Column**

If the file you wish to see appears in the File Column of the Project Window, double-click on the file name to open it.

CodeWarrior opens the file in an Editor window.

Another way to open a file is to select it, and then press the Enter key. You can select multiple files in the Project window, and open them all by pressing the Enter key. If you don't know how to select multiple files in a project, you can learn how by reading "Selecting Files and Groups" on page 52.

For more information about the File Column, see "File Column" on page 49.

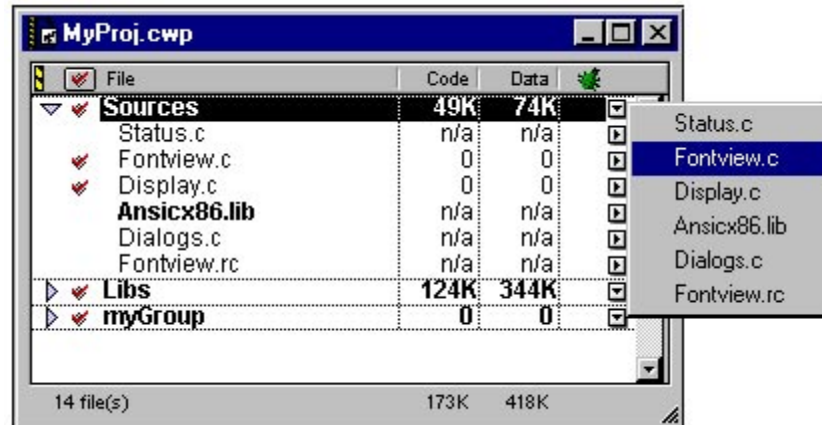
#### **Group File Pop-up Menu**

One way to open a source file is to click on the Group File Pop-up for a particular group so that it pops up, as shown in Figure 4.2. From this pop-up menu you may select the file in the group that you want to open.

You can open a source file by choosing it from the Group File Pop-up menu for the group that contains the file. This works even if the group is collapsed and the file is not visible in the Project window.



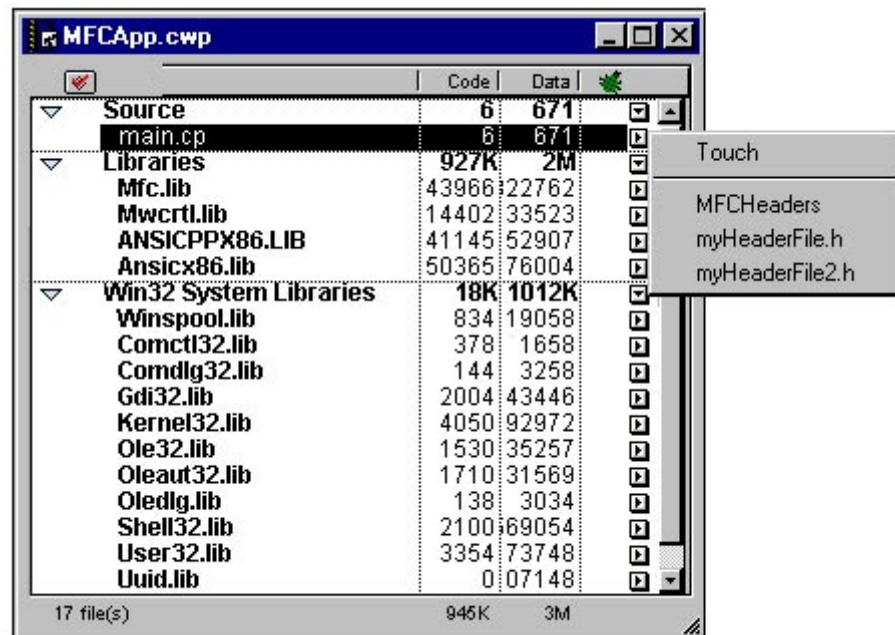
Figure 4.2 Group Files Pop-up Menu in the Project window



### Interfaces Files Pop-up Menu

To open a header or interface file, click on the Interfaces File Pop-up to see a list of files. Select the file you want to open from this list, as shown in Figure 4.3.

Figure 4.3 Interfaces Files Pop-up Menu in the Project window



## Working with Files

### Opening an Existing File

---

Any files shown in **bold** typeface in this pop-up cannot be opened in the Editor because of the file's format.

When the Interfaces File Pop-up is clicked for a library file that is part of your project, you will only have the option to Touch or Un-touch the library file. Since libraries do not contain header files, no header files can be opened from a pop-up corresponding to a library file.

## Opening Files from an Editor Window

To open an interface file from within a source file you are editing, click the Interface Pop-Up Menu at the top right of the Editor Window as shown in Figure 5.2. This pop-up menu lists all interface files used by the source file. Select a file from this menu to open that file in a new Editor window.



---

**NOTE:** *If there are no files available in the menu, it means your text file does not contain source code, or that the source file has not yet been compiled.*

---

Here's a different method. If you're editing any source code file, you can open an interface file mentioned anywhere in the text file with the Open Selection command.

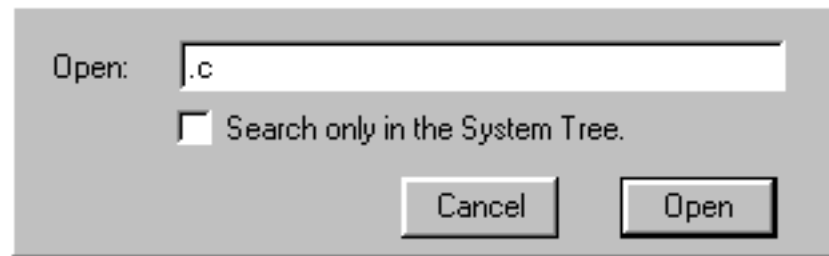
First, select text in the editor window containing the name of the interface file you would like to open. An example of a file name you might see in a C source code file is `stdio.h`. You could select `stdio.h` by double-clicking on the `stdio` portion of the text. Then, choose the Open Selection command from the File Menu.

CodeWarrior then searches for the file and opens the file in an Editor window.

If you're editing a source code file and want to open a file without selecting any text, you may choose the Open File command from the File Menu. This command will use the settings in the Access Paths Panel for the project to search for the file to open.

CodeWarrior then displays an open file dialog, as shown in Figure 4.4. Type the name of the file you wish to search for in the Open editable text field.

**Figure 4.4 Open Selection**



To search only the CodeWarrior directory structure (the paths specified in the System Include Path Pane of the Access Paths Panel), click on the Search only in the System Tree option to turn it on.

If you want to search both System Include Path Pane and User Include Path Pane directory trees (all paths specified in the Access Paths Panel), turn the Search only in the System Tree option off.

If no project is open when you attempt an Open Selection operation, CodeWarrior will search the paths specified by the choices in both the System Include Path Pane and User Include Path Pane.

To learn more about Access Paths and how to configure them, refer to “Access Paths Panel” on page 195 for more information.

## **Saving a File**

This section describes the many ways that CodeWarrior can save files. The topics discussed are:

- Saving One File
- Saving All Files
- Saving Files Automatically
- Renaming and Saving a File
- Backing Up Files

## Working with Files

### *Saving a File*

---

- Saving as a UNIX or DOS text file

#### **Saving One File**

To save your changes to the current Editor file, choose the Save command from the File Menu. CodeWarrior saves your file to disk.

The Save command is dimmed if the window is new and has no data, if the contents of the active window have already been saved, or when the active window is the Project window.

Projects are saved when they are closed, when you quit CodeWarrior, or when the Save A Copy As command is selected.



---

**NOTE:** *If the file is new and untitled, CodeWarrior displays the Save As dialog, described in “Renaming and Saving a File” on page 77. Choose a name and location for your saved file with this dialog.*

---

#### **Saving All Files**

To save your changes to all the files currently open, choose the Save All command from the File Menu.

The CodeWarrior Editor saves all the modified files to disk.

#### **Saving Files Automatically**

CodeWarrior automatically saves the changes to all your modified files whenever you choose the Run, Make, or Bring Up To Date commands from the Project Menu.

This feature can save your work if your program should crash while running, but if you’re experimenting with a change and don’t want to save it, you may want to turn this option off.

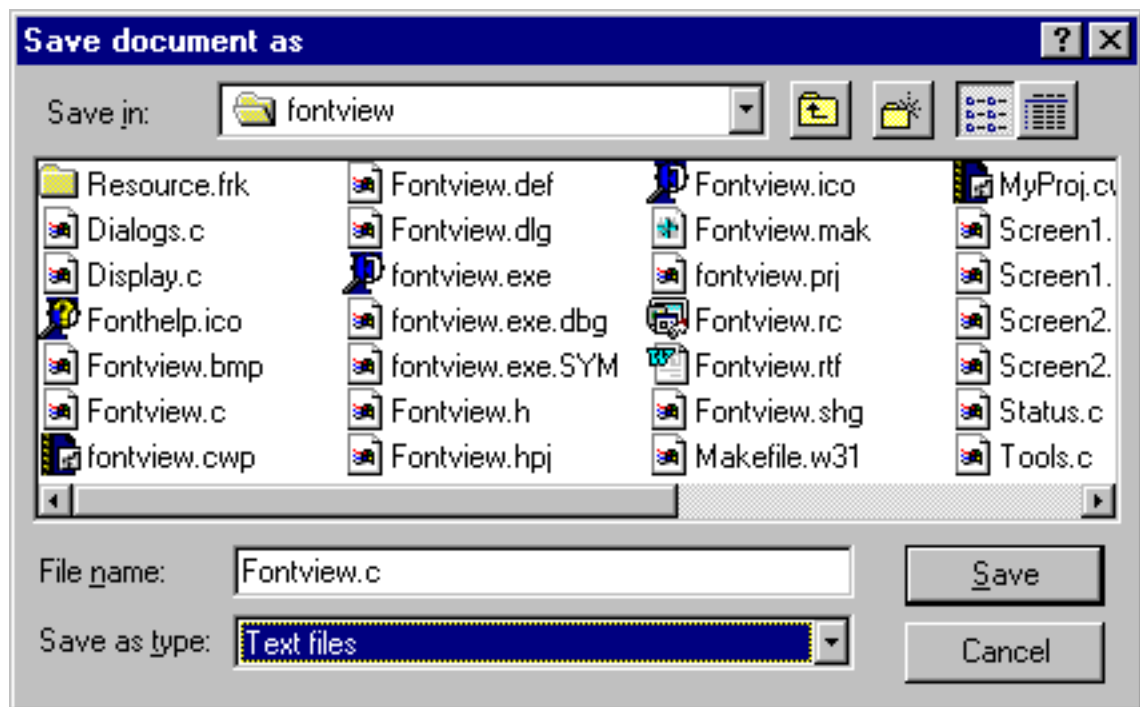
To learn about how to enable or disable this feature, refer to the Save All Before “Update” option in the section titled “Editor Settings Panel” on page 180.

### Renaming and Saving a File

If you want to save a new untitled file or save a file under a new name, use the Save As command on the File Menu. If the file is in the current project, CodeWarrior updates the project to use the new name.

When you choose Save As from the File Menu, CodeWarrior displays the dialog shown in Figure 4.5.

**Figure 4.5** Save As dialog



Choose the file location and name the file, then click the Save button.

CodeWarrior saves the file and changes the name of the Editor window to the name you entered.

If the file is in the current project, CodeWarrior changes the file's entry in the project to match the saved name. If you don't want to

## Working with Files

### *Saving a File*

---

change the project, but still want to save the file, you can read how to do this in “Backing Up Files” on page 78.

### **Backing Up Files**

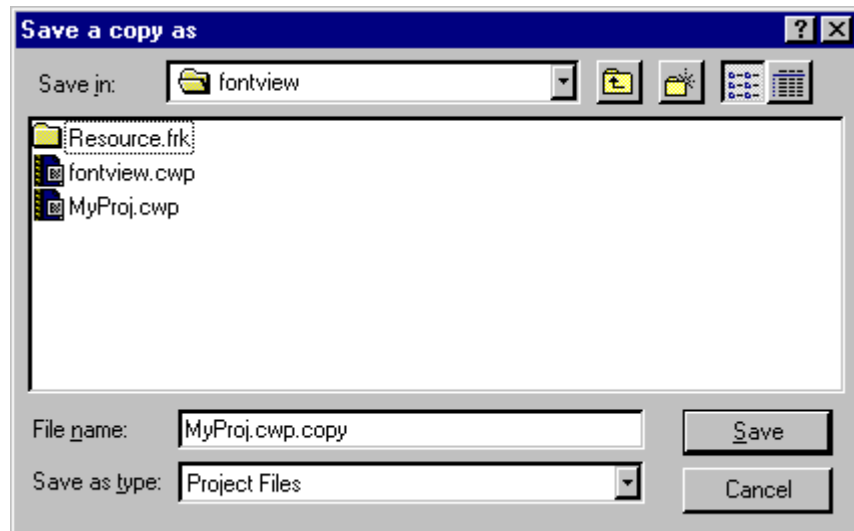
If you want to save a backup copy of a file before you make some changes to the original, use the Save A Copy As command on the File Menu. CodeWarrior creates a copy of the file under a new name that you specify, but leaves the original file unchanged and does not change the currently-open project to use the new file name.

After choosing Save A Copy As from the File Menu, the CodeWarrior Editor displays the dialog shown in Figure 4.5. Specify the file’s new location and choose a unique name for the file. Click Save and CodeWarrior saves a version of the file with your new name. It does not change the file in the Editor window or in the current project.

If the project window is the active window, Save A Copy As allows you to save the project using a new name, or as a text file. You decide which type of project to create using the Save Project As Type pop-up menu shown in Figure 4.6. Saving the project as a text file creates a text file that contains the names of all the files in the project.

When moving a project file, you need to be aware of special considerations. To learn more about how to copy your project’s saved information when you copy your project files, refer to “Moving a Project” on page 64.

**Figure 4.6** Saving a Copy of a Project Window



### **Saving as a UNIX or DOS text file**

When you open a text file originally formatted in a UNIX or Macintosh editor, CodeWarrior automatically converts it to a DOS-compatible text file. When you save the file, CodeWarrior saves it in its original format.

To save a Macintosh, Unix, or DOS text file in a different format, refer to discussion about “Saving as a UNIX or DOS text file” on page 79.

## **Closing a File**

Every Editor window that you have opened is associated with a file on the hard disk. When you close the window, you close the file. You can close all windows or just a single Editor window.

The topics in this section are:

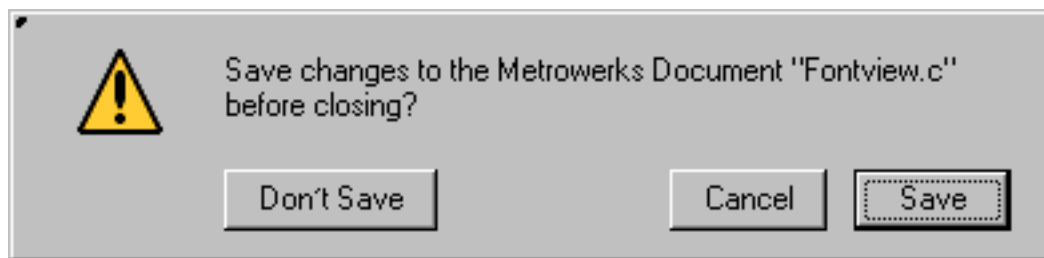
- Closing One File
- Closing All Files

## Closing One File

To close a window, choose Close from the File Menu.

If you close a text file using the File Menu and have not yet saved your changes, CodeWarrior asks if you want to save the changes before closing the window, as shown in Figure 4.7. If you choose to close the file without saving your changes, all changes are lost.

**Figure 4.7** The dialog box for unsaved changes



You can also close a window by clicking the close box of the active window. This is exactly the same as choosing the Close item in the File Menu.

If the active window is the Project window, closing the window automatically saves the project before the window closes. For more on saving project files, consult “Saving a Project” on page 45.

The Close command also saves other properties of the window, such as the size, location, and the selected text in the active window. Refer to “Editor Settings Panel” on page 180 for information on how to configure these options. If the appropriate options are enabled, the next time the source code file is opened, it will occupy the same position on your screen and the same text will be selected.

## Closing All Files

To close all the open Editor windows, use the Close All command on the File Menu.



Close All doesn't close all CodeWarrior windows, just Editor Windows. The Find dialog and the Project Window remain open when using this command.



---

**TIP:** *To close all Editor windows at once, press the Alt key and click on the close box of an Editor window.*

---

## Printing a File

Use the print options in CodeWarrior to print open files, a project file, or the contents of a window.

The topics in this section are:

- Setting Print Options
- Printing a Window

### Setting Print Options

To configure printing options, choose Print Setup from the File Menu. CodeWarrior displays the Print Setup dialog, similar to that shown in Figure 4.8.

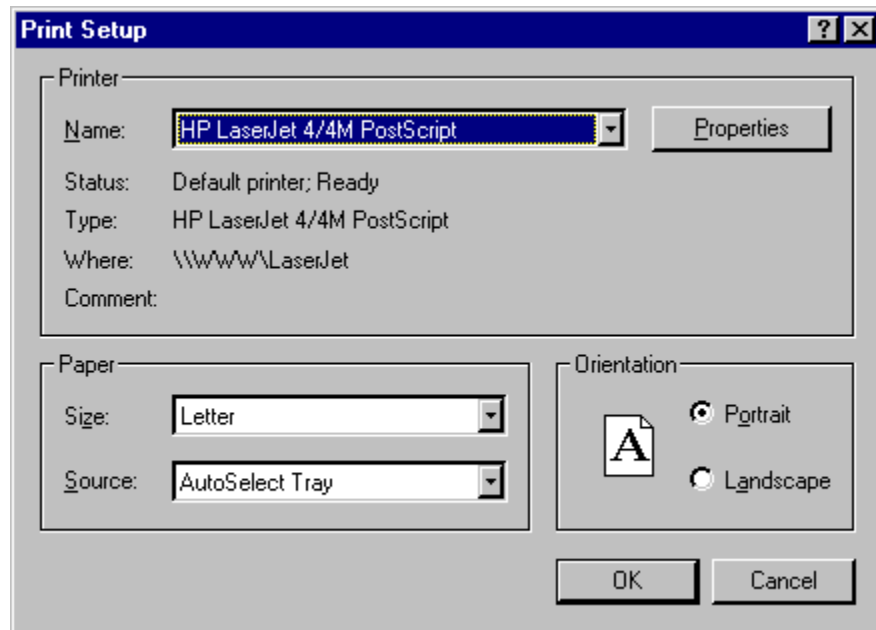
Use this dialog box to select the paper size, orientation, and other settings. The specific settings and options depend on the printer you have connected to your computer. For more information on printing using your printer, consult the documentation packaged with your computer and printer, or the documentation that came with your operating system.

## Working with Files

### *Printing a File*

---

**Figure 4.8** Print Setup dialog

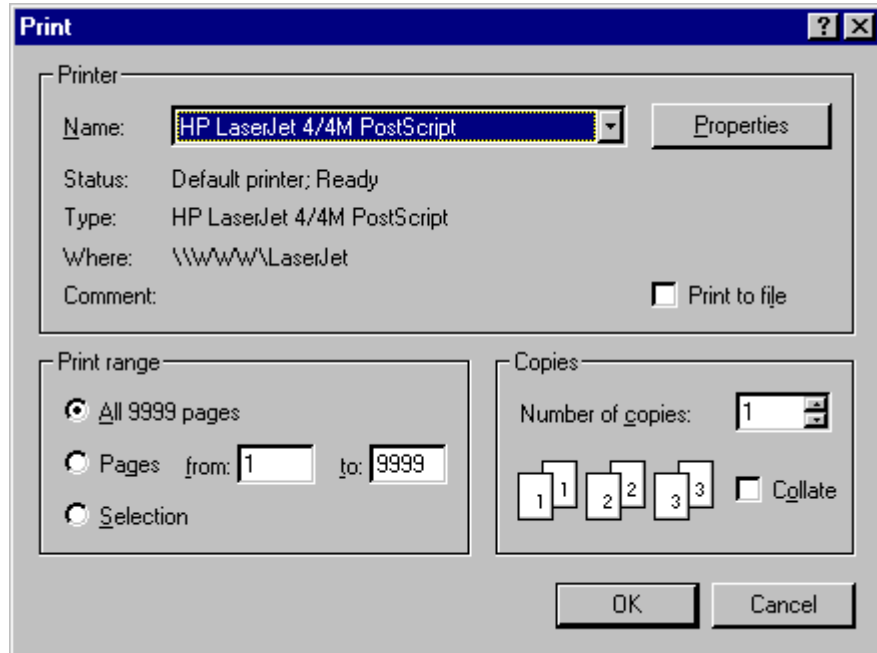


If you Click OK, CodeWarrior saves the options for the next time you print any files.

## Printing a Window

To print a window in CodeWarrior, make the window active and then choose the Print command from the File Menu. A dialog similar to that shown in Figure 4.9 appears. Make any changes you require to the print settings, then click the OK button to begin printing your window.

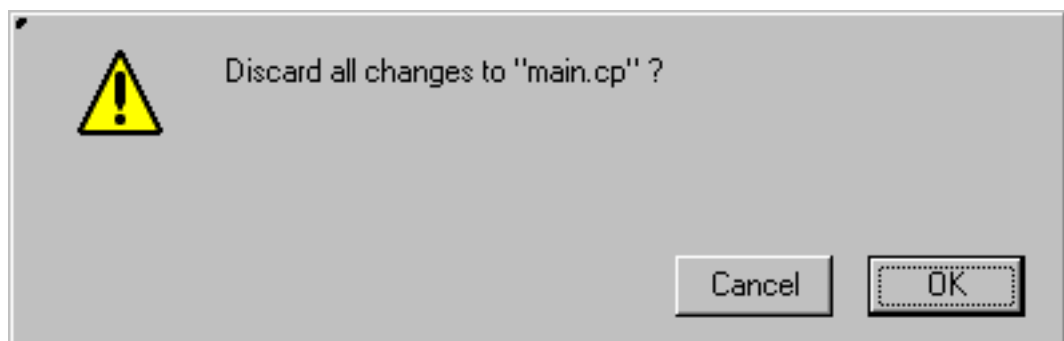
**Figure 4.9** Print dialog



## Reverting to a Previously-Saved File

If you've opened a text file and started editing it, then realize that you don't want to use the changes you've made, you can use the Revert command on the File Menu. When you select this command the dialog shown in Figure 4.10 appears.

**Figure 4.10** Revert to a Previous File



## **Working with Files**

### *Reverting to a Previously-Saved File*

---

If you click the OK button, the last copy of the file you're working with will be opened, and all changes you have made since the last time you saved the file are lost. If you choose Cancel, the file you're working with is not changed or saved to disk, and you can continue editing it.



# Editing Source Code

---

This chapter explains how to use the CodeWarrior text editor to edit your source code.

## Source Code Editor Overview

The CodeWarrior editor is a full-featured text editor specially designed for programmers, with features such as:

- Pop-up menus on every editor window for opening your interface files and navigating your routines quickly
- Colored text highlights for easy identification of comments and keywords in your source files

The topics in this chapter are:

- Guided Tour of the Editor Window
- Editor Window Configuration
- Basic Text Editing
- Navigating in Text

You can also change options that affect the way the editor works. To learn more about how to do this, refer to “Editor Settings Panel” on page 180.

## Guided Tour of the Editor Window

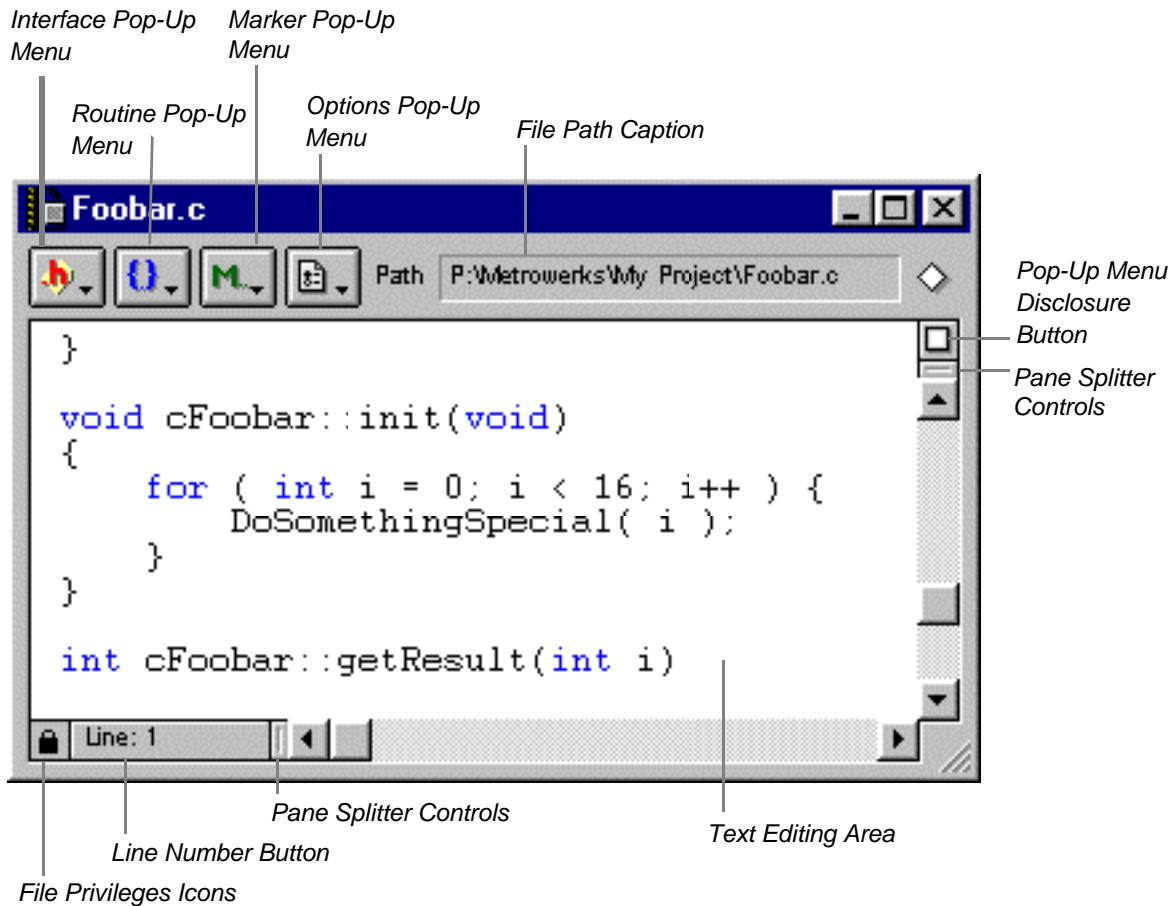
The CodeWarrior Editor, shown in Figure 5.1, contains elements you’ll find useful when viewing and editing your source files.

## Editing Source Code

### Guided Tour of the Editor Window

---

**Figure 5.1** The Editor window



To see an Editor window, create a new text file using the New command on the File Menu.

The sections that follow describe the elements of the Editor window shown in Figure 5.1.

- Text Editing Area
- Interface Pop-Up Menu
- Routine Pop-Up Menu
- Marker Pop-Up Menu
- Options Pop-Up Menu
- File Path Caption

- File Privileges Icons
- Line Number Button
- Pane Splitter Controls
- Pop-Up Menu Disclosure Button

## Text Editing Area

The Text Editing Area of the Editor window is where your text is entered in your new window.

You may select and drag text out of an Editor window to any destination that can accept a drop, such as another open Editor window. You may also drag selected text into an Editor window from other applications that support drag and drop.

For more information about drag and drop operations with text, see “Moving Text (drag and drop)” on page 102.

## Interface Pop-Up Menu



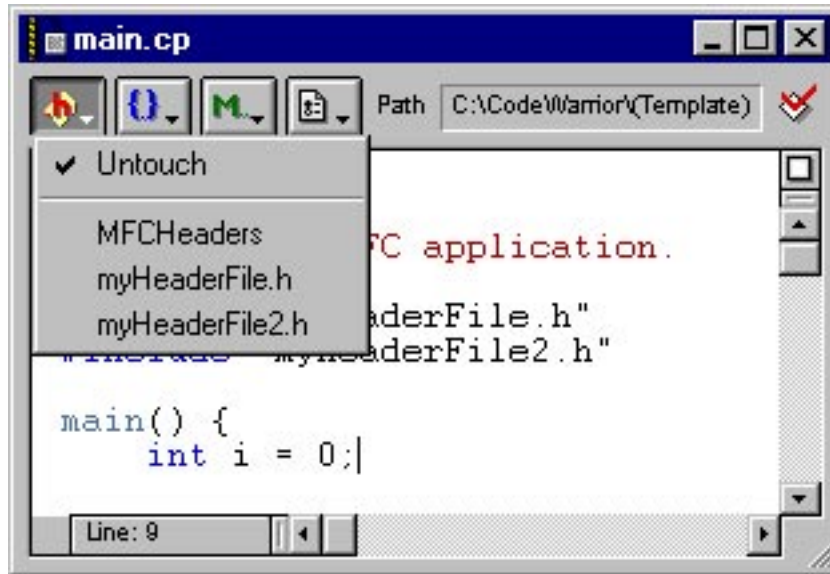
Use the Interface Pop-up Menu shown in Figure 5.2 to open interface or header files referenced by the current file. You can also use the Touch and Untouch commands from this pop-up.

## Editing Source Code

### *Guided Tour of the Editor Window*

---

**Figure 5.2** The Interface Pop-Up menu



To open a file in the list, scroll down to the file you'd like to see and release the mouse button. Note that in order to see a list of files in the menu, the project file must be opened. Note also that some files cannot be opened, such as precompiled header files.

For more information on opening files, see "Opening an Existing File" on page 70.

To cause your file to be recompiled the next time the project is built, you choose the Touch command. If you click on the Interface pop-up again you can deselect the file for compilation with the Untouch command on the menu.

For more information on Touch and Untouch, see "Touching and Untouching Files" on page 213.

## Routine Pop-Up Menu

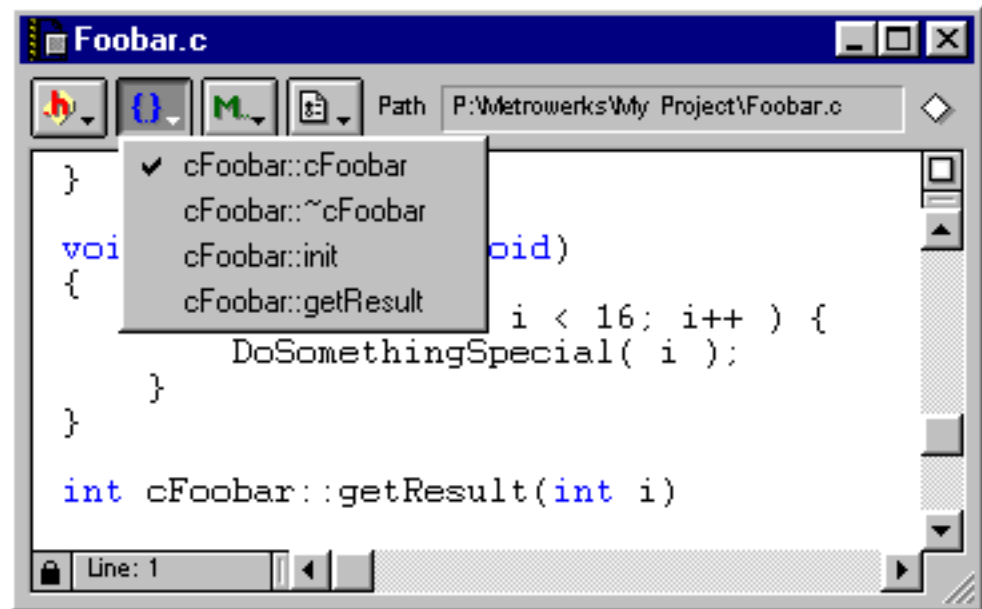


Use the Routine Pop-up Menu shown in Figure 5.3 to set the current location of the text insertion point in your text files.



The Routine pop-up menu lists the routines in your source file. The checked routine in the pop-up tells you where the text insertion point is currently located.

**Figure 5.3** The Routine Pop-Up menu



---

**NOTE:** *If the pop-up is empty, the file is not a source file.*

---

Note that, by default, the menu lists the routines in the order in which they appear in the file. If you'd like to list routines alphabetically, hold down the Alt key as you click on the routine icon.

If you'd like to change the default display order of the routines to alphabetical, enable the Sort Function Popup option. See "Editor Settings Panel" on page 180 for more information about editor options.



---

**TIP:** *If you're editing a Pascal file, procedure names are in italics, routine names are in plain face, and the main program is in bold.*

---

## Editing Source Code

### *Guided Tour of the Editor Window*

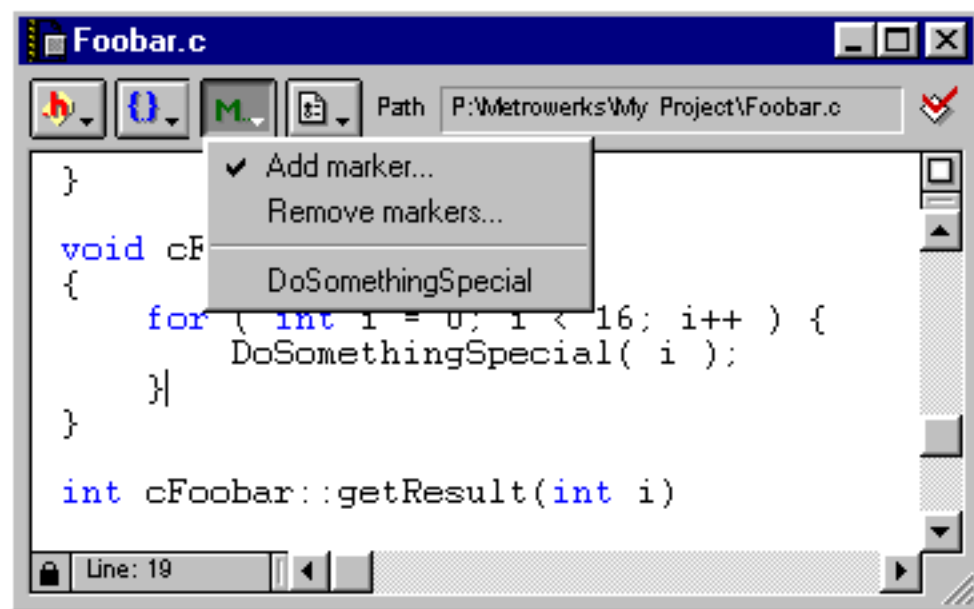
---

## Marker Pop-Up Menu



Use the Marker Pop-up Menu shown in Figure 5.4 to add and remove markers in your text files. These markers are easy to use and convenient for quick access to a certain line of code, to remember where you left off, and for other identification purposes. The item named “DoSomethingSpecial” in the pop-up shown in Figure 5.4 is a marker that was set by the user, as a reminder of a location in the code that needed more work.

**Figure 5.4** The Marker pop-up menu



To learn more about how to set, remove, and use markers, see “Adding, Removing, and Selecting a Marker” on page 109.

## Options Pop-Up Menu

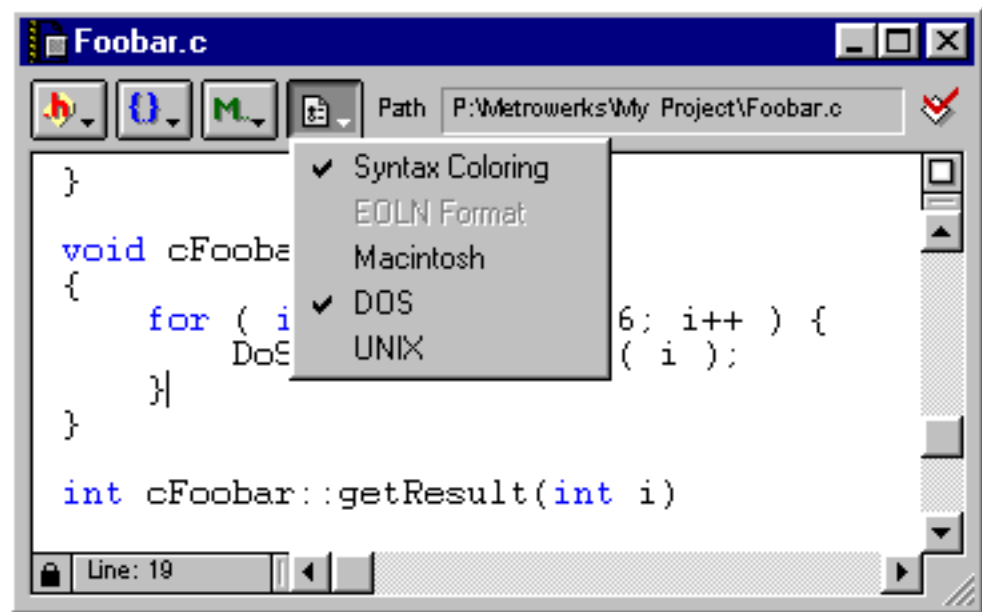


Use the Options Pop-up Menu, shown in Figure 5.5, to choose syntax coloring for the current file, and also to set the format for how to save the file.

The Macintosh, DOS, and UNIX options indicate the type of file currently open in the Editor window. The checkmark indicates the cur-

rent file type. The next time you save the file, CodeWarrior saves it in the format you select.

**Figure 5.5 The Options Pop-up Menu**



The Options Pop-up Menu allows you to change the format of your text file EOL delimiters (UNIX, DOS or Macintosh), and also enable or disable syntax coloring for the window.

For more information on how to use the EOL delimiter options on this pop-up menu, see “Saving as a UNIX or DOS text file” on page 79.

For more information on the Syntax Coloring option shown in this menu, see “Syntax Coloring Panel” on page 185.

## File Path Caption


CodeWarrior automatically displays the directory path of the current file in the File Path Caption, which is at the top right of the window shown in Figure 5.1 on page 86.

## File Privileges Icons

The File Privileges Icons (shown in Table 5.1), indicate the read/write status of the current file. If the box is empty, you can modify the file you're working with. These icons are usually only visible when you are using a source code revision control system for your files.

At this time there is only one icon to reflect the status of files in the Windows environment.

**Table 5.1** File privileges icons

Icon	File Status	Description
	Locked	You cannot modify this file because the Read-only option is turned on in the file's properties.

## Line Number Button

The line number box shown in Figure 5.1 displays the number of the line that contains the text insertion point. You can also use this button to go to another line in the file.

For information about setting the text insertion point on another line, see “Going to a Particular Line” on page 114.

## Pane Splitter Controls

Pane Splitter Controls split the Editor windows into panes so you can view different portions of a file in the same window.

You use these controls to adjust the sizes of the panes after you've created them. Figure 5.8 on page 96 shows an Editor window with multiple panes.

For more information on this topic, see “Splitting the Window into Panes” on page 95.

## **Pop-Up Menu Disclosure Button**

The Pop-up Menu Disclosure Button allows you to modify the Editor window to create a view that puts all the pop-up controls along the bottom of the Editor window (see Figure 5.7 on page 95).

For more information on using the Pop-up Menu Disclosure Button, refer to “Seeing Window Controls” on page 93.

## **Editor Window Configuration**

The Editor allows you to customize your view of the file you’re working with. In this section, you’ll learn about the following topics:

- Setting Text Size and Font
- Seeing Window Controls
- Splitting the Window into Panes
- Saving Window Settings

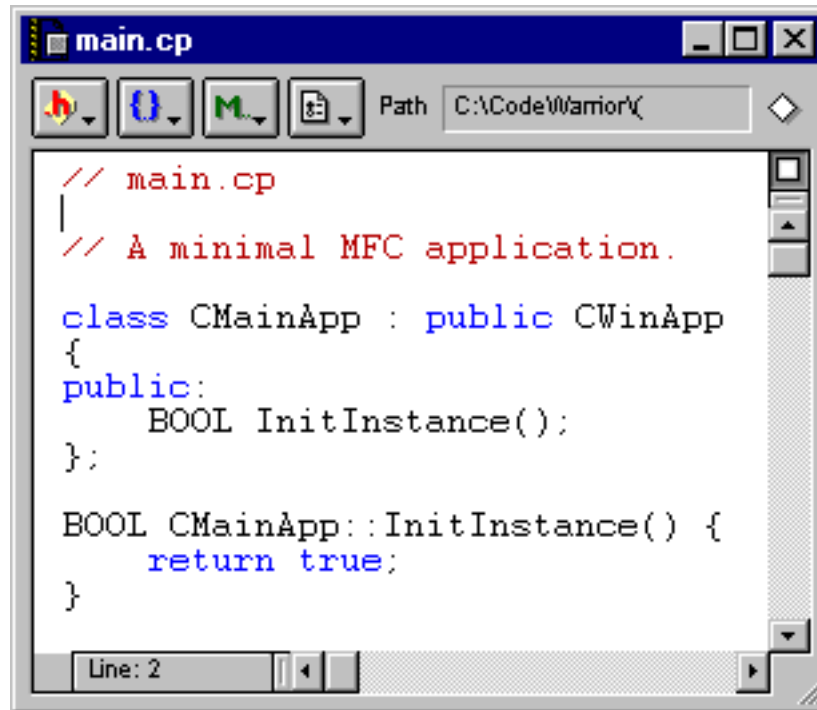
### **Setting Text Size and Font**

You set the size or font used to display text in an Editor window in the Fonts & Tabs preference panel. For more information on this topic, refer to “Fonts and Tabs Panel” on page 184.

### **Seeing Window Controls**

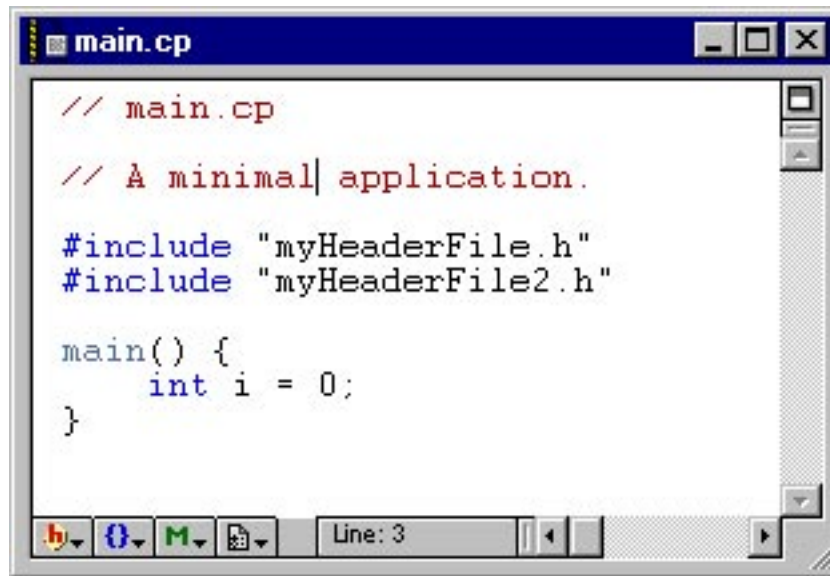
To toggle the row of pop-up menus and controls that appear along the top of the Editor window to the bottom of the window, you can click the Pop-Up Menu Disclosure Button, as shown in Figure 5.6.

**Figure 5.6** Editor Window When Clicking the Pop-up Disclosure Button



Clicking this button changes the Editor window to look like Figure 5.7. Note that the File Path Caption is no longer visible.

**Figure 5.7** Pop-ups Along the Editor Window Bottom (After Clicking)



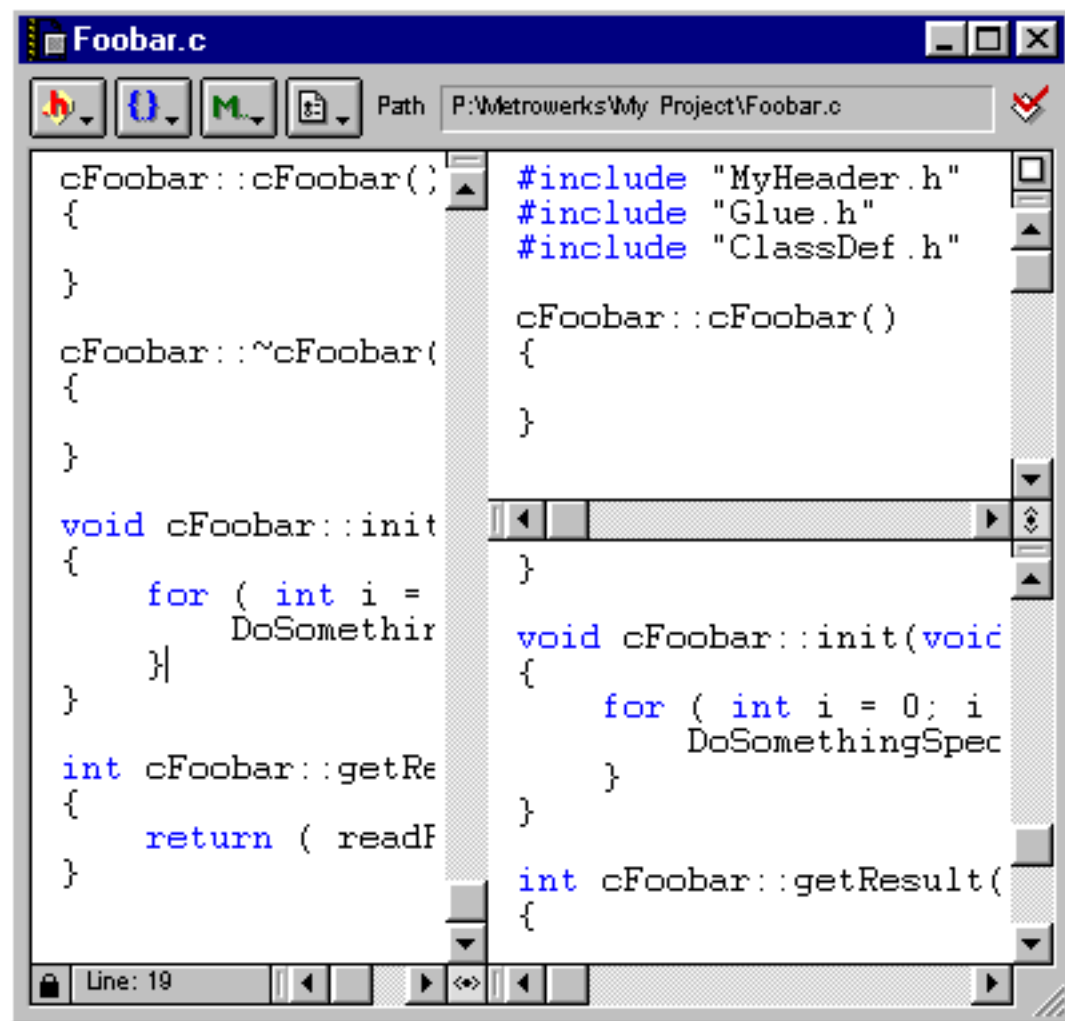
To show the pop-up menus along the top of the Editor window again, click the Pop-up Menu Disclosure Button once more.

You can make your choice for pop-up menu buttons the default choice for Editor windows. To do so, choose the Save Default Window item in the Window menu. For more information on this topic, refer to "Save Default Window" on page 263.


## Splitting the Window into Panes

You can split the Editor window into panes, so you can view different parts of a file in the same window, as shown in Figure 5.8. This section describes creating, adjusting, and removing multiple panes.

**Figure 5.8** Multiple panes in a window



### Creating a new pane


 To create a new pane in an Editor window, click and drag a Splitter Bar. Splitter bars are on each scroll bar of the Editor window, on the top and left sides.

As you drag a Splitter Bar, grayed lines track your progress and indicate where the new pane will go. When you release the mouse button, the Editor creates a new pane.

Double-click on the Splitter Bar to split a pane into two equal parts.




### Resizing a pane

 To change the sizes of the panes in an Editor window, click and drag the pane resize boxes.

As you drag a resize box, grayed lines indicate your progress. When you release the mouse button, the Editor redraws the panes in their new positions.

### Removing a pane

 To remove a pane from an editor window, click and drag a Resize Box all the way to an edge of the window.

As you drag the Resize Box, grayed lines indicate your progress. If you drag close to the edge of the window, the gray lines are no longer displayed. If you release the mouse button at that time, the editor removes one of the panes from the window.

Double click on the Resize Box to remove a split.

## Saving Window Settings

The Save Default Window menu command on the Window Menu allows you to save the settings for your currently-active Editor window. This allows you to maintain the same settings the next time the window is opened.

The settings saved are the size and location of the window, as well as the setting of the Pop-Up Menu Disclosure Button. Any new windows you open will have these new default settings. Any windows you presently have open will need to be closed and reopened to get the new settings.

You must save each window's setup individually, while that window is the active window.

To learn more about configuring Editor window settings, refer to "Fonts and Tabs Panel" on page 184.

To learn about using this command with the CodeWarrior Browser, see “Saving a Default Browser” on page 174.

## Basic Text Editing

CodeWarrior gives you lots of help in editing source files, all of it described in the sections that follow.

The topics in this section are:

- Basic Editor Window Navigation
- Adding Text
- Deleting Text
- Selecting Text
- Moving Text (drag and drop)
- Using Cut, Copy, Paste, and Clear
- Balancing Punctuation
- Shifting Text Left and Right
- Undoing Changes
- Controlling Color

### Basic Editor Window Navigation

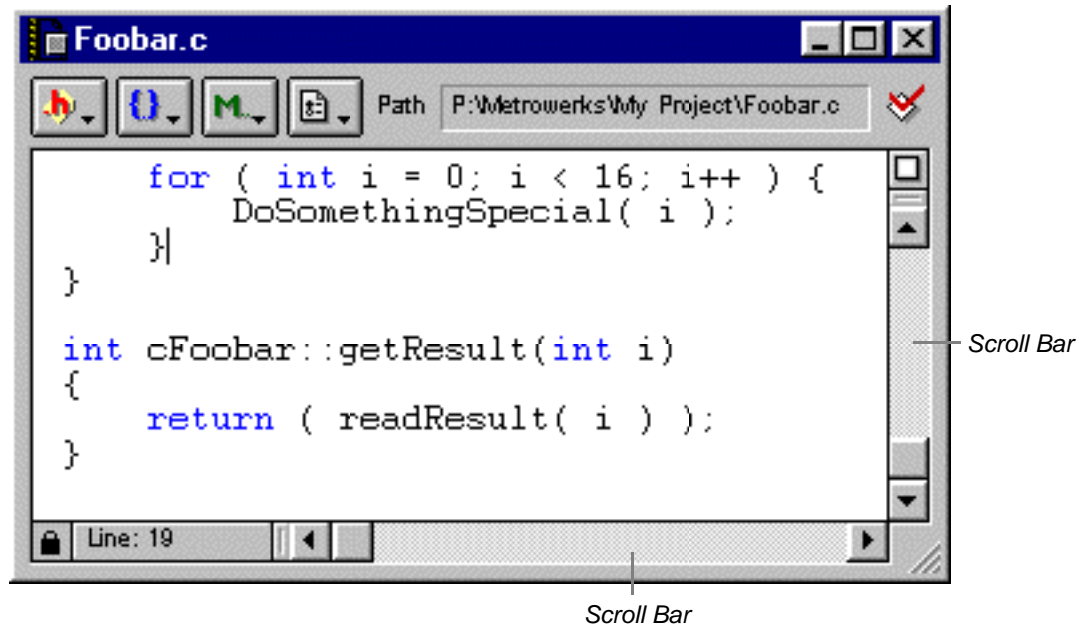
CodeWarrior gives you several ways to move the text insertion point in a file to get to where you want. It would be useful to review this section again after you become more familiar with the CodeWarrior Editor’s features.

#### Scroll bar navigation

The CodeWarrior Editor provides dynamic scrolling, which means that as you drag the Scroll Bar at the far right of the Editor window, the CodeWarrior Editor updates the contents. See Figure 5.9 to see what the Scroll Bars look like.

You can see the text scroll by in the Editor window as you click and drag the Scroll Bar. Use the Scroll Bar at the bottom of the editor window to scroll left and right.

**Figure 5.9** Scroll Bars in an Editor Window



CodeWarrior lets you configure how the scroll bars affect the window view when you drag the scroll bar box around. You can modify the way that scrolling behaves in the Editor windows of your project by changing the Dynamic Scroll option. To learn how to do this, refer to “Dynamic Scroll” on page 181.

### Keyboard navigation

Table 5.2 describes how to move the insertion point around in a file with function keys.

**Table 5.2** Text navigation with the keyboard

To move insertion point to	Press
Beginning of the line	Ctrl–Left Arrow
End of the line	Ctrl–Right Arrow
Beginning of the file	Ctrl–Up Arrow
End of the file	Ctrl–Down Arrow

Table 5.3 describes how to scroll to different locations in a file, without moving the insertion point. Note that some of the keys listed in the table may not be on your keyboard, depending on what kind of keyboard you have.

**Table 5.3** Scroll with the keyboard

To scroll to the...	Press this...
Previous page	Page Up
Next page	Page Down
Beginning of the file	Home
End of the file	End
Insertion point	Left Arrow, and Right Arrow

## Adding Text

To add text to a file you’ve opened, click once in the Text Editing Area of the window to set the new location of the text insertion point. After you see the insertion point at the new location, you may begin typing on the keyboard to enter text.

To read about different ways to move the insertion point in an Editor window, see “Basic Editor Window Navigation” on page 98.

## Deleting Text

There are several different methods for deleting text.

To delete text that you just typed, hit the Backspace key.

To delete more than one contiguous character at a time, select the text you want to delete and hit the Backspace key.

The Delete key on your keyboard can also be used for deleting text. You can select text and then press Delete. Or, you can use the Delete key to delete the character to the right of the text insertion point. This differs from Backspace, which deletes text to the left of the text insertion point.

If you don't know how to select text, you can learn how by reading "Selecting Text" on page 101.

## **Selecting Text**

There are several different ways to select text in the Editor window.

To select a word, double-click on the word.

To select a line, triple click anywhere in the line.

You can also select text by holding down the Shift key while pressing any of the shortcuts listed in Table 5.2.

To select a range of text, click and drag the mouse in a portion of your window where there is text. Another way to select a range is to set your text insertion point somewhere in your window. Then, press the Shift key and click at the place in your text where you want the range to end. All text between the insertion point and your Shift-click location will be selected.

To list and display an entire routine in the Editor window, press the Shift key while selecting a routine in the Routine Pop-Up Menu. This is particularly useful for copy and paste operations and for using drag and drop to move code around in your file.

For more information about using drag and drop with text, see "Moving Text (drag and drop)" on page 102.

## Editing Source Code

### Basic Text Editing

---

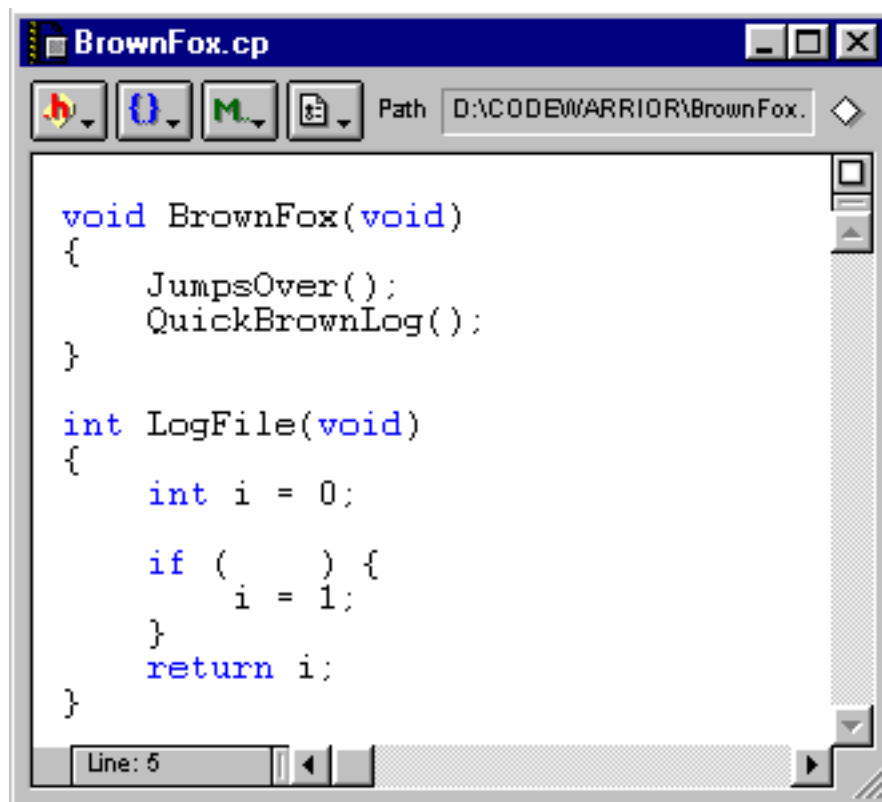
You can also select blocks of code quickly using the Balance command. To learn how to do this, refer to “Balancing Punctuation” on page 106.

## Moving Text (drag and drop)

If you have some text in your Editor window that you would like to move to a new location, you can use the drag and drop features of the Editor to do it. In order to use drag and drop editing, this feature must be enabled in the IDE. To learn more about turning this feature on or off, refer to “Editor Settings Panel” on page 180.

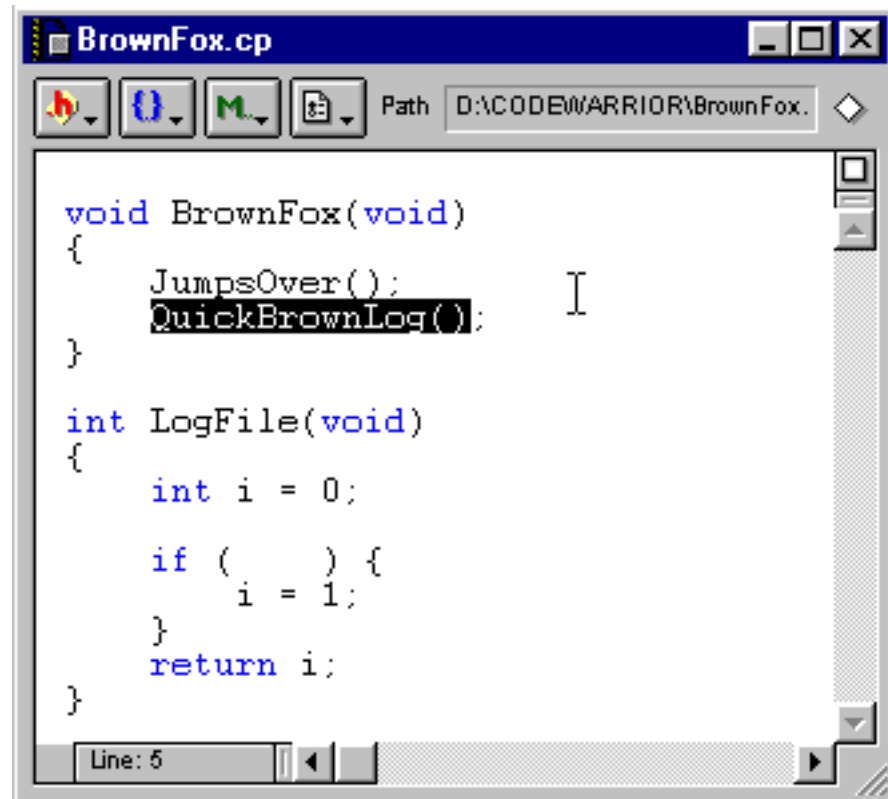
Figure 5.10 shows a file before any rearrangement has been done on the text. Let’s see how we can modify this text using drag and drop.

**Figure 5.10** Original File Before Drag and Drop



First, select the text you want to move. Figure 5.11 shows a window with the text we're going to move. For information on how to select text, see "Selecting Text" on page 101.

**Figure 5.11** File with Text Selected



Next, click on the selected text and drag it to a new location in the file. You will see the text insertion point blink in the window as you move to a new location on each line of the file. The insertion point indicates where the text will be inserted when you release the mouse. Figure 5.12 shows what the screen looks like as you drag the selected text to a new location.

## Editing Source Code

### Basic Text Editing

---

**Figure 5.12** File with Drag and Drop in Progress

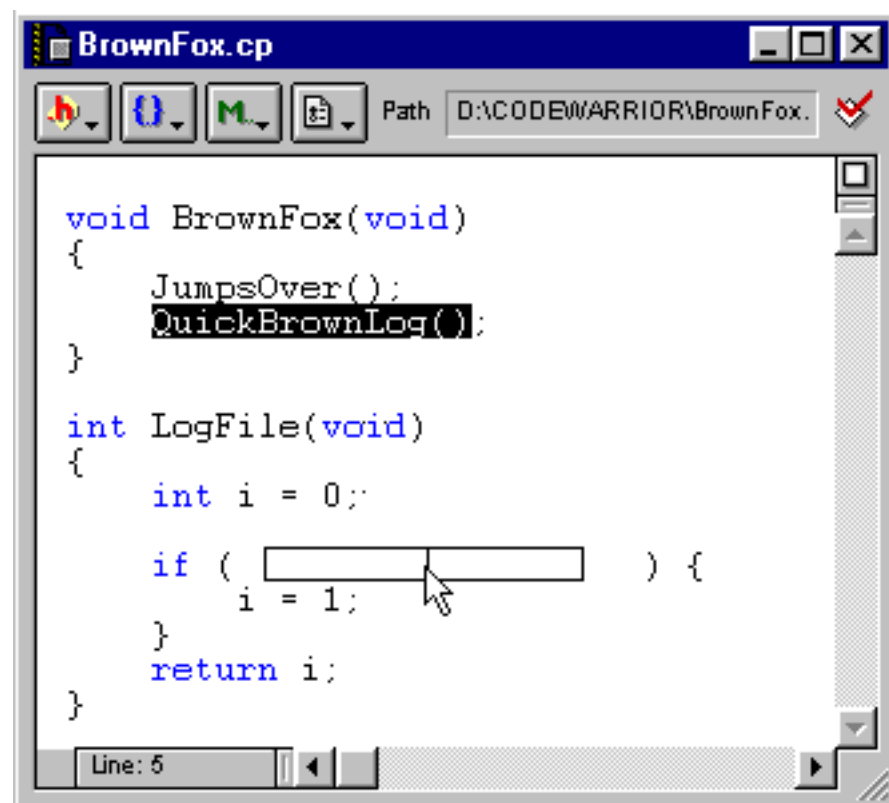
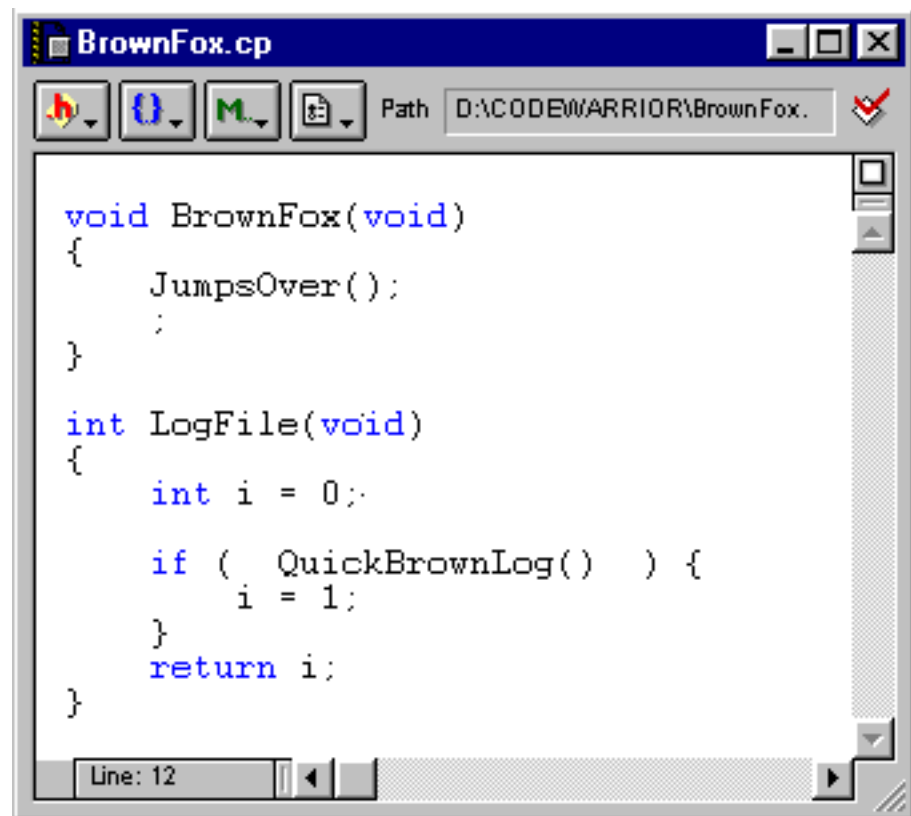




Figure 5.13 shows the final result after dragging the text selection down to a new line.

**Figure 5.13** Final Drag and Drop-Edited File



## Using Cut, Copy, Paste, and Clear

There are standard menu commands available on most computer applications, called Cut, Copy, Paste, and Clear. In CodeWarrior, these commands appear on the Edit Menu.

You use these commands to remove text, or to copy and paste in a window, between windows, or between applications.

For more information about these commands, refer to “Edit Menu” on page 248.

## Balancing Punctuation

When you're editing source code, you must make sure that every parenthesis ( ( ) ), bracket ( [ ] ), and brace ( { } ) has a mate, or the compiler could misinterpret your code or give you an error.

CodeWarrior provides several checks that help you balance these elements correctly.

To check for balanced parentheses, brackets, or braces, place the insertion point in the text you want to test. Then, choose Balance from the Edit Menu. Alternatively, you can double-click on a parenthesis, bracket, or brace character that you want to test.

The CodeWarrior Editor searches starting from the text insertion point until it finds a parenthesis, bracket, or brace, then it searches in the opposite direction until it finds the matching half. When it finds the match, it selects the text between them. If the insertion point isn't enclosed or if the punctuation is unbalanced, the computer beeps.



---

**NOTE:** Use the Balance command as described in the step above to select blocks of code quickly.

---

## Using automatic balancing

You can have the CodeWarrior Editor check for balanced punctuation automatically. If you would like to learn more about checking the balance of code automatically as you type, refer to “Balance While Typing” on page 181.

## Shifting Text Left and Right

Use the Shift Left and Shift Right commands on the Edit Menu to shift a block of text to the left or right.

To shift blocks of text, select a block of text. If you don't know how to do this, refer to “Selecting Text” on page 101. After selecting, choose Shift Right or Shift Left from the Edit Menu.

The CodeWarrior Editor shifts the selected text one tab stop to the right or left by inserting or deleting a tab at the beginning of every line in the selection.

To learn more about controlling the number of spaces the text is indented, refer to “Fonts and Tabs Panel” on page 184.

## **Undoing Changes**

The CodeWarrior editor supplies ways to Undo mistakes as you edit a file.

### **Undoing the last edit**

The Undo command reverses the effect of your last action. The name of the Undo command on the Edit Menu varies depending on what you last did. For example, if you just typed in some text, the command changes to Undo Typing.

### **Undoing and redoing multiple edits**

When the Use Multiple Undo option is enabled, you can Undo and Redo multiple previous actions by continuing to choose the Undo or Redo commands.

For instance, if you Cut a word, then Paste it, then type some text, you can backtrack all those actions by choosing Undo. The first Undo removes the text you typed, the second Undo unpastes the text you pasted, and the third Undo uncuts the text you Cut, therefore returning the text to its original condition.

You can perform those activities again in the same order by choosing the Redo command three times.

To learn how to enable the Use Multiple Undo option, refer to “Use Multiple Undo” on page 182.



---

**WARNING!** *Handle multiple Undo carefully. If the Use Multiple Undo option is turned on and you use Undo repeatedly, you can lose text permanently.*

---

### Reverting to the last saved version of a file

The Revert command on the Edit Menu returns a file to its last saved version. To learn more about how to revert to the previous version of a file, refer to “Reverting to a Previously-Saved File” on page 83.

### Controlling Color

You can use color to highlight many elements in your source code, such as comments, keywords, and quoted character strings. Highlighting these elements helps you identify them in the text, so you can check your spelling and syntax as you type by recognizing color patterns. For information on configuring color syntax options, see “Syntax Coloring Panel” on page 185.

You can also highlight custom keywords, which are in list of words you designate. See “Syntax Coloring Panel” on page 185 for instructions on configuring the Editor to do this for you.

## Navigating in Text

The CodeWarrior Editor provides several methods for navigating in a file that you are editing.

This section covers these methods:

- Finding a Routine
- Adding, Removing, and Selecting a Marker
- Opening a Related File
- Going to a Particular Line
- Using Go Back and Go Forward

In addition, the integrated code browser has many powerful techniques for navigating through your code. To learn more about using the CodeWarrior Browser, refer to “Browser Overview” on page 145.

## Finding a Routine



Click the Routine icon to display the Routine pop-up menu, discussed in “The Routine Pop-Up menu” on page 89, then select the routine you want to go to.



---

**NOTE:** *If the pop-up is empty, the file is not a source code file.*

---

## Adding, Removing, and Selecting a Marker

You can Add or Remove a marker in any of your text files using the facilities built into the CodeWarrior Editor. Markers are useful for setting places in your file that you can jump to quickly, or for leaving notes to yourself about work in progress on your code.

### Adding a Marker

The example text file in Figure 5.14 has one marker set in it named “InitInstance”, as you see by looking at the selections on the Marker Pop-Up Menu.

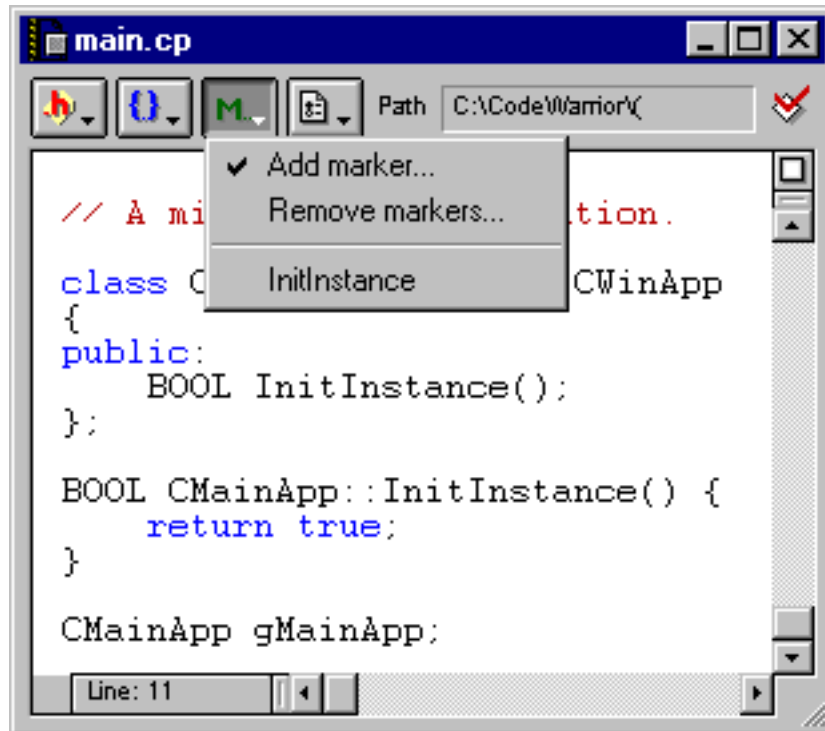
To add a marker, move the text insertion point to the location in the text you want to mark, then click on the Marker Pop-Up Menu and choose Add marker from the pop-up.

## Editing Source Code

### Navigating in Text

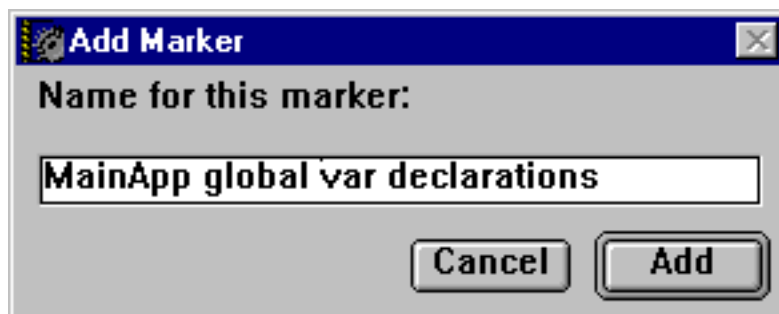
---

**Figure 5.14** File Marker Menu



After choosing Add marker from the pop-up, you will see the Add Marker dialog shown in Figure 5.15.

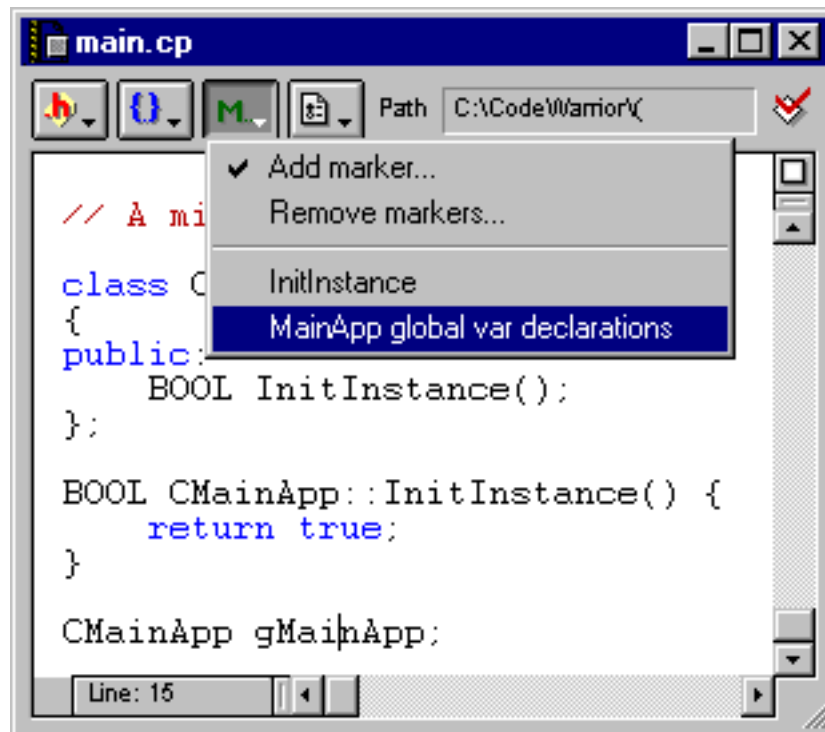
**Figure 5.15** Add Marker Dialog



Enter text in the Add Marker dialog to mark your insertion point location in the file with a note, comment, routine name, or other text that would be helpful to you.

When you are through adding your text note, click Add and your marker will be visible in the Marker Pop-Up Menu the next time you pop it up, as shown in Figure 5.16.

**Figure 5.16** Example Text File with a Marker Added



There is another method for marking files on a more permanent basis. The `#pragma mark` directive, for C/C++ language programs only, can be used to leave markers in a file wherever the following syntax is inserted in the file. Unlike the markers we've been talking about in this section, these markers don't appear in the Marker Pop-up Menu. Instead, markers created with `#pragma mark` appear in the Routines Pop-up menu

When embedded in your file, this example adds myMarker to the Routine Pop-Up Menu automatically when the file is opened in the Editor.

## Editing Source Code

### *Navigating in Text*

---

---

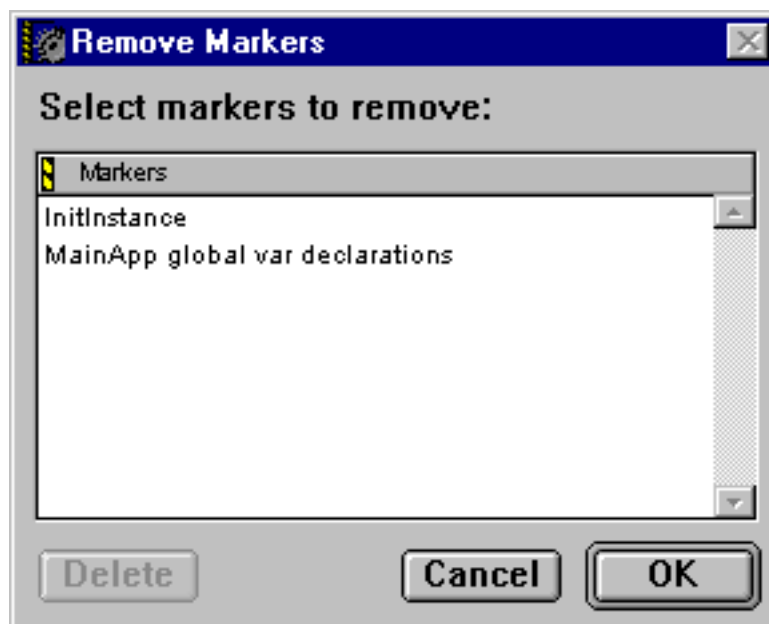
```
#pragma mark myMarker
```

---

### Removing a Marker

To remove a marker, click the Marker Pop-Up Menu and choose the Remove markers command. The dialog shown in Figure 5.17 is shown, and you may select the marker you wish to delete. After you select the marker, click Delete to remove it permanently from the marker list.

**Figure 5.17** Remove Markers



### Jumping to a Marker

Click the Marker Pop-Up Menu and choose the name of the marker from the list shown on the pop-up to set the text insertion point at the location of the marker.

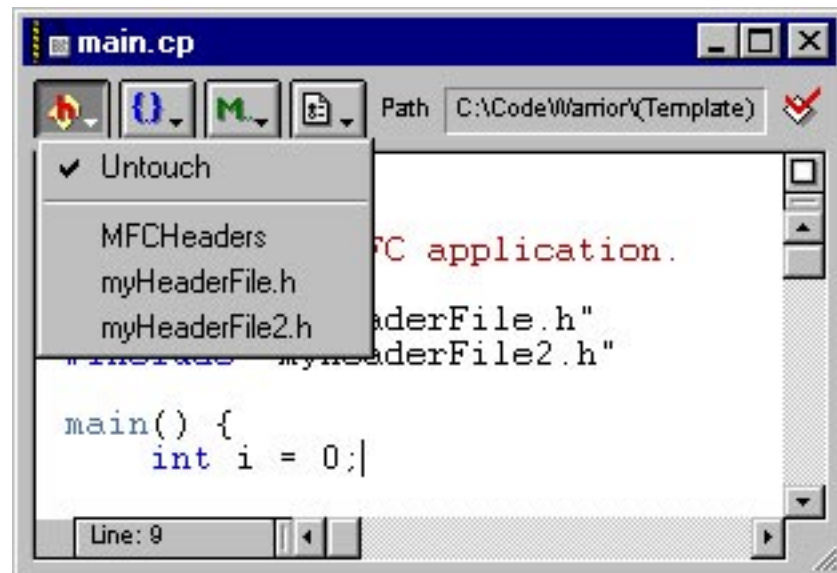


## Opening a Related File

There are a few ways to open files related to the active Editor window. For example, if you are looking at a C++ .cpp source code file and want to view a .h header file that is used by the .cpp file, there are different ways to do this.

Use the Interface Pop-Up Menu shown in Figure 5.18 to open interface or header files referenced by the current file. You can also use the Touch and Untouch commands from this pop-up.

**Figure 5.18** The Interface pop-up menu




To open a file in the list, choose the corresponding item from the menu.

There is another method for opening an interface or header file that your source code file uses. To open the related file, you type a special key combination after selecting the file name in the active window. To learn more about this method for opening files, refer to “Opening an Existing File” on page 70.

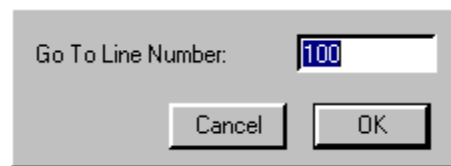
To learn about the Touch and Untouch commands, refer to “Touching and Untouching Files” on page 213.

## Going to a Particular Line

You can go to specific line in an Editor window if you know its number. Lines are numbered consecutively, with the first line designated as line 1.

 Click the Line Number Button on the Editor window to open the Go To Line Number dialog shown in Figure 5.19. Then enter the number of the line you want to go to and select OK.

**Figure 5.19** Go to Line Number Dialog



## Using Go Back and Go Forward

The Go Back and Go Forward commands are only available when you use the Browser. If you already have the Browser enabled you can see “Go Back and Go Forward” on page 169 for information about how to use these commands.

If you aren’t using the Browser and want to learn how to use it, see “Browser Overview” on page 145.



# Searching and Replacing Text

---

This chapter explains how to use the CodeWarrior facilities to search and replace text in files.

## Searching and Replacing Text Overview

CodeWarrior provides comprehensive search and replace features with the Find window. You can search for and replace text in a single file, in every file in a project, or in any combination of files. You can also search for regular expressions, such as those used in UNIX's `grep` command.

CodeWarrior also provides facilities for searching text without using the Find window, so that you can do quicker searches.

The topics in this chapter are:

- Guided Tour of the Find Window
- Searching for Selected Text
- Searching and Replacing Text in a Single File
- Searching and Replacing Text in Multiple Files
- Using Regular Expressions (`grep`)

## Guided Tour of the Find Window

The Find window, shown in Figure 6.1 and Figure 6.4 on page 122, is a versatile feature of the CodeWarrior development environment. To show the Find window, choose the Find command under the Search Menu. With this window you do text searching through a

## Searching and Replacing Text

### *Guided Tour of the Find Window*

---

single file or multiple files in your project. You can search and replace text strings and text substrings (using pattern matching), using groups of different files that you specify.

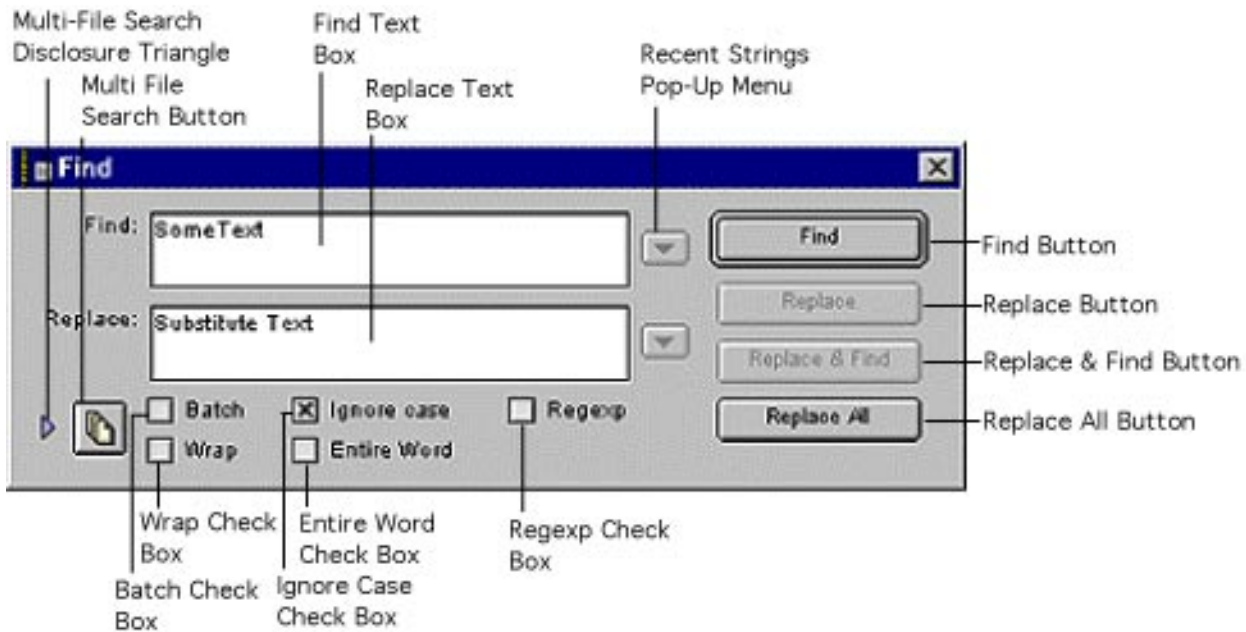
There are two different sections in the Find window.

- Search and Replace Section
- Multi-File Search Section

### Search and Replace Section

This section presents a short tour of the search and replace user interface items in the Find window shown in Figure 6.1. The items in the window are:

**Figure 6.1 The Find Dialog Search and Replace Section**



- Find Text Box
- Replace Text Box
- Multi-File Search Button
- Multi-File Search Disclosure Triangle

- Recent Strings Pop-Up Menu
- Find Button
- Replace Button
- Replace & Find Button
- Replace All Button
- Batch Check Box
- Wrap Check Box
- Ignore Case Check Box
- Entire Word Check Box
- Regexp Check Box

### **Find Text Box**

The Find Text Box is one of the editable text fields in the Find window, shown in Figure 6.1. It allows you to input text that you want to find in a file.

You can use the Cut, Paste, Clear, and Copy commands with the Find Text Box. These commands are documented in the section called “Edit Menu” on page 248.

### **Replace Text Box**

The Replace Text Box is one of the editable text fields in the Find window, shown in Figure 6.1. It allows you to input text for substitution in place of text that you are searching for in a file.

You can use the Cut, Paste, Clear, and Copy commands with the Find Text Box. These commands are documented in the section called “Edit Menu” on page 248.

### **Recent Strings Pop-Up Menu**

The Recent Strings Pop-up Menu is shown in Figure 6.2. It contains strings that were recently used for searches.

There are actually two of these pop-ups. Each pop-up is to the right of both the Replace Text Box and the Find Text Box. When you select

## Searching and Replacing Text

### *Guided Tour of the Find Window*

---

one of the strings from the pop-up it is substituted in the appropriate editable text field.

**Figure 6.2 Recent Strings Pop-up Menu**



#### **Find Button**

The Find Button is one of the buttons in the Find window, shown in Figure 6.1 on page 116. It allows you to begin a text search operation once you set the other Find window controls, and have completed certain required fields in the Find window.

To learn more about finding text, see “Searching for Selected Text” on page 125.

#### **Replace Button**

The Replace Button is one of the buttons in the Find window, shown in Figure 6.1 on page 116.

When you enter text in the Replace Text Box and the Find Button is clicked, CodeWarrior will search for text matching that described by the other control settings, and the text in the Find Text Box. If text matching the Find Text Box text is found, the Replace Button can be clicked to replace the found text with that shown in the Replace Text Box.

To learn more about searching and replacing text, see “Replacing Found Text” on page 130.

### Replace & Find Button

The Replace & Find Button is shown in Figure 6.1 on page 116. This button behaves much like the Replace Button, but also initiates another Find operation after the text substitution is performed.

To learn more about searching and replacing text, see “Searching and Replacing Text in a Single File” on page 126 and “Searching and Replacing Text in Multiple Files” on page 133.

### Replace All Button

The Replace All Button is shown in Figure 6.1 on page 116. This button behaves much like the Replace Button, but replaces *every* occurrence of the Find Text Box text with the Replace Text Box in the appropriate window.

To learn more about searching and replacing text, see “Replacing Found Text” on page 130.

### Batch Check Box

The Batch Check Box is shown in Figure 6.1 on page 116. Selecting this check box causes the results of the Find command to print into the Message window.

If you’d like to learn more about the role of the Batch Check Box in searching, see “Using Batch Searches” on page 132.

### Wrap Check Box

The Wrap Check Box is shown in Figure 6.1 on page 116. This check box causes the search to continue from the beginning of the file once the search reaches the end.

To learn more about this feature, consult “Controlling Search Range” on page 129.

### Ignore Case Check Box

The Ignore Case Check Box is shown in Figure 6.1 on page 116. This check box causes the CodeWarrior search engine to disregard the

## Searching and Replacing Text

### *Guided Tour of the Find Window*

---

case (uppercase or lowercase) of the text entered into the Find Text Box.

To learn more about this feature, consult “Controlling Search Parameters” on page 129.

#### **Entire Word Check Box**

The Entire Word Check Box is shown in Figure 6.1 on page 116. This check box causes the search engine to ignore occurrences of the text in the Find Text Box that occur within words.

To learn more about this feature, consult “Controlling Search Parameters” on page 129.

#### **Regexp Check Box**

The Regexp Check Box is shown in Figure 6.1 on page 116. This check box causes the search engine to Interpret the Find Text Box string as a regular expression.

CodeWarrior’s regular expressions are similar to the regular expression for `grep` in UNIX™. To learn more about this feature, refer to “Using Regular Expressions (`grep`)” on page 140.

#### **Multi-File Search Disclosure Triangle**

The Multi-File Search Disclosure Triangle is shown in Figure 6.1 on page 116. Clicking this triangle exposes the Multi-File Search Section of the Find window, so that the window looks as shown in Figure 6.4 on page 122, or Figure 6.3 on page 121.

To learn more about multi-file searching using the Find window, see “Searching and Replacing Text in Multiple Files” on page 133.

#### **Multi-File Search Button**

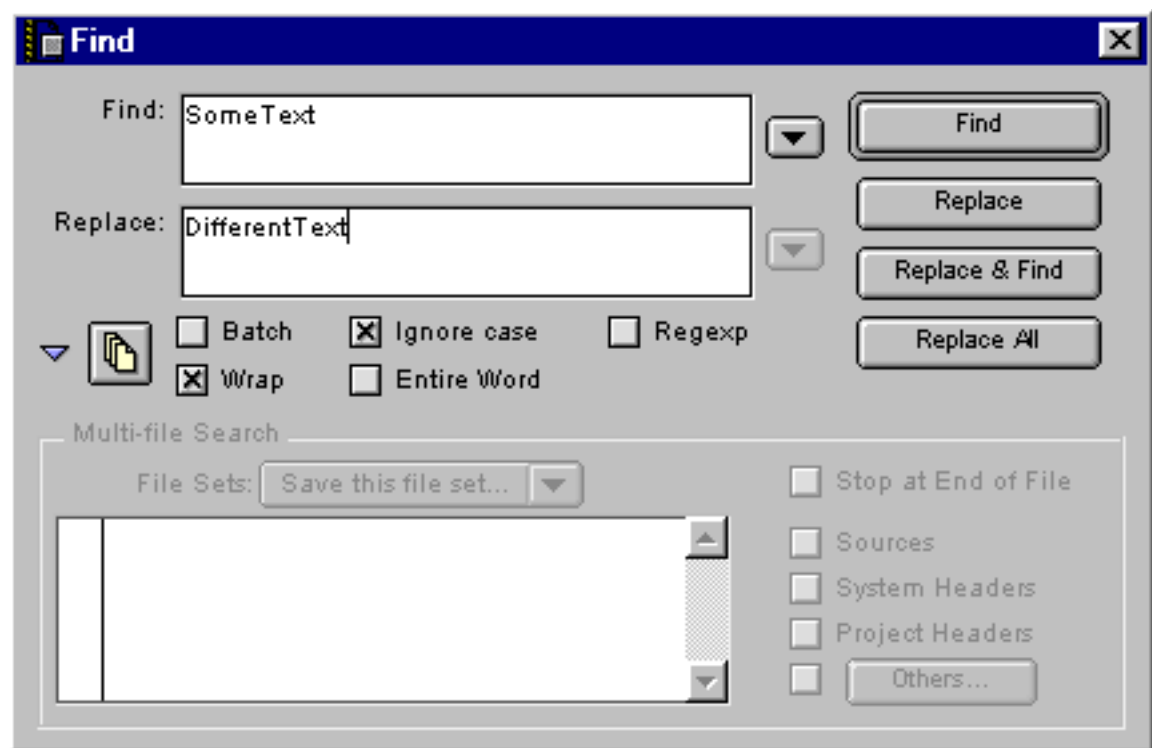
The Multi-File Search Button is shown in Figure 6.3. When depressed, as shown in Figure 6.2 on page 118, the items in the bottom portion of the Find window are enabled for use in searches.



When the Multi-File Search Button is not depressed, as shown in Figure 6.3, the items in the Multi-File Search Section of the Find window are dimmed.

To learn more about Multi-File Search Button, see “Activating Multi-File Search” on page 133.

**Figure 6.3 Multi-File Search Button Not Selected**



### Multi-File Search Section

This section presents a short tour of the Multi-File Search user interface items in the Find window, shown in Figure 6.4. These items are:

- File Sets Pop-Up Menu
- File Sets List
- Stop at End of File Check Box
- Sources Check Box

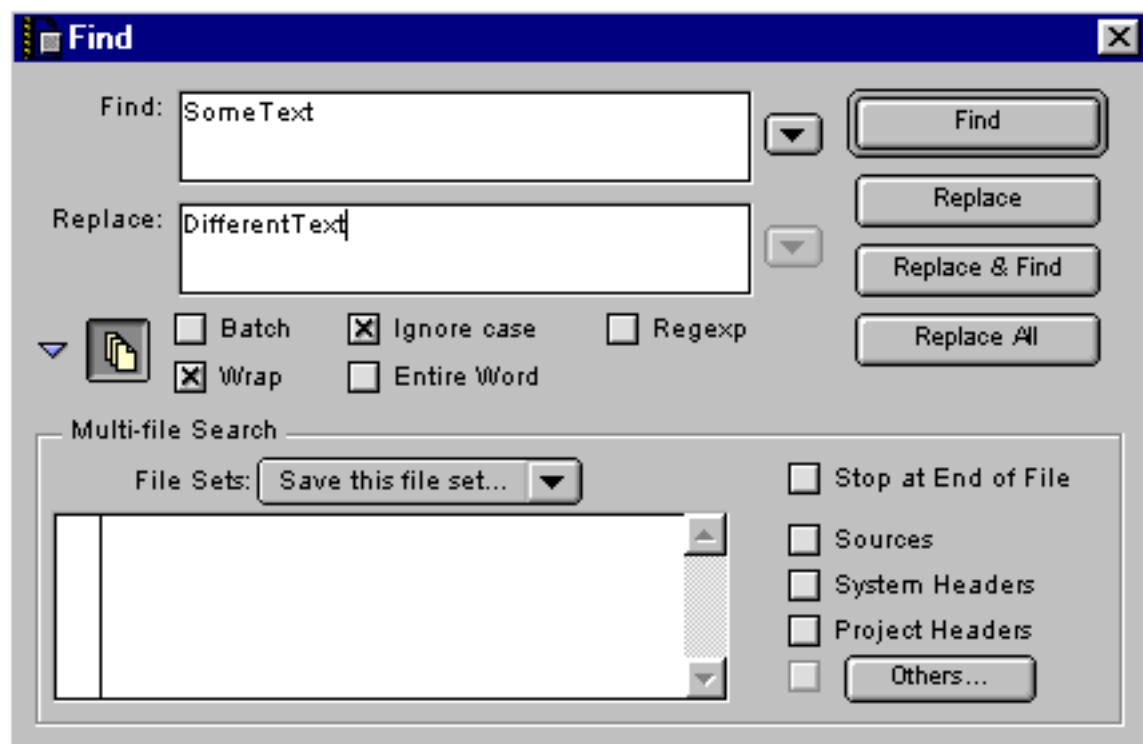
## Searching and Replacing Text

### *Guided Tour of the Find Window*

---

- System Headers Check Box
- Project Headers Check Box
- Others Button

**Figure 6.4 The Find Dialog for a Multiple File Search**



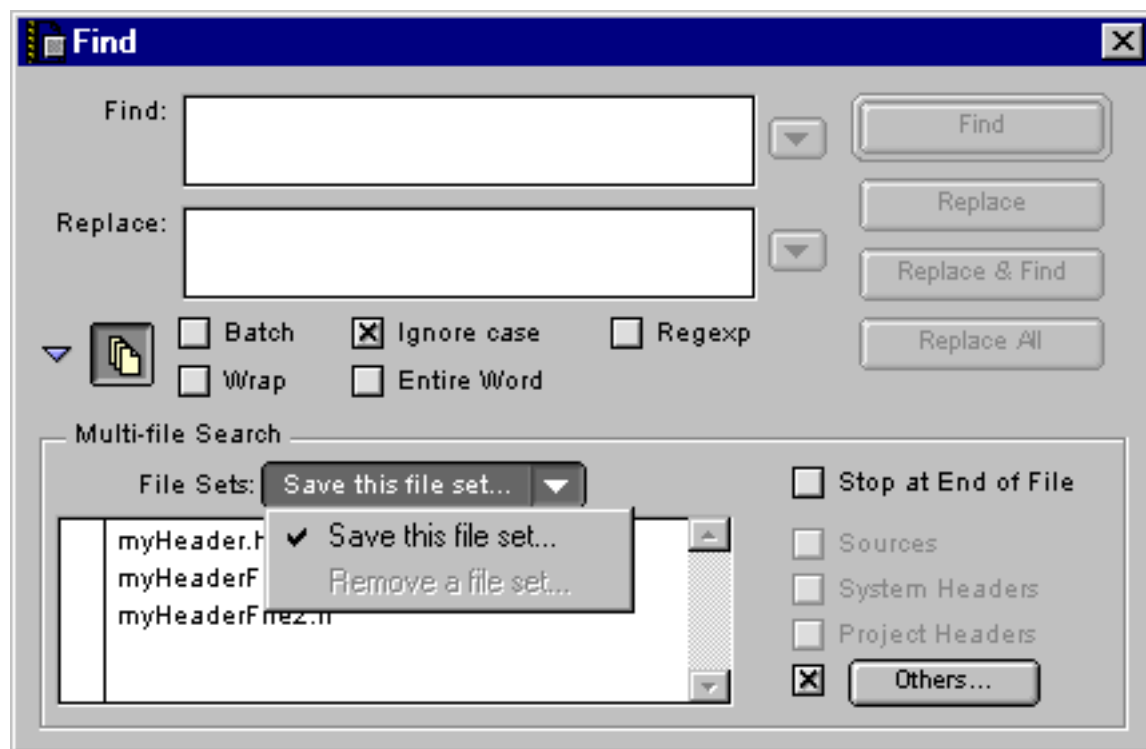
### **File Sets Pop-Up Menu**

The File Sets Pop-up Menu is shown in Figure 6.5. This pop-up menu is used with Multi-file searches. When you select this pop-up, options appear that allow you to select, remove, or save sets of files to search through.

You can build up sets of files that you care about, such as collections of header files, that will be available whenever you want to search through the files for text.

For more information about Multi-file sets of files, see “Choosing Files to be Searched” on page 134.

**Figure 6.5** File Set Pop-up Menu



### File Sets List

The File Sets List is shown in Figure 6.5. This is a list of the files that will be searched in a Multi-file search. You add files to this list by using other controls.

For more information about adding files and removing files in file sets, see “Choosing Files to be Searched” on page 134.

### Stop at End of File Check Box

If you turn off the Stop at End of File Check Box, all the files in the File Sets List are searched as though they are one large file. When the search engine reaches the end of one file, it starts searching the next. When it reaches the end of the last file to search, it beeps.

To search each file individually, turn on the Stop at End of File Check Box. When the search engine reaches the end of a file, it

## Searching and Replacing Text

### *Guided Tour of the Find Window*

---

beeps. You must choose Find in Next File from the Search Menu to continue the search.

For more information about using the Stop at End of File Check Box, see “Controlling Search Range” on page 129.

#### **Sources Check Box**

The Sources Check Box is shown in Figure 6.4 on page 122. This check box adds all the source files from the current project to the File Sets List.

For more information about source files in file sets, and their role in Multi-file searches, see “Adding project source files” on page 135.

#### **System Headers Check Box**

The System Headers Check Box is shown in Figure 6.4 on page 122. This check box adds all the system header files from the current project to the File Sets List.

For more information about system headers in file sets, and their role in Multi-file searches, see “Adding system header files” on page 135.

#### **Project Headers Check Box**

The Project Headers Check Box is shown in Figure 6.4 on page 122. This check box adds all the header files from the current project to the File Sets List.

For more information about project headers in file sets, and their role in Multi-file searches, see “Adding project header files” on page 135.

#### **Others Button**

The Others Button is shown in Figure 6.4 on page 122. This button and its check box allows you to add one or many additional files to the File Sets List.

For more information about adding file to file sets, see “Adding and removing arbitrary files” on page 135.

## Searching for Selected Text

The CodeWarrior Editor provides two ways of searching for text without using the Find window. In both of these methods, you select text in a window, and CodeWarrior finds the text for you without displaying the Find window.

When you search for text using this method, CodeWarrior uses the option settings that you last chose in the Find window. To change these option settings, you must use the Find window.

You should know how to select text in the Editor window before reading this section. If you don’t know how to select text, refer to “Selecting Text” on page 101.

### Finding text in the active Editor window

This method is useful if you want to find additional occurrences of a text string in the same open Editor window that you’re working with.

First, select an instance of the text you want to find. After selecting your text, choose Find Selection from the Search Menu.

CodeWarrior looks for the next occurrence of your text string in the current file only.

To search toward the end of the file for the next occurrence of the text string, click the Find button or choose Find Next from the Search Menu.

To search toward the beginning of the file for the previous occurrence of the text string, choose Find Previous from the Search Menu.

CodeWarrior finds the previous occurrence of the text string and selects it. If the string is not found, then CodeWarrior beeps.

## Searching and Replacing Text

### *Searching and Replacing Text in a Single File*

---

Search for more occurrences of the text string by continuing to use Find, Find Next, or Find Previous on the Search Menu.

#### **Finding text in another window**

This method is useful when your text string is in one file and you want to search for the same text string in another file.

First, select an instance of the text you want to find. After selecting your text, choose Enter 'Find' String from the Search Menu. The Editor enters the text in the Find Text Box of the Find window.

Now make the window you want to search active. Then, choose Find Next or Find Previous from the Search Menu depending on whether you want to search forwards or backwards in the window for the next occurrence of your text string.

CodeWarrior looks for the Find Text Box string in the active Editor window, starting from the location of the text insertion point in that window.

If you want to search toward the end of the file for the next occurrence of the Find Text Box string, click the Find Button or choose Find Next from the Search Menu.

To search toward the beginning of the file for the previous occurrence of the Find string, choose Find Previous from the Search Menu.

Search for more occurrences of the Find Text Box string by continuing to use Find, Find Next, or Find Previous from the Search Menu.

## **Searching and Replacing Text in a Single File**

The Find window allows you to search for text patterns in the Editor window you are working in. When you find the text you are interested in, you can change it or look for another occurrence of it.

This section discusses how to use the Find window to locate specific text you want to replace in the active Editor window.

If you don't yet have a window open, see "Opening an Existing File" on page 70.

If you haven't yet created a file, see "Creating a New File" on page 69.

The topics in this section are:

- Finding Search Text
- Controlling Search Range
- Controlling Search Parameters
- Replacing Found Text
- Replacing Found Text
- Using Batch Searches

### Finding Search Text

To enter text in the Find Text Box, bring up the Find window using the Find command in the Search menu. Type a text string into the Find Text Box on the dialog, or choose a string from the Recent Strings Pop-Up Menu, as shown in Figure 6.6.

**Figure 6.6** Find Text Box and Recent Strings Pop-up Menu



Before searching, you can set other search options that control the range of your search.

## Searching and Replacing Text

### *Searching and Replacing Text in a Single File*

---

The search range defines whether you want to search the entire file or just from the text insertion point in one direction. To set up the range of your search, see “Controlling Search Range” on page 129.

The search parameters define whether you want to search for text regardless of upper or lower case, and whether to search partial words for the text. To set up the parameters of your search, see “Controlling Search Parameters” on page 129.

Before proceeding, make sure that multi-file searching is turned off since you are only interested in searching the active Editor window. To learn about how to determine whether multi-file searching is turned off, refer to “Activating Multi-File Search” on page 133.

Click the Find Button in the Find window to search forward from the text insertion point in the file, or select Find or Find Next from the Search Menu. Select Find Previous from the Search Menu if you want to search backwards from the text insertion point in the file. CodeWarrior now searches for the Find Text Box string in the active Editor window.

To continue searching toward the end of the file for the next occurrence of the Find Text Box string, click the Find Button or choose Find Next from the Search menu.

To continue searching toward the beginning of the file for the previous occurrence of the Find Text Box string, choose Find Previous from the Search Menu.

The Editor finds and selects the Find Text Box string. If the string is not found, the editor beeps.

Search for more occurrences of the Find Text Box string by continuing to use Find, Find Next, or Find Previous.

From this point, you can replace some or all of the text you find with a new text string.

To replace text, see “Replacing Found Text” on page 130.



## Controlling Search Range

The Wrap Check Box option in the Find window controls what happens when you reach the beginning or end of a file in a search.

For example, say that the text insertion point is somewhere in the middle of your text file in the active Editor window. Also, suppose that you have the Wrap Check Box option checked. When you choose Find Previous on the Search Menu, CodeWarrior searches from the end of the file after the search reaches the beginning. In other words, the search “wraps” around the ends of the file. The Find Next command operates in a similar fashion when the end of the file is reached.

If you have the Wrap Check Box option unchecked, and you choose Find Previous on the Search Menu, the search stops when it reaches the beginning of the file.

If you’re searching multiple files with the Wrap Check Box option checked, CodeWarrior searches from the first file in the file list after it reaches the last file.

## Controlling Search Parameters

There are two easily-accessible options for choosing how to match the text you are searching for.

### Ignore Case Check Box

The Ignore Case Check Box is shown in Figure 6.1 on page 116. This check box causes the CodeWarrior search engine to disregard the case (upper or lower) entered into the Find Text Box.

For example, if “Foobar” is in the Find Text Box, then the search engine will also find occurrences like “foobar” or “FOOBAR”, as well as other possible combinations of upper and lower-case text characters.

## Searching and Replacing Text

### *Searching and Replacing Text in a Single File*

---

#### Entire Word Check Box

The Entire Word Check Box is shown in Figure 6.1 on page 116. This check box causes the search engine to ignore occurrences of the text in the Find Text Box that occur within words. For example, if the Find Text Box string is “Word”, CodeWarrior finds only “Word”. If this option is off, it matches text like “Words”, “WordCount”, and “BigWordCount”.

#### Replacing Found Text

When you find an occurrence of text you are interested in, you can either replace one occurrence at a time, or you can replace all occurrences in the entire file.

#### Replace All

To replace text, first enter some text to find in the Find Text Box, then choose the Find operation on the Search Menu, or click the Find Button in the Find window. You can read more about how to find text by referring to “Finding Search Text” on page 127.

Next, enter the replacement text string in the Replace Text Box field of the Find window.

To replace all the occurrences of the Find Text Box string, click the Replace All Button in the Find window, or choose Replace All from the Search Menu.



---

**WARNING!** *Be careful when you use the Replace All command, since Undo is not available for this operation.*

---

#### Selective Replace

To selectively replace text, first enter some text to find, then choose the Find operation on the Search Menu, or click the Find Button in the Find window. You can read more about how to find text by referring to “Finding Search Text” on page 127.

Next, enter the replacement text string in the Replace Text Box field of the Find window.

Type the string in the Replace Text Box field or choose a string from the Recent Strings Pop-Up Menu of the Replace Text Box by clicking the arrow icon just to the right. The Recent Strings Pop-Up Menu (Figure 6.7) contains the last five strings you have used.

**Figure 6.7** Recent Strings pop-up menu



Now choose whether to replace the string you found. For convenience, there are three buttons in the Find window for doing this, the Replace Button, the Replace & Find Button, and the Replace All Button. Each button performs a different operation.

To replace the string and see the results, click the Replace Button in the Find window or choose Replace from the Search Menu. The Editor replaces the text that was found with the Replace Text Box string.

To continue searching forward, choose Find Next from the Search Menu, or click the Find Button in the Find window.

To continue searching backward, press the Shift key as you choose Find Previous from the Search Menu, or press the Shift key and click the Find Button in the Find window.

To replace the string and find the *next* occurrence, choose Replace and Find Next from the Search Menu, or click the Replace & Find Button in the Find window. The Editor replaces the selected text with the Replace Text Box string and finds the next occurrence of the Find Text Box string. If it can't find another occurrence, it beeps.

To replace the Find Text Box string and find the *previous* occurrence, hold down the Shift key as you choose Replace & Find Previous from the Search Menu, or press the Shift key as you click the Replace & Find Button in the Find window. The Editor replaces the selected text with the Find Text Box string and searches for a previous occur-

## Searching and Replacing Text

### *Searching and Replacing Text in a Single File*

---

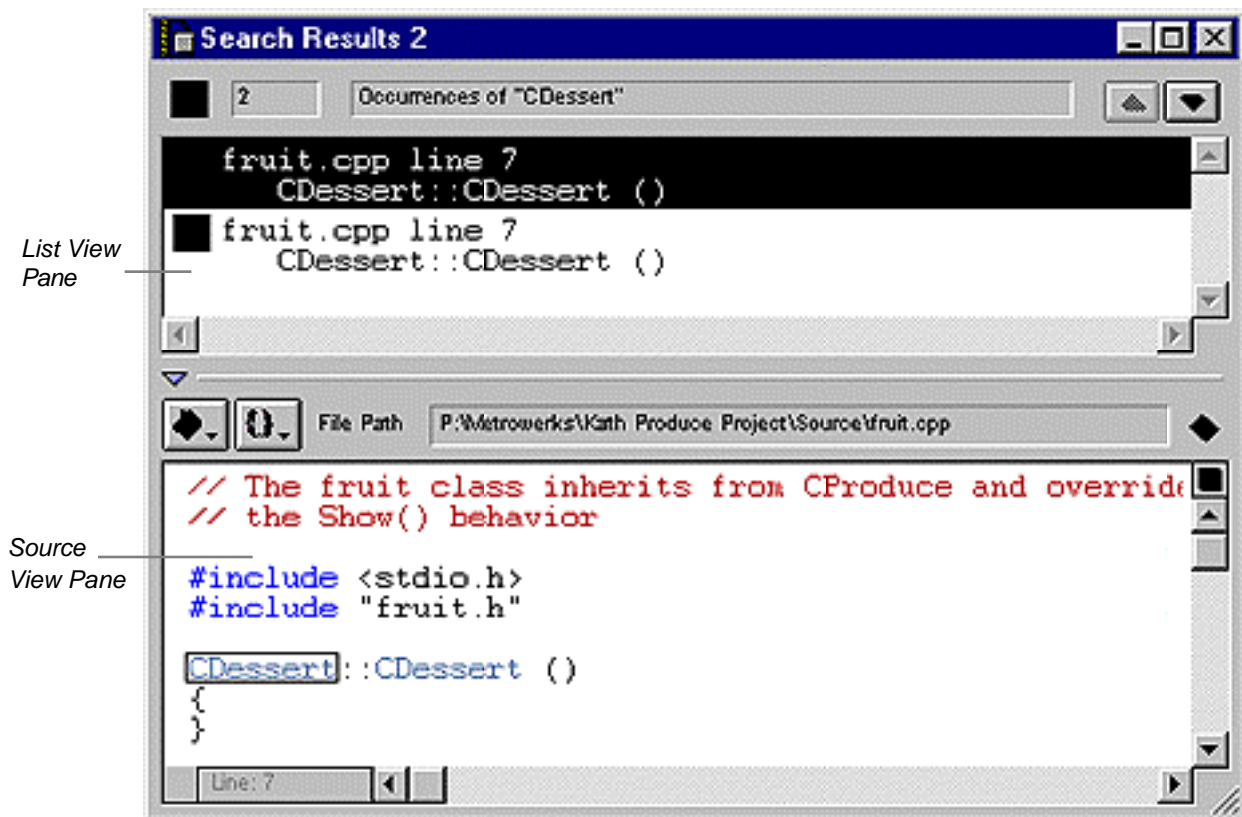
rence of the Find Text Box string. If the search engine can't find another occurrence, it beeps.

## Using Batch Searches

CodeWarrior gives you a way to collect all matching descriptions of your text search in one window for easy reference.

If the Batch Check Box option is checked in the Find window, and the Find button is clicked, CodeWarrior searches for all occurrences of the Find Text Box string and lists them in the Search Results message window, as shown in Figure 6.8.

**Figure 6.8** Batch search results



The Search Results window shown in Figure 6.8 has a List View and a Source View.

To go to a particular occurrence of the Find Text Box string, so that it is shown in the Source View pane of the window, double-click on its entry in the List View.

To learn more about the features of this window, refer to the discussion of the Message Window in “Guided Tour of the Message Window” on page 229.

## Searching and Replacing Text in Multiple Files

CodeWarrior allows you to search multiple files for the occurrence of text strings.

In this section you will learn how to do text searches through multiple files.

Another way to quickly access information and search in multiple files is with the Browser’s Go Back and Go Forward commands on the Search menu. To learn about how to use these commands, refer to “Go Back and Go Forward” on page 169.

The topics in this section are:

- Activating Multi-File Search
- Choosing Files to be Searched
- Saving a File Set
- Removing a File Set
- Controlling Search Range

### Activating Multi-File Search

To configure CodeWarrior to search through multiple files, you need to activate multi-file searching in the Find window.



When the Multi-File Search Button is on, the button appears to be depressed.



When the Multi-File Search Button is off, the button looks three-dimensional.

## Searching and Replacing Text

### *Searching and Replacing Text in Multiple Files*

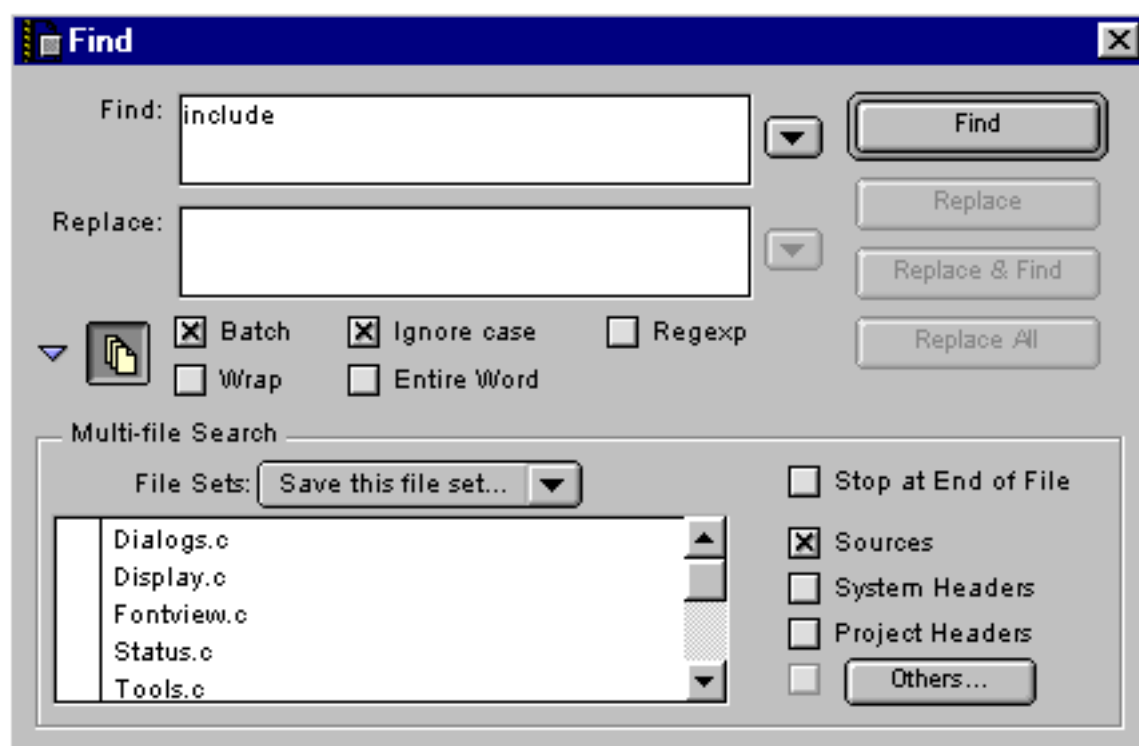
---

Click the Multi-File Search Disclosure Triangle to the left of the Multi-File Search Button, shown in Figure 6.9 on page 134, so that the triangle points down.

CodeWarrior displays the Find dialog with the Multi-File Search Section enabled, as shown in Figure 6.9.

To learn about how to configure the Multi-File Search Section of the Find window, refer to “Choosing Files to be Searched” on page 134, “Saving a File Set” on page 137, “Removing a File Set” on page 138, and “Controlling Search Range” on page 139.

**Figure 6.9 The Find dialog with Multi-File Search options**



## Choosing Files to be Searched

There are several ways to choose files for searching through.

### **Adding project source files**

To add all the source files from the current project, turn on the Sources Check Box. When you turn off the Sources Check Box, CodeWarrior removes all associated files from the file list.

If turning on this option doesn't add any files, update your project's internal list of header files with the Make command. To learn how to do this, refer to "Making a Project" on page 215.

### **Adding project header files**

To add all the project header files from the current project, turn on the Project Headers Check Box. When you turn off the Project Headers Check Box, CodeWarrior removes all associated files from the file list.

If turning on this option doesn't add any files, update your project's internal list of header files with the Make command. To learn how to do this, refer to "Making a Project" on page 215.

### **Adding system header files**

To add all the system header files from the current project, turn on the System Headers Check Box. When you turn off the System Headers Check Box, CodeWarrior removes all associated files from the file list.

If turning on this option doesn't add any files, update your project's internal list of header files with the Make command. To learn how to do this, refer to "Making a Project" on page 215.

### **Adding and removing arbitrary files**

For your multi-file searches, you can add and remove files using the Add File dialog shown in Figure 6.10. This method is particularly useful for adding files not included in your current project.

First, click the Others Button in the Multi-File Search Section of the Find window. Then, choose any files from the dialog's File List.

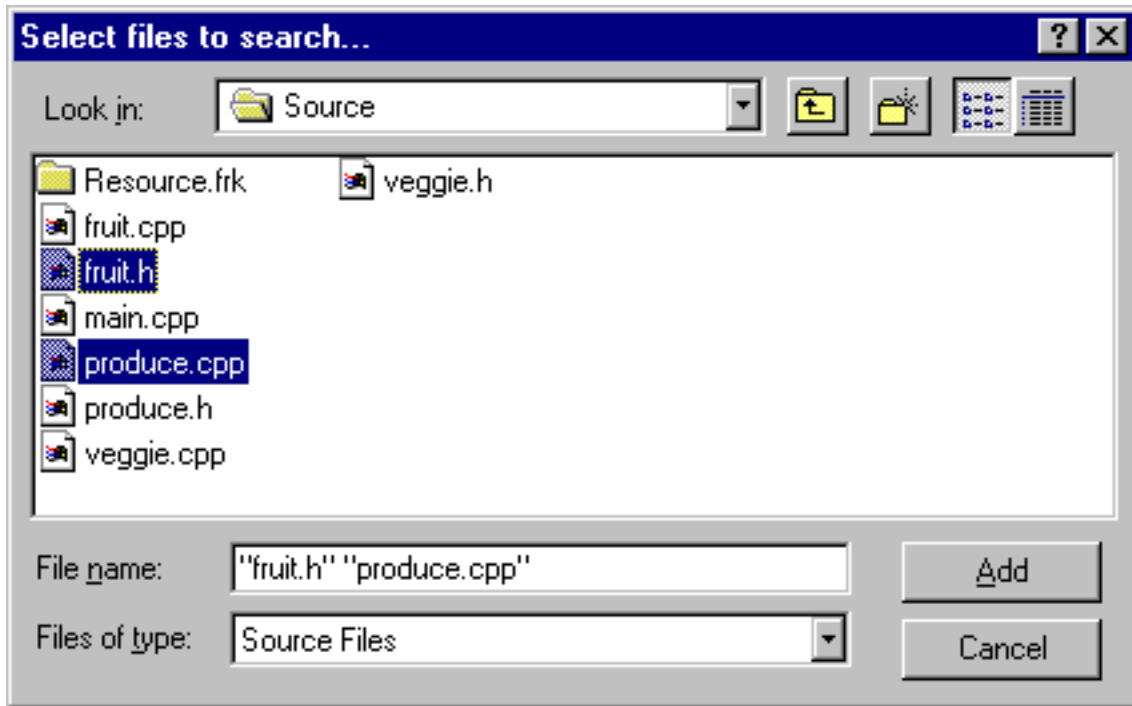
## Searching and Replacing Text

### *Searching and Replacing Text in Multiple Files*

---

Alternatively, you can drag files from the Desktop to the File dialog. Just drag individual or groups of files or complete folders to the Multi-file Search list.

**Figure 6.10** Adding files to a file set with the Add Dialog



The Select files to search dialog shows the files you can choose to add to the file set.

To add a file, select it and click Add. You can add multiple files by pressing the Control key and clicking a file at the same time.

To remove files from the list, click another file or click Cancel to abort the dialog.

When you're finished selecting files, click Add. If you change your mind, click Cancel and the file set is unchanged.

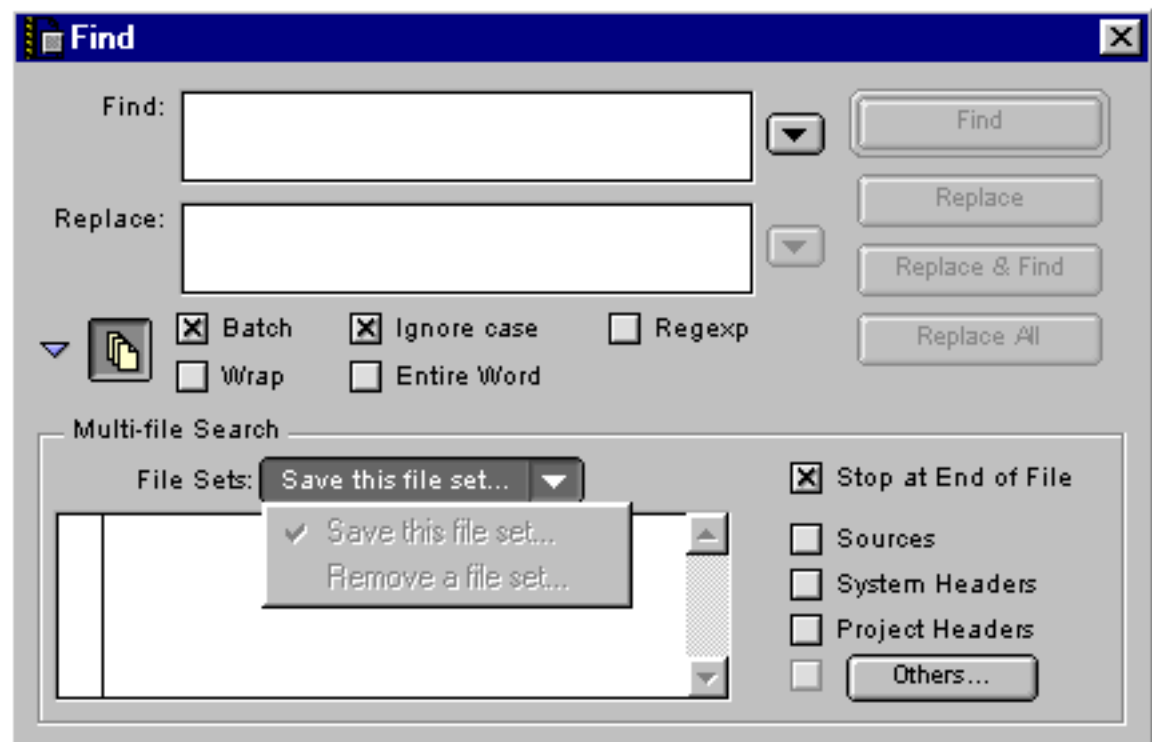
To add more files later, just click the Others Button and the same dialog appears again.



### Choosing a file set

To select a previously-saved file set to include in your search, click on the File Sets Pop-Up Menu and choose a file set from the menu, as shown in Figure 6.11.

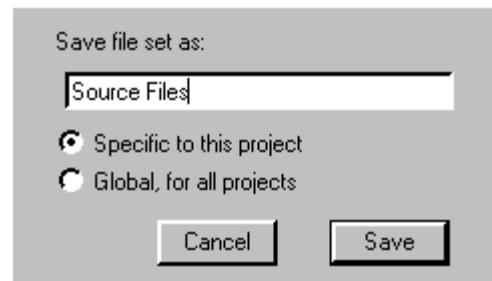
**Figure 6.11** File Sets Pop-up menu



### Saving a File Set

To save a file set for use in future multi-file searches, choose Save this File Set from the File Sets pop-up menu. CodeWarrior displays the Save File Set dialog shown in Figure 6.12.

**Figure 6.12** The Save File Set dialog



Name the file set by entering a name in the Save File Set as text field.

To choose which projects can use this file set, select either the Global or Specific radio buttons in the dialog.

If you plan to use this file set only with the current project, choose Specific to this project. CodeWarrior stores the file set in the project.

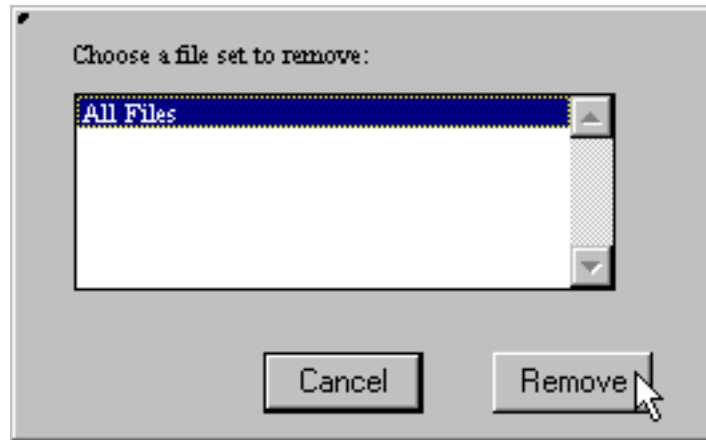
If you think you'll use this file set with other projects, choose Global, for all projects. CodeWarrior stores the file set in its preferences file, so all projects (even existing projects) can use it.

After making your selection and naming the file set, click the Save button. If you change your mind and don't want to save the file set, click Cancel.

## Removing a File Set

To remove a previously-saved file set, choose Remove a file set from the File Sets Pop-Up Menu. Select the set you want to delete in the dialog box that appears, shown in Figure 6.13, and click the Remove button. If you change your mind about removing the file set, click Cancel instead.

**Figure 6.13** Remove File Set Dialog



## Controlling Search Range

CodeWarrior lets you search any number of files for a string. The files can be in the current project or any text file on disk. If you frequently search a particular set of files, you can save that set and restore it later.

You can choose whether to stop searching at the end of each file, or you can choose to search all files without stopping.

To treat all the files in the file set as one large file, turn off the Stop at End of File Check Box. When the editor reaches the end of one file, it starts searching the next file until the selected text is found. When it reaches the end of the last file to search, it beeps. After text is found, you may resume your searching for the next occurrence using the Find, Find Next, or Find Previous menu commands.

To search each file individually, turn on the Stop at End of File Check Box. When the editor reaches the end of a file, it beeps. The arrow to the left of the file set indicates the file the editor is currently searching. You must choose Find in Next File or Find in Previous File from the Search Menu to continue the search. To start the search from a particular file, just select the file and click in the column to its left.

## Searching and Replacing Text

*Using Regular Expressions (grep)*

---

After choosing your option, proceed just as you would if you were searching only one file.

To learn more about text searching, see “Searching for Selected Text” on page 125, or “Searching and Replacing Text in Multiple Files” on page 133.

## Using Regular Expressions (grep)

A regular expression is a text substring that is used as a mask for comparing text in a file. When the regular expression is compared with the text in your file by the search engine, the search engine analyzes whether the text matches the regular expression you have entered.

This section discusses regular expressions CodeWarrior recognizes and how they can be used to find and replace text. CodeWarrior’s regular expressions are similar to the ones that UNIX’s `grep` command uses.



---

**NOTE:** *Make sure the Regexp checkbox is selected in the Find dialog box.*

---

### Matching simple expressions

Most characters match themselves. The only exceptions are called special characters: the asterisk (\*), plus sign (+), backslash (\), period (.), caret (^), square brackets ([ and ]), dollar sign (\$), and ampersand (&). To match a special character, precede it with a backslash, like this `\*`.

For example,

This expression...	matches this...	but not this...
a	a	b
\.\*	.*	dog
100	100	ABCDEFG

## Matching any character

A period (.) matches any character except a newline character.

This expression...	matches this...	but not this...
.art	dart cart tart	art hurt dark

## Repeating expressions

You can repeat expressions with an asterisk or plus sign.

- A regular expression followed by an asterisk (\*) matches zero or more occurrences of the regular expression. If there is any choice, the editor chooses the longest, left-most matching string in a line.
- A regular expression followed by a plus sign (+) matches one or more occurrences of the one-character regular expression. If there is any choice, the editor chooses the longest, left-most matching string in a line. For example:

This expression...	matches this...	but not this...
a+b	ab aaab	b baa
a*b	b ab aaab	baa
.*cat	cat 9393cat the old cat c7sb@#puiercat	dog

## Grouping expressions

If an expression is enclosed in parentheses ( ( and ) ), the editor treats it as one expression and applies any asterisk (\*) or plus (+) to the whole expression. For example

## Searching and Replacing Text

Using Regular Expressions (*grep*)

---

This expression...	matches this...	but not this...
(ab)*c	abc ababababc	aabbbbc abaac
(.a)+b	xab ra5afab	b gaab

### Choosing one character from many

A string of characters enclosed in square brackets ([ ]) matches any one character in that string. If the first character in the brackets is a caret (^), it matches any character *except* those in the string. For example, [abc] matches a, b, or c, but not x, y, or z. However, [^abc] matches x, y, or z, but not a, b, or c.

A minus sign (-) within square brackets indicates a range of consecutive ASCII characters. For example, [0-9] is the same as [0123456789]. The minus sign loses its special meaning if it's the first (after an initial ^, if any) or last character in the string.

If a right square bracket is immediately after a left square bracket, it does not terminate the string but is considered to be one of the characters to match. If any special character, such as backslash (\), asterisk (\*), or plus sign (+), is immediately after the left square bracket, it doesn't have its special meaning and is considered to be one of the characters to match.

This expression...	matches this...	but not this...
[aeiou][0-9]	a6 i3 u2	ex 9a \$6
[^cfl]og	dog bog	cog fog
END[.]	END.	END; END DO ENDIAN

### Matching the beginning or end of a line

You can specify that a regular expression match only the beginning or end of the line.

- If a caret (^) is at the beginning of the entire regular expression, it matches the beginning of a line.
- If a dollar sign (\$) is at the end of the entire regular expression, it matches the end of a line.
- If an entire regular expression is enclosed by a caret and dollar sign (^like this\$), it matches an entire line.

This expression...	matches this...	but not this...
^(the cat).+	the cat runs	see the cat run
.(the cat)\$	watch the cat	the cat eats

### Using the Find string in the Replace string

You can include the contents of the Find string in the Replace string by using an ampersand (&) in the Replace string. For example, suppose the Find string is `[a-z]+123` and the Replace string is `my_&`. If the editor finds `func123`, the editor replaces it with `my_func123`.

To use an ampersand in the Replace without any special meaning, use `\&`. An ampersand has no special meaning in the Find string.

### Remembering sub-expressions

You can remember and recall a part of a regular expression. Enclose the part to remember with parentheses. To recall it use `\n`, where *n* is a digit that specifies which expression in parentheses to recall. Determine *n* by counting occurrences of ( from the left.

For example:

This expression...	matches this...	but not this...
(ab)\1	abab	abc
(ab. )\1	abcab1 ablab1	abcab1 abab

## Searching and Replacing Text

*Using Regular Expressions (grep)*

---

Notice that in the last example `\1` does not re-apply `(ab. )` but matches exactly what `(ab. )` matched.

You can also use `\n` in a Replace string to recall part of an expression from the Find string. For example, suppose the Find string is `([a-z]+)123` and the Replace string is `my_\1`. If the editor finds `func123`, the editor replaces it with `my_func`.





# Configuring IDE Options

---

This chapter discusses the many options available in CodeWarrior's Preferences and Project Settings dialogs.

## Configuring IDE Options Overview

You control many options in CodeWarrior. The CodeWarrior IDE has two dialogs for handling options. One dialog handles global preferences. The other handles project-specific settings. This chapter discusses all the options available in these dialogs.

To set options that are global to all projects that you work with in the CodeWarrior IDE, you choose the Preferences command from the Edit Menu. To set options specific to the Project that you are working with, you choose the Project Settings command from the Edit Menu.

In each case, the many options are organized into a series of panels devoted to a particular topic. For example, one panel controls the font and tab settings in the editor. Another panel controls the target for which you are compiling code.

The topics in this chapter include:

- Option Dialogs Guided Tour
- Choosing Preferences
- Choosing Project Settings

# Option Dialogs Guided Tour

The various options available to you for configuring CodeWarrior settings are found on the Edit Menu, using the Preferences and Project Settings commands.

The topics in this section are:

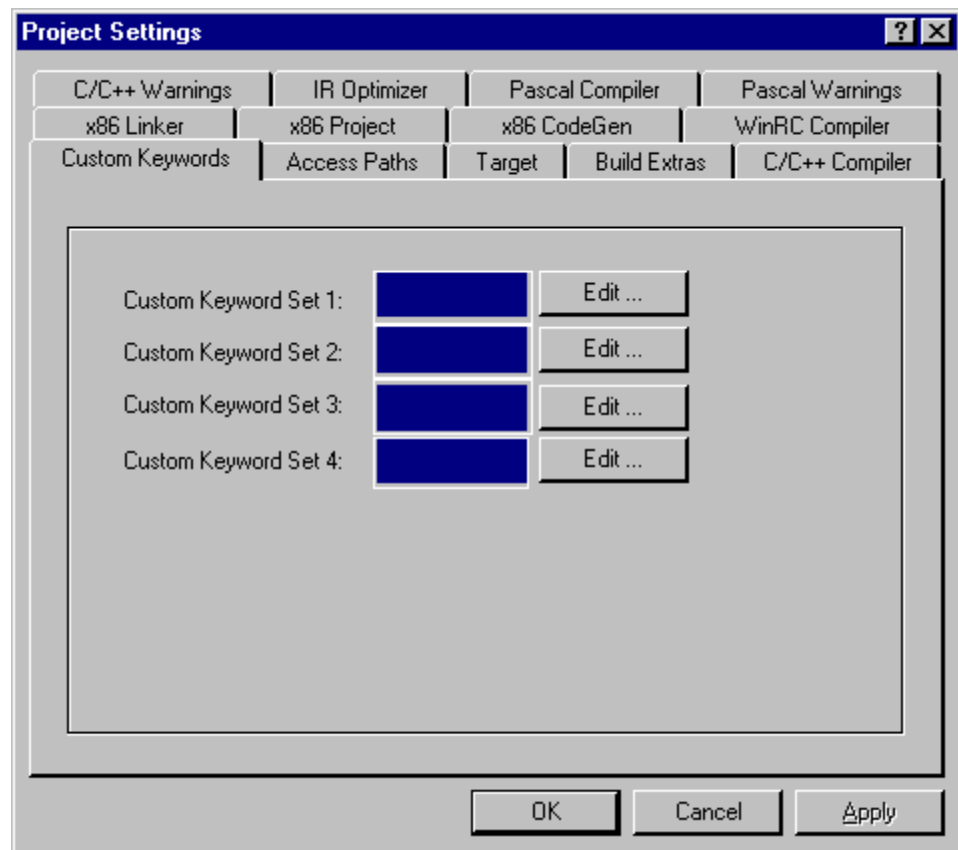
- Options Panels
- Property Sheet Dialog Buttons

## Options Panels

To see an options panel that you configure to set your CodeWarrior Project Settings, choose the Project Settings command from the Edit Menu. You see a dialog such as that shown in Figure 8.1. The actual panels available to you will vary depending upon the CodeWarrior product you are using.

A typical Project Settings panel for Windows development, for example, allows you to configure C/C++ Warnings, IR Optimizer, x86 CodeGen, x86 Linker, x86 Project, Custom Keywords, Access Paths, Target, Build Extras, and C/C++ Compiler options. To select one of these to configure, click on the tab that contains the name of the panel you want to see.

**Figure 8.1** Project Settings Dialog



For example, to view the Target Options Panel, as shown in Figure 8.2, click on the Target tab.

## Configuring IDE Options

### Option Dialogs Guided Tour

---

**Figure 8.2** Target Options Panel

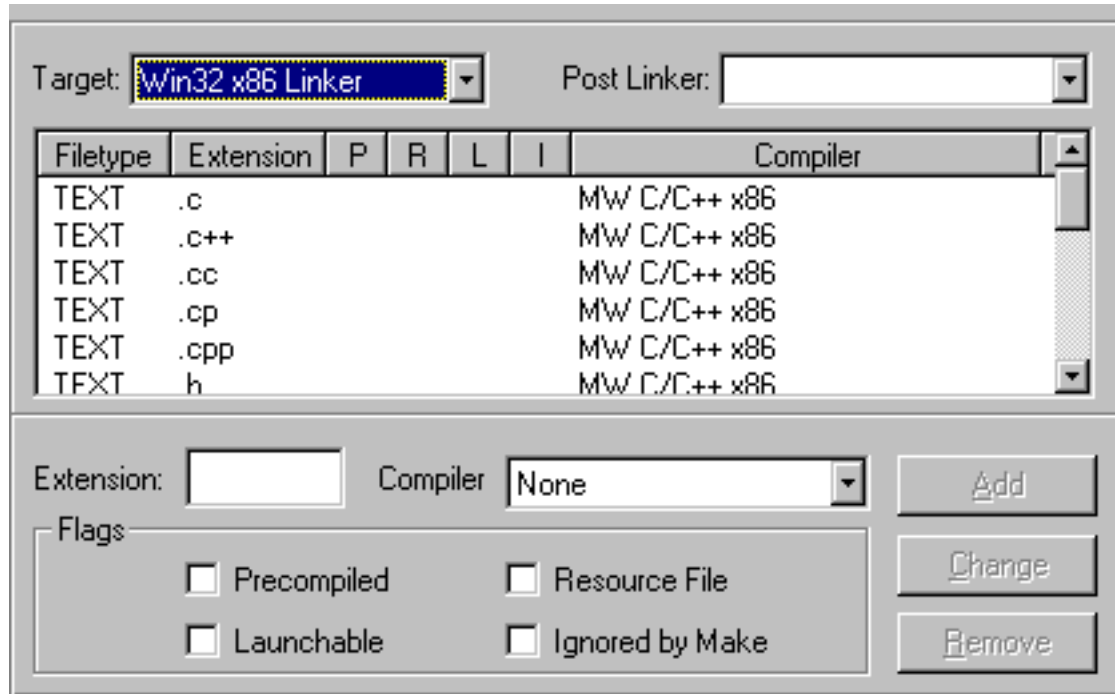
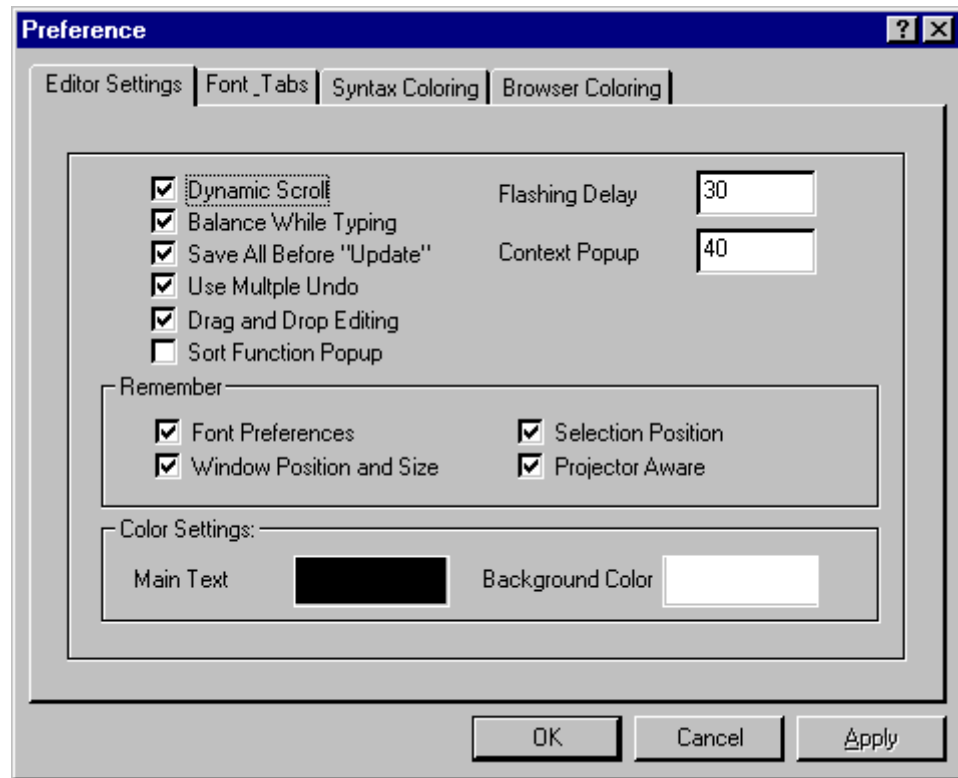


Figure 8.2 shows that when a given Options Panel is selected, the dialog fills with appropriate user interface items that you configure to your liking.

To set global preferences, you use the Preferences dialog, shown in Figure 8.3. This dialog allows you to change the Editor Settings, Fonts and Tabs, Syntax Coloring, and Browser Coloring parameters.

**Figure 8.3** Preferences Panel Dialog



## Property Sheet Dialog Buttons

### Apply button

When you click the Apply button in the Preferences or Project Settings dialogs, you are telling CodeWarrior to save any changes you have made. You may continue to make changes. When you are finished, click OK to close the dialog.

In the DR1 release of CodeWarrior for Windows, the Apply button may not be functional. Check the release notes for the latest information.

## Configuring IDE Options

### *Choosing Preferences*

---

#### **Cancel button**

When you click the Cancel button in the Preferences or Project Settings dialogs, you are telling CodeWarrior to discard any changes you have made in all panels and close the dialog.

#### **OK button**

When you click the OK button in the Preferences or Project Settings dialogs, you are telling CodeWarrior to save any changes you have made and close the dialog.

## Choosing Preferences

To configure any of these panels, click on the appropriate tab to bring the panel forward.

The panels discussed in this section include:

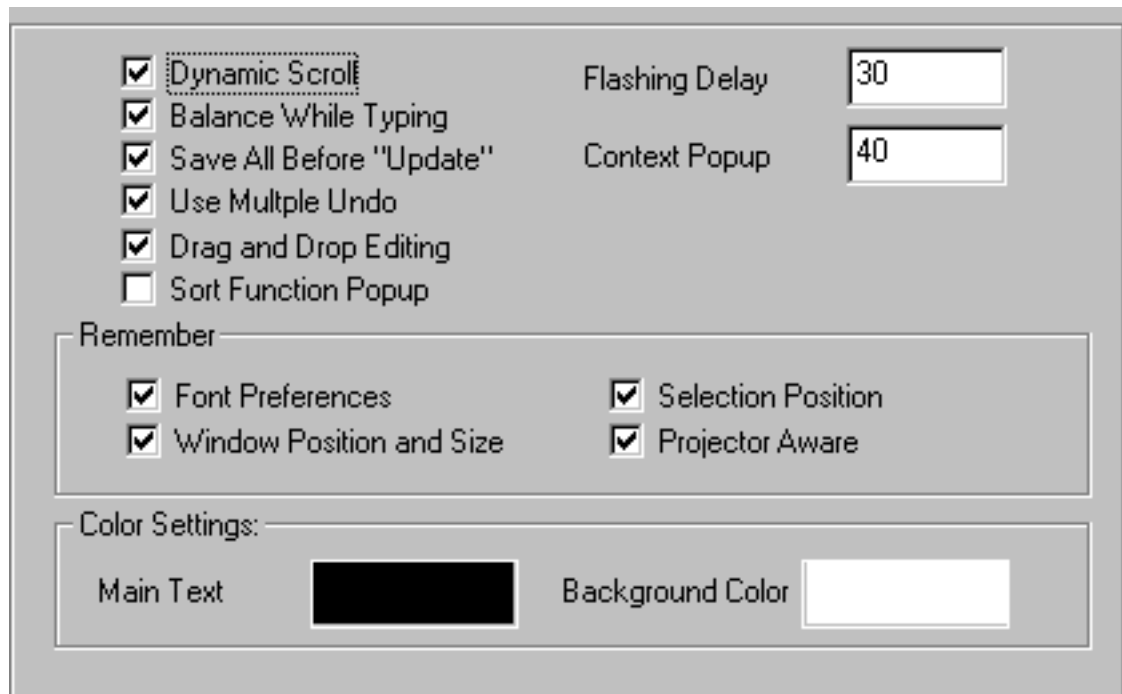
- Editor Settings Panel
- Fonts and Tabs Panel
- Syntax Coloring Panel
- Browser Coloring Panel

### **Editor Settings Panel**

You can configure the CodeWarrior Editor to conform to the way you work. This section tells how to configure the Editor's behavior to make your text editing chores easier. The Editor options panel is shown in Figure 8.4.

This preference panel has three areas of options, Editor Settings, Remember, and Color Settings. Editor Settings (such as Dynamic Scroll and Balance While Typing) controls how the Editor works. The Remember group of preferences determines what state information is saved for editor windows. Color settings controls the main text color (non-syntax) and the background color in the editor and browser windows.

**Figure 8.4 Editor Options Panel**



### **Dynamic Scroll**

If this option is enabled, dragging the scroll box in the scrollbar causes the text to scroll while dragging, instead of just dragging a gray outline of the scrollbox and jumping to the new location when you release the thumb.

### **Balance While Typing**

When the Balance While Typing option is enabled, CodeWarrior checks for balanced parentheses, brackets, and braces as you type.

When you type a right parenthesis, bracket, or brace, the editor searches for the left counterpart. If the editor finds it, the editor highlights it for a specified length of time called the Flashing Delay (scrolling to bring it into view, if necessary) and then resumes (scrolling back to where you were, if necessary). If the editor doesn't find it, it beeps.

By default, the Balance While Typing option is on.

## Configuring IDE Options

### Choosing Preferences

---

You will want to learn about the Flashing Delay parameter to get the most flexibility from this feature.



---

**TIP:** *If you want to check for balanced punctuation without flashing it, set the Flashing Delay to 0.*

---

### Save All Before “Update”

Enable this option if you want all text documents to be saved automatically before a Make, Bring Up To Date, or Run, or Debug command is executed.

### Use Multiple Undo

Enable this option if you want to use the multiple undo feature. When active, the Undo command and Redo command are separate items with separate command keys. If this option is off, the Undo command works as normal. See “Multiple Undo” on page 249 for more information.

### Flashing Delay

The Flashing Delay is the amount of time the CodeWarrior editor displays and highlights an item. It is measured in  $\frac{1}{60}$  of a second. This option is for balancing punctuation. To learn more about balancing punctuation, refer to “Balancing Punctuation” on page 106.

### Context Popup Delay

Context Popup Delay determines how long the mouse button must be held down before the browser’s Context Pop-Up Menu appears. The range of acceptable values is 0-240.



---

**WARNING!** *If you enter 0 (zero) for the time delay, you disable the popup menu entirely.*

---



To learn more information about this feature of the browser, refer to “Context Pop-Up Menu” on page 166 and “Using the Context Pop-Up Menu” on page 169.

### **Drag and Drop Editing**

Enable this option to enable Drag & Drop support in the editor. To learn more about Drag & Drop Editor features, refer to “Moving Text (drag and drop)” on page 102.

### **Sort Function Popup**

Enable this option if you want the Routine Pop-Up Menu in the Editor window to always be sorted by default. To learn more about this feature, refer to “Routine Pop-Up Menu” on page 88.

### **Font Preferences**

You can configure the font information for an individual file if you use this option. Otherwise, all files inherit the default font settings from CodeWarrior.

### **Window Position and Size**

This option saves the window position and size so files open in the same location on the screen each time. This feature requires that your Editor files be writable.

### **Selection Position**

This option tells CodeWarrior to remember what text was scrolled into view, and the location of the insertion point or selection. Turn this option off if you always want the editor to go to the top of the file when it opens a window. This feature requires that your files be writable.

### **Projector Aware**

This option applies to the Mac OS only. Ignore this option.

## Configuring IDE Options

### *Choosing Preferences*

---

#### **Main Text Color**

This option configures the color of any text not colored by the Browser Coloring Panel, Syntax Coloring Panel, or Custom Keywords Panel color sets. Click the color area to change the color.

#### **Background Color**

Click the color area to change the background color of the Editor and Browser windows.

#### **Fonts and Tabs Panel**

To change the settings for Fonts and Tabs you use the preference panel shown in Figure 8.5. If you change the font with this panel for each file in your project when the file is the active Editor window, you can have the font remembered each time the file is opened. If no editor window is open, this preference applies to the CodeWarrior defaults.

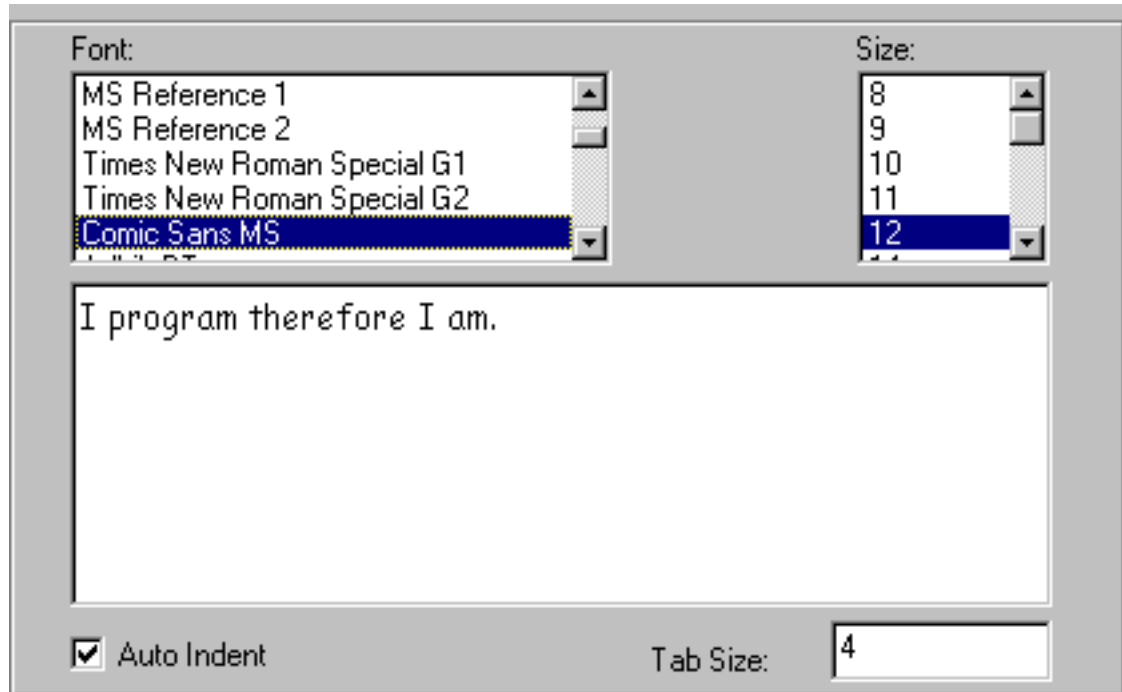
The Font & Tabs panel is shown in Figure 8.5. This preference panel sets the font and tab information for the frontmost editor window.

To change the font and tab settings for a file, make it the active Editor window, then open this options panel and make your changes. As long as you have write permissions on the file, your changes will be remembered.

Choose Auto Indent if you want the editor to automatically indent text on a new line to match up with the text on the previous line.

Tab Size is the number of spaces CodeWarrior inserts to make up a 'tab' using the Tab key.

**Figure 8.5**    **Fonts and Tabs Options Panel**



## Syntax Coloring Panel

The Syntax Coloring options panel provides four Custom Keyword Set settings you can use to make lists of custom keywords to highlight. The list can contain routine names, type names, or anything else you want to have stand out in your Editor windows.

To enable Syntax Coloring, enable the Use Color Syntax checkbox in the top left corner of the options panel, as shown in Figure 8.6.

## Configuring IDE Options

### Choosing Preferences

---

**Figure 8.6** Syntax Coloring Options Panel

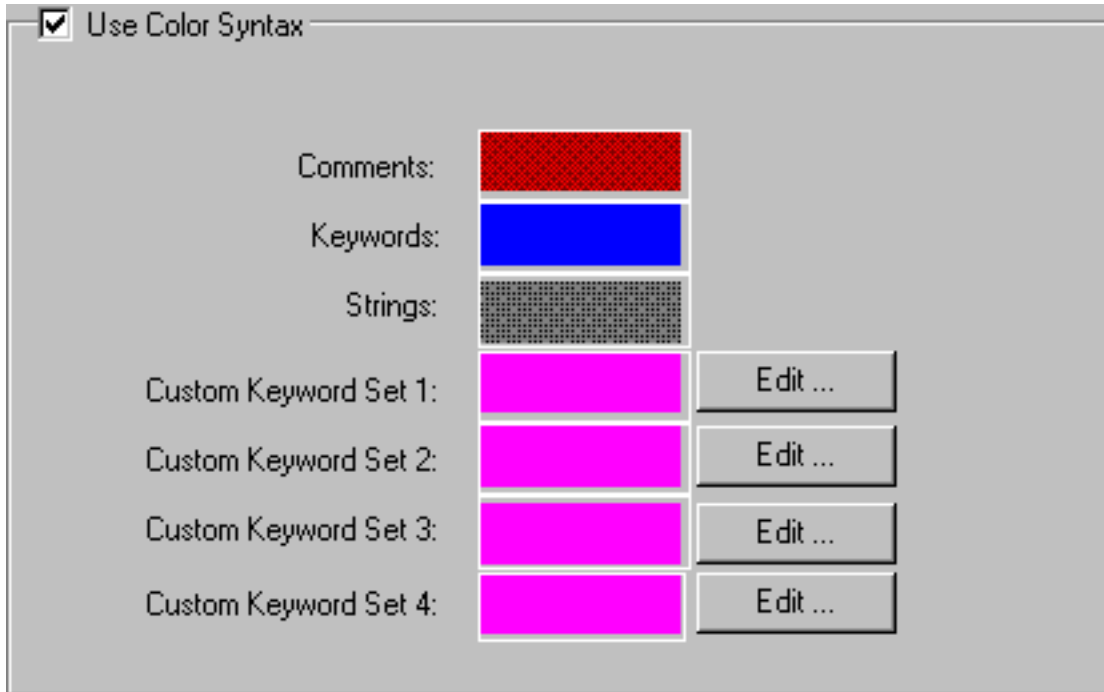


Table 8.1 describes each element of text CodeWarrior highlights in color.

**Table 8.1** Syntax coloring highlights

Element	Description
Main Text	Anything that's not a comment, keyword, or custom keyword, such as literal values, variable names, routine names, and type names. By default, characters are black.
Comments	Code comments. In C or C++, a comment is text enclosed by /* and */ or text from // to the end of the line. In Pascal, a comment is text enclosed by { and }. By default, characters are red.

Element	Description
Keywords	The language's keywords. It does not include any macros, types, or variables that you or the system interface files define. By default, characters are blue.
Custom Keywords	Any keyword listed in the Custom Keyword List. This list is useful for macros, types, and other names that you want to have stand out. By default, characters are blue.

### **Changing syntax highlighting colors**

In the Syntax Coloring preference panel, click on any color you want to change.

CodeWarrior uses different colors for each type of text. To change these colors, click on the color sample beside the name. CodeWarrior displays a dialog (Figure 8.7) you use to select string color. The next time you view a text file, CodeWarrior uses the new color.

## Configuring IDE Options

### Choosing Preferences

---

**Figure 8.7** Select color dialog



### Controlling syntax highlighting within a window

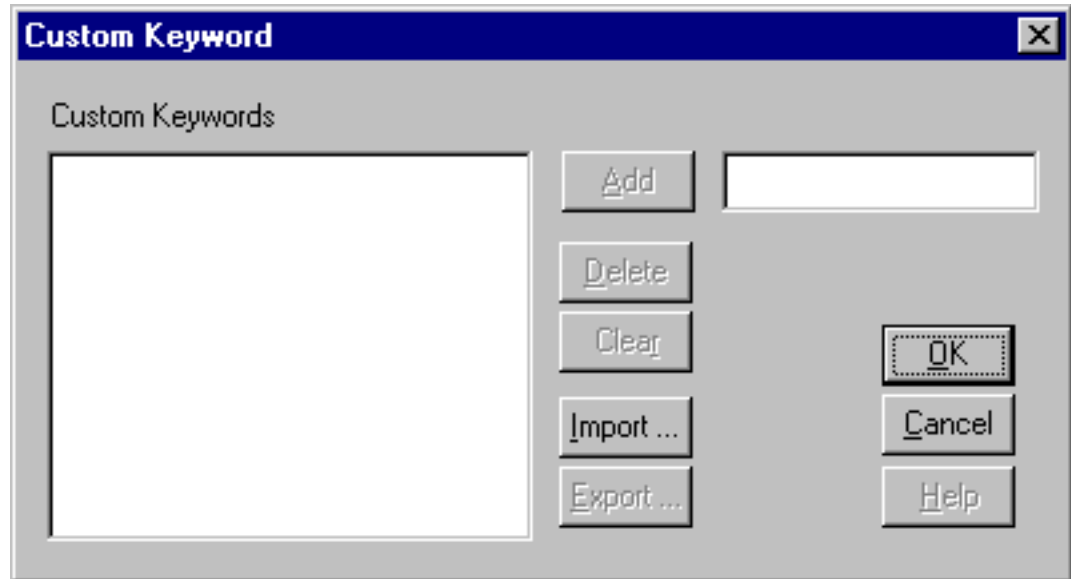
You can turn syntax coloring on or off with the “Adding, Removing, and Selecting a Marker” on page 109.

### Using color for custom keywords

Use the Custom Keyword Sets to choose additional words to display in color. These words can be macros, types, or other names that you want to have stand out. These keywords are global to CodeWarrior and will apply to every project. See also “Custom Keywords Panel” on page 200.

In the options panel, click the Edit button to the right of the Custom Keyword Set you want to modify. CodeWarrior displays the Custom Keywords List dialog, shown in Figure 8.8.

**Figure 8.8** Custom keyword dialog box



Type a keyword in the Add field, then click Add. CodeWarrior adds the keyword to the Custom Keywords List. You can add as many keywords as you want.

To delete a keyword, select the keyword, then click delete. CodeWarrior removes the keyword from the Custom Keywords List.

When you're done, press OK. The dialog disappears. When you next view a source file, all the custom keywords you entered are colored.

### **Importing or exporting custom keywords**

CodeWarrior lets you import or export an entire group of keywords at once, with the Import From File buttons.

## Configuring IDE Options

### *Choosing Preferences*

---

Click the Edit button to the left of the appropriate Custom Keyword Set (shown in Figure 8.6 on page 186).

Choose the Import or Export button. Complete the standard Open file dialog that appears by navigating to your file and opening it.

CodeWarrior adds or subtracts the custom keywords from that file to or from your Custom Keyword Set.

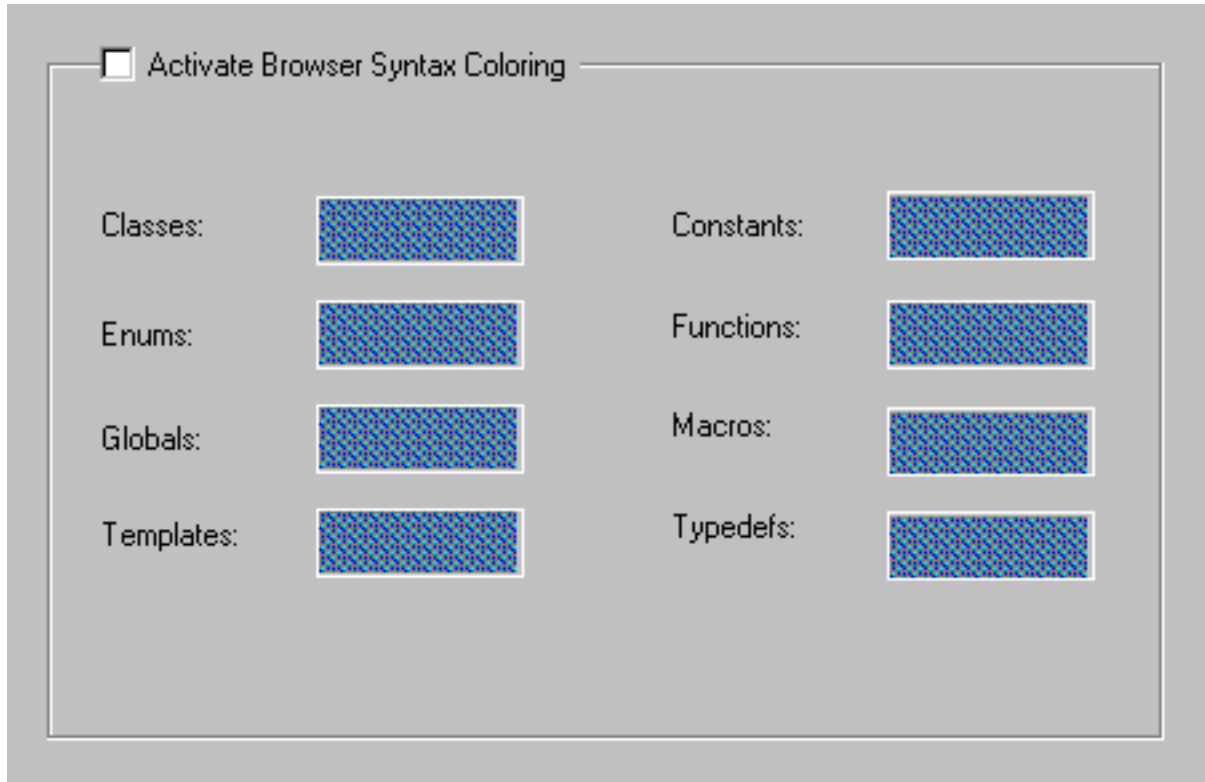
When you're done, click OK. The dialog disappears. When you next view a source file, all the custom keywords you entered will be highlighted as you designated.

## Browser Coloring Panel

The Browser Coloring preferences are shown in Figure 8.9. The Browser can export its lists of symbols and their types to the editor. This enables the editor to use different colors for various types of symbols. Choose Activate Browser Syntax Coloring to use this feature. When active, the color choice for each symbol type will be displayed in the Editor window and the Browser window. Click on a color area to bring up the dialog (shown in Figure 8.7 on page 188) to change the color.



**Figure 8.9** Browser highlighting options



## Choosing Project Settings

You can change many different settings to configure CodeWarrior to build your project the way you want. This section discusses the changes you can make to Project Settings specific to your project.

The topics in this section include:

- Target Panel
- Access Paths Panel
- Build Extras
- Custom Keywords Panel
- C/C++ Compiler Panel
- C/C++ Warnings Panel

## Configuring IDE Options

### Choosing Project Settings

---

- Pascal Compiler
- Pascal Warnings
- x86 Project Panel
- IR Optimizer Panel
- x86 CodeGen Panel
- x86 Linker Panel
- IR Optimizer Panel
- WinRC Compiler

## Target Panel

The Target options panel, shown in Figure 8.10, is used to associate a file name extension such as .c or .p with a plugin compiler. This tells CodeWarrior which compiler to use when a file with a certain name is encountered.

When you change the target or application type, you must change the libraries contained in the project. Creating a new project does this for you automatically. Choosing a new value for a Target does not change these files for you. For this reason, you should be careful when changing values in this section.

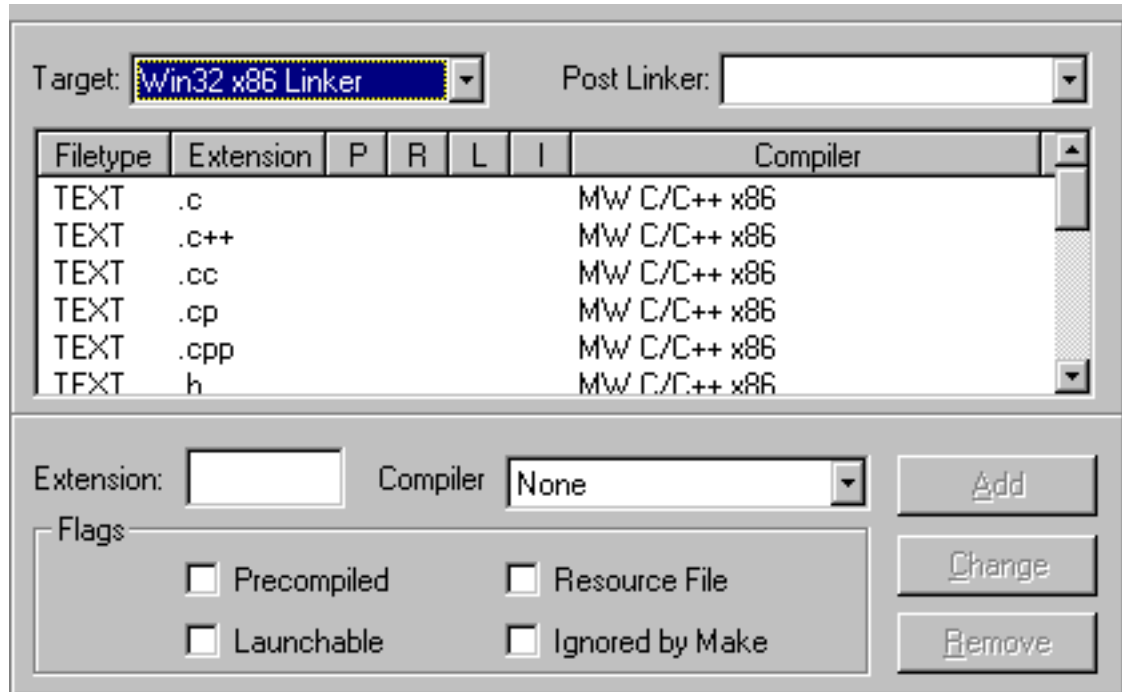


---

**TIP:** *To change the target or project type, it's best to create a new project and add the source code files from the old project.*

---

**Figure 8.10 Target Options Panel**



### Target

The choices for target vary depending on the platform you are compiling on. Available choices appear in this pop-up menu.

To learn about a particular target, refer to the individual Targeting manual for that target. For example, for Win32 platforms, refer to the *Targeting Win32* manual.

### Post Linker

Some targets have post linkers that perform additional work (such as a data format conversion) on the final executable. You can learn about such post linker options (if any) in the individual Targeting manual.

### File Name List

The File Name List contains a Filetype, associated extension, and compiler choice for each \*. file name extension. This list tells

## Configuring IDE Options

### Choosing Project Settings

---

CodeWarrior which compiler to invoke when a given file name is encountered.

To add a new extension to this list, choose an existing entry in the list, edit the Extension and Compiler fields, and click the Add button.



---

**NOTE:** *To add the new file kind to all new empty projects, make sure you close your Project window (if it is open).*

---

#### Extension

This field allows you to enter a file name extension such as .c or .h for a Filetype you are working with in the File Name List.

#### Compiler

This field allows you to choose a compiler for a Filetype you are working with in the File Name List.

#### Precompiled

Compile these files before other files. This is useful if these files create documents that other source files or compilers use. For example, a precompiled header source file (a .pch file) must be compiled before the source files that use it. Also, this option lets you create a compiler that translates a file into a C source code file and then compile the C file.

#### Launchable

Open the source code file with the application that created it when you double-click it in a project window.

#### Resource File

Include the resources from these files in your finished product.

### Ignored by Make

Ignore these files when compiling or linking the project. This is useful if the files contain comments or documentation that you want to include with your project.

### Access Paths Panel

If you need to define additional access paths for CodeWarrior to search while compiling and linking your project, you would use the Access Paths options panel, shown in Figure 8.11.

You can use drag and drop to add paths to the Access Paths options panel. Just drag the folder on to the path list. You can then remove paths by dragging a path to the Recycle Bin.

If a folder icon appears beside the name of a folder in either the User Include Path Pane, or the System Include Path Pane, CodeWarrior performs a recursive search on the path. That is, CodeWarrior searches that folder and all the folders within it.

By clicking the folder icon to the left of any path in the User Include Path or System Include Path panes, you can disable recursive searching of all subdirectories below that path. If the folder is visible, recursive search is turned on. If the folder is not visible, all subdirectories of that path will not be searched by the compiler.



---

**TIP:** *If you turn off recursive searching of paths, and add each specific path to every directory to either the System Include Path or User Include Path panes, you will speed compilation of your project.*

---

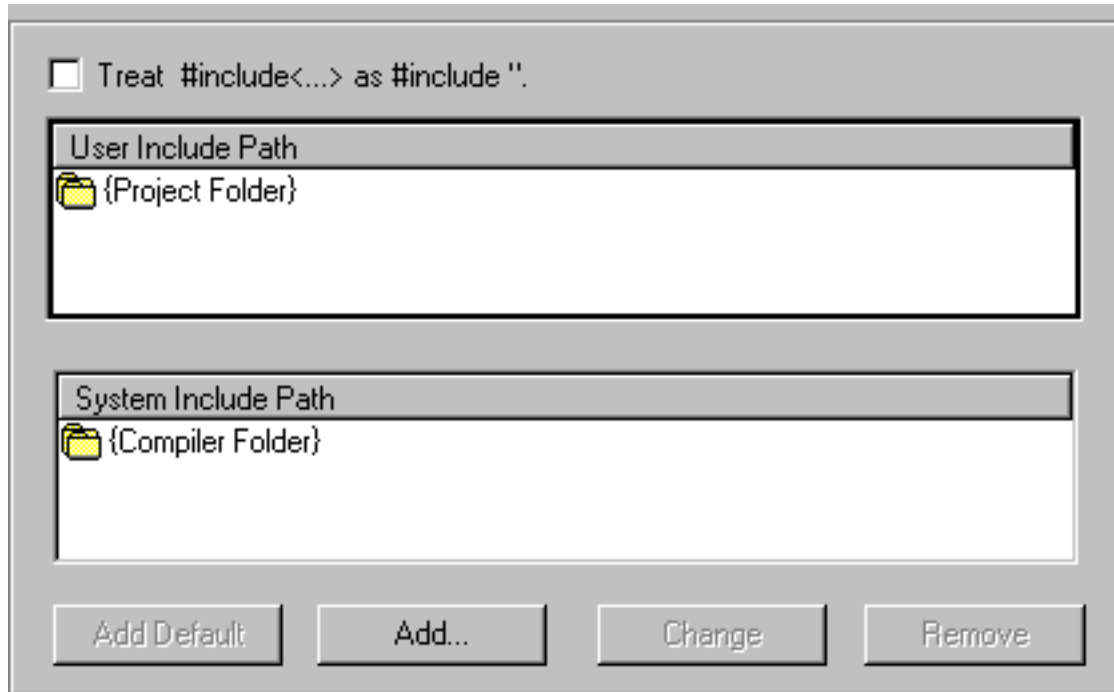
If your project's files or libraries are not in either of the default access paths, CodeWarrior will not find them when compiling, linking, or running your project. You must add their access path to tell the environment where to look.

## Configuring IDE Options

### Choosing Project Settings

---

**Figure 8.11** Access Paths Options Panel



#### **Treat #include <...> as #include “...”**

To search for system header files in the same way as user header files, turn on this option.

#### **User Include Path Pane**

In Pascal, these access paths are searched first. In C, an #include “...” statement searches these access paths. By default, it contains {Project Folder}, which is the folder that contains the open project.

#### **System Include Path Pane**

In Pascal, these access paths are searched after those in the User Include Path Pane. In C, an #include < . . . > statement searches these access paths. By default, it contains {Compiler Folder}, which is the folder that contains the CodeWarrior product.

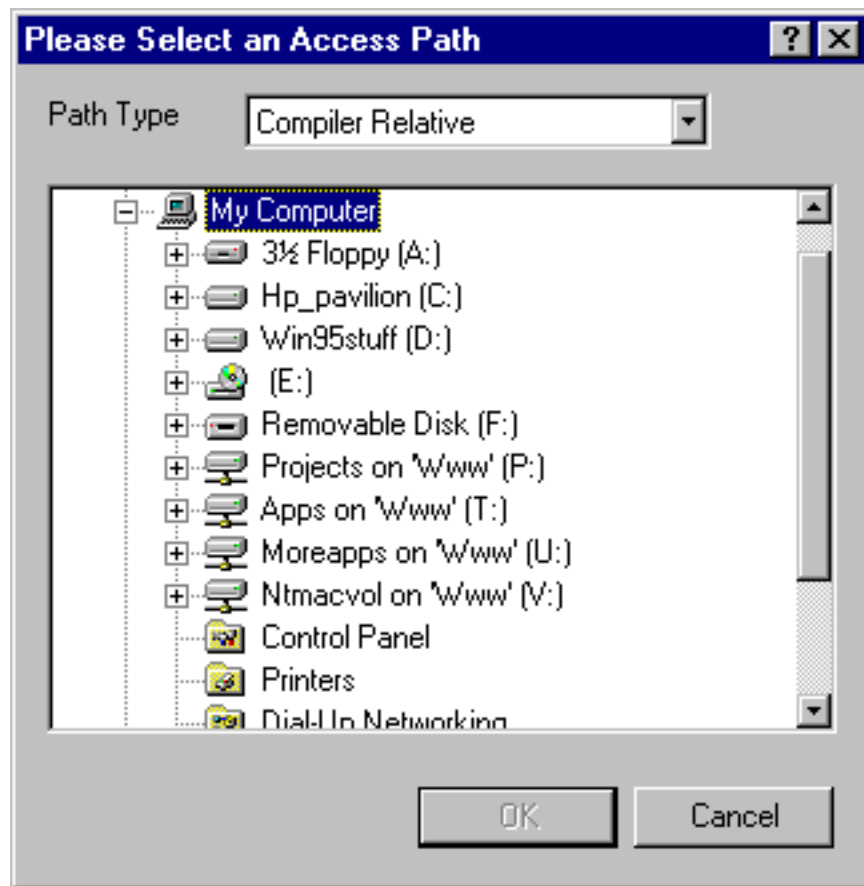
### Add Default

CodeWarrior lets you add the default path for the User Include Path Pane or System Include Path Pane after you have deleted the default path. To add the default path to the access path pane you are working with, click the Add Default button. CodeWarrior adds the default path back into the relevant path pane.

### Add

To add a new access path, first select the System Include Path Pane or the User Include Path Pane. Then, click the Add button. The dialog shown in Figure 8.12 appears.

**Figure 8.12** Access Path Selection Dialog



## Configuring IDE Options

### *Choosing Project Settings*

---

Use the standard Windows tree controls to navigate to the folder you want to add to the Access Paths.

You can choose how CodeWarrior will store the access path with the four radio buttons above the OK button. The four buttons are Absolute Path, Project Relative, Compiler Relative, and System Relative.

Absolute Path means that all the folders that lead to your project folder, including the hard disk, will be incorporated in the Access Path. You need to update an absolute access path if you move a project to another system, or rename the hard drive, or rename the folder that contains the project.

Project Relative Path means that the folders that lead to your folder, in relation to the folder that contains the current project, will be incorporated in the Access Path. You do not need to update relative access paths if you move a project, as long as the hierarchy of the relative path is the same. However, you cannot create a relative path to a folder on different hard disk than where your project file resides.

Compiler Relative means that the folder leading to your folder, in relation to the folder that contains CodeWarrior, will be incorporated in the Access Path. You do not need to update relative access paths if you move a project, as long as the hierarchy of the relative path is the same. However, you cannot create a relative path to a folder on different hard disk than where your project file resides.

System Relative means that folders leading to the Windows folder will be incorporated in the Access Path. You do not need to update relative access paths if you move a project as long as the hierarchy of the relative path is the same.

Click the OK button, and CodeWarrior adds the Access Path to the list.

### **Change**

To change an access path, first select a path in the System Include Path Pane or the User Include Path Pane. Then, click the Change button. The dialog shown in Figure 8.12 on page 197 appears. Use



this dialog to navigate to the location of the access path you want to change to.

To learn more information about the options in the dialog, refer to “Add” on page 197.

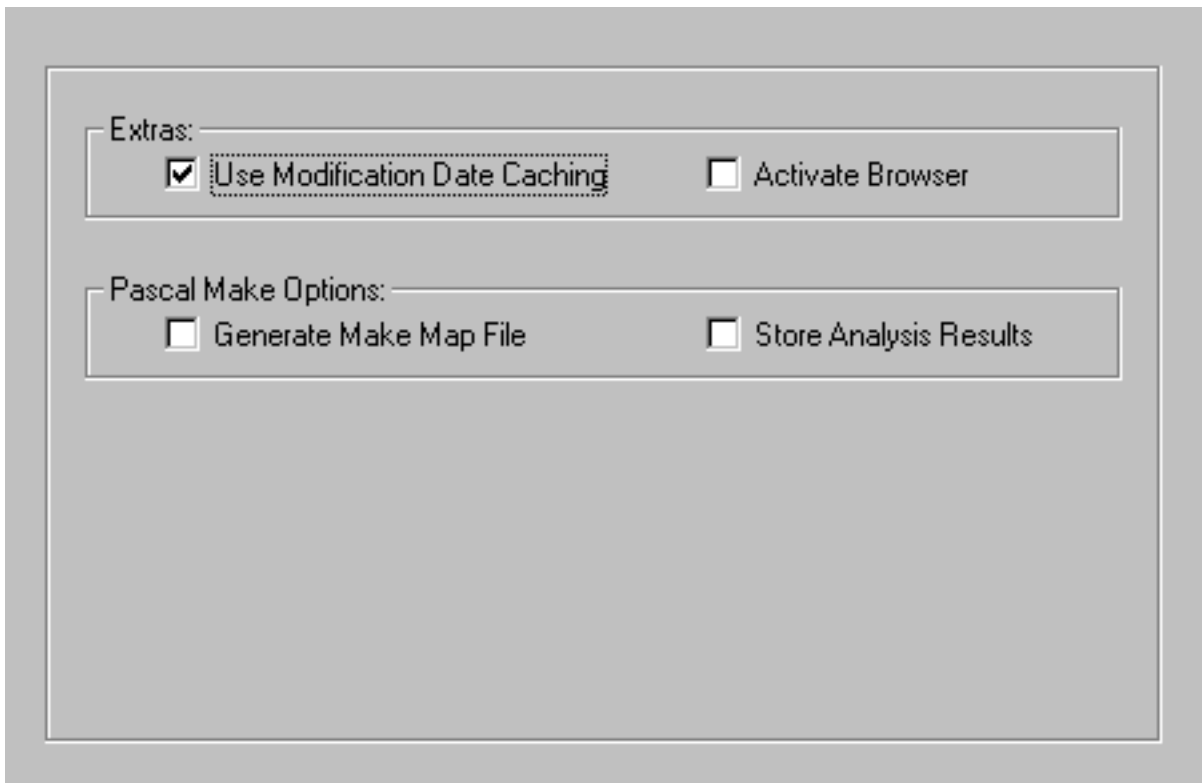
### **Remove**

To remove an access path, first select the System Include Path Pane or the User Include Path Pane. Then, click the Remove button. The path is removed.

### **Build Extras**

The Build Extras panel contains various options that affect how a project builds. These options are shown in Figure 8.13.

**Figure 8.13 Build Extras Options Panel**



## Configuring IDE Options

### *Choosing Project Settings*

---

#### **Use Modification Date Caching**

Before making a project, CodeWarrior checks the files' modification dates to see if you've changed them outside CodeWarrior. If you edit files with CodeWarrior only, turn on the Use Modification Date Caching option to shorten compilation time.

#### **Activate Browser**

If you activate the Browser for a project, you must rebuild your project (see "Making a Project" on page 215) so the compiler can generate the necessary information for each file.

For more information on browser settings and options, see "Browser Overview" on page 145.

#### **Generate Make Map File**

This option causes the compiler to generate a Pascal Make Map file.

#### **Store Analysis Results**

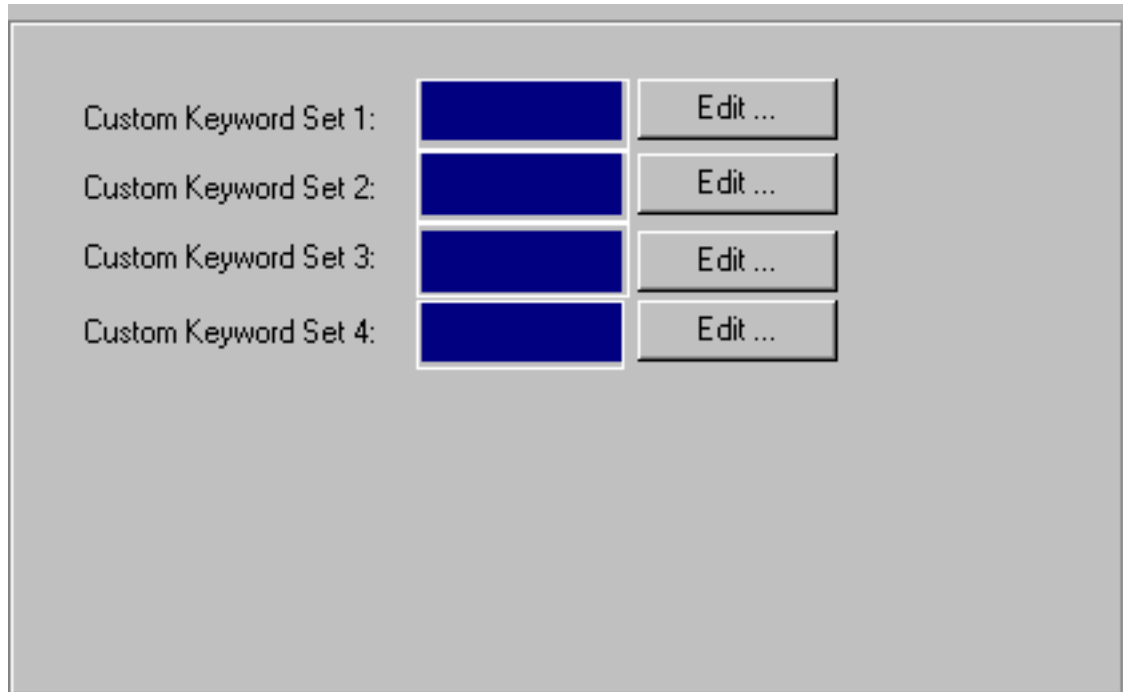
This option causes the compiler to store Pascal dependency information between compiles.

### **Custom Keywords Panel**

The Custom Keywords options panel, shown in Figure 8.14, allows you to define your own keyword sets that have certain colors associated with them when they appear in your Editor files. These keywords are project-specific, not global to CodeWarrior. See also "Using color for custom keywords" on page 188.

To change a keyword set, click the Edit button and make the appropriate entries in the dialog that is presented.

**Figure 8.14** Custom Keywords Options Panel



## **C/C++ Compiler Panel**

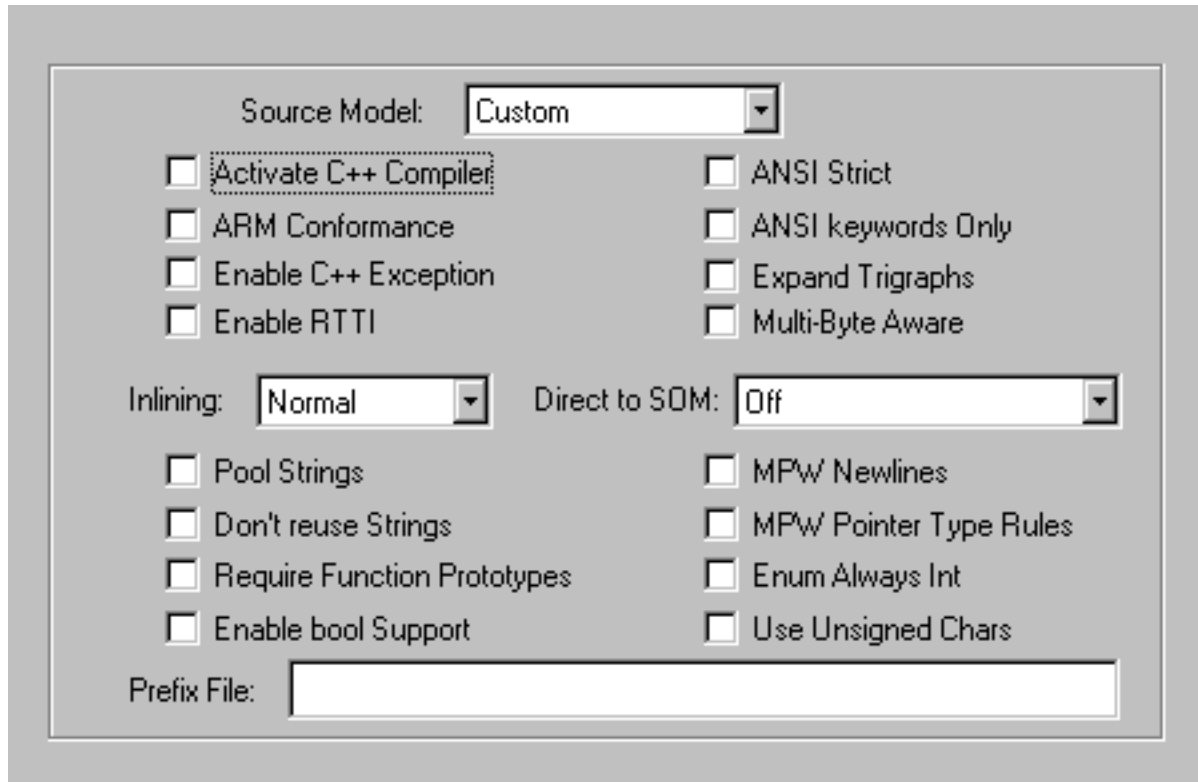
There are several settings you can configure to tailor the C/C++ compilation of your project's source code files, as shown in Figure 8.15. You may also refer to the *C, C++ and Assembly Language Reference* manual for detailed information on these options.

## Configuring IDE Options

### Choosing Project Settings

---

**Figure 8.15 C/C++ Compiler Options Panel**

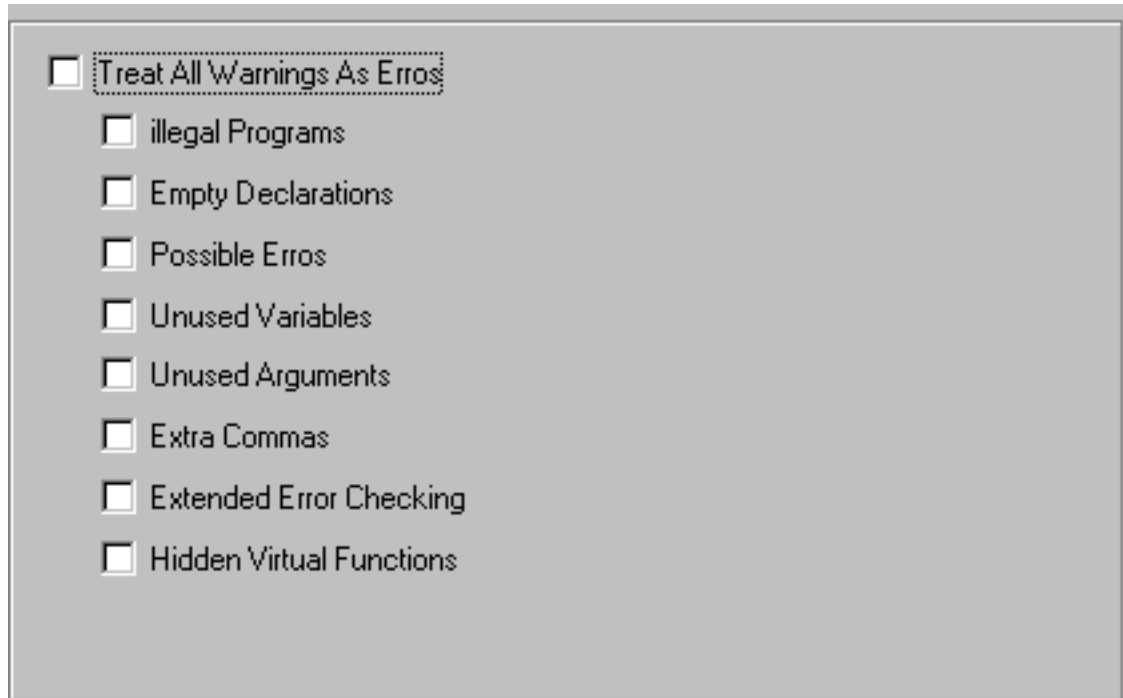


## C/C++ Warnings Panel

The C/C++ Warnings options panel, shown in Figure 8.16, allows you to configure the warning messages that are emitted during compilation of your C and C++ code.

To learn more information about how to configure these options, refer to the *C, C++, and Assembly Language Reference* documentation.

**Figure 8.16** C/C++ Warnings Options Panel



## Pascal Compiler

The Pascal Compiler options panel, shown in Figure 8.17, allows you to configure your compilation options for the Pascal compiler.

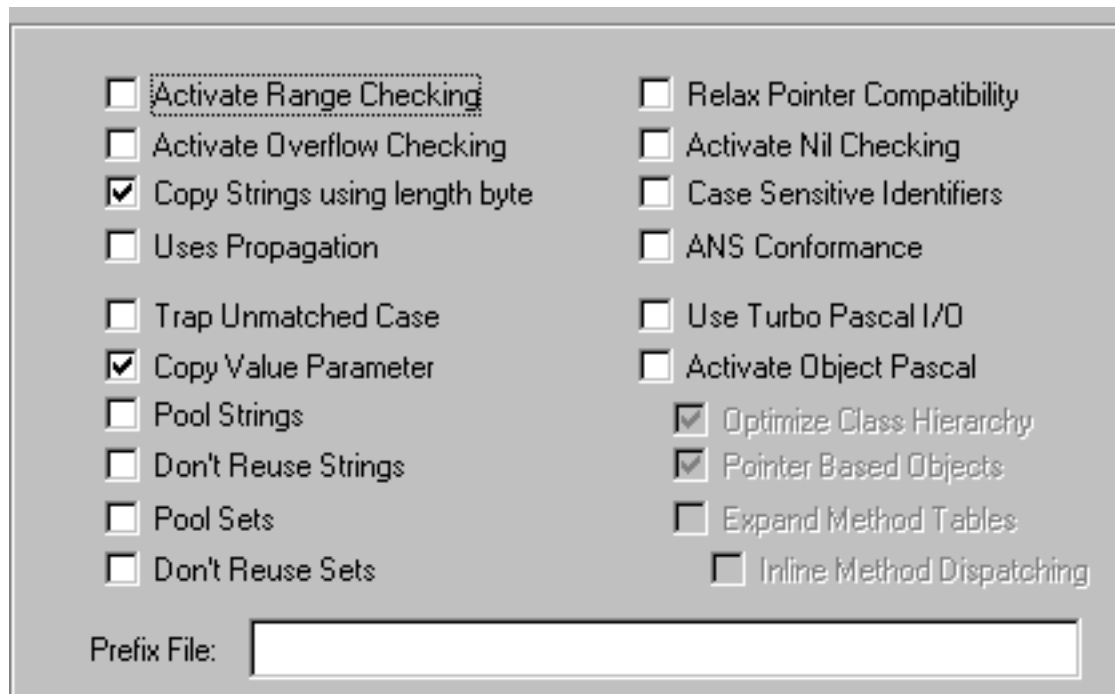
For a detailed explanation of how to configure these options, refer to the *Pascal Language Reference* manual on the CodeWarrior CD.

## Configuring IDE Options

### Choosing Project Settings

---

**Figure 8.17** Pascal Compiler Options Panel



## Pascal Warnings

The Pascal Warnings options panel, shown in Figure 8.18, allows you to configure your options for warning messages in the Pascal compiler.

For a detailed explanation of how to configure these options, refer to the *Pascal Language Reference* manual on the CodeWarrior CD.

**Figure 8.18 Pascal Warnings Options Panel**



## **x86 Project Panel**

The x86 Project options panel, shown in Figure 8.19, allows you to configure your project binary options depending on whether you are creating an application, library, or shared library (DLL).

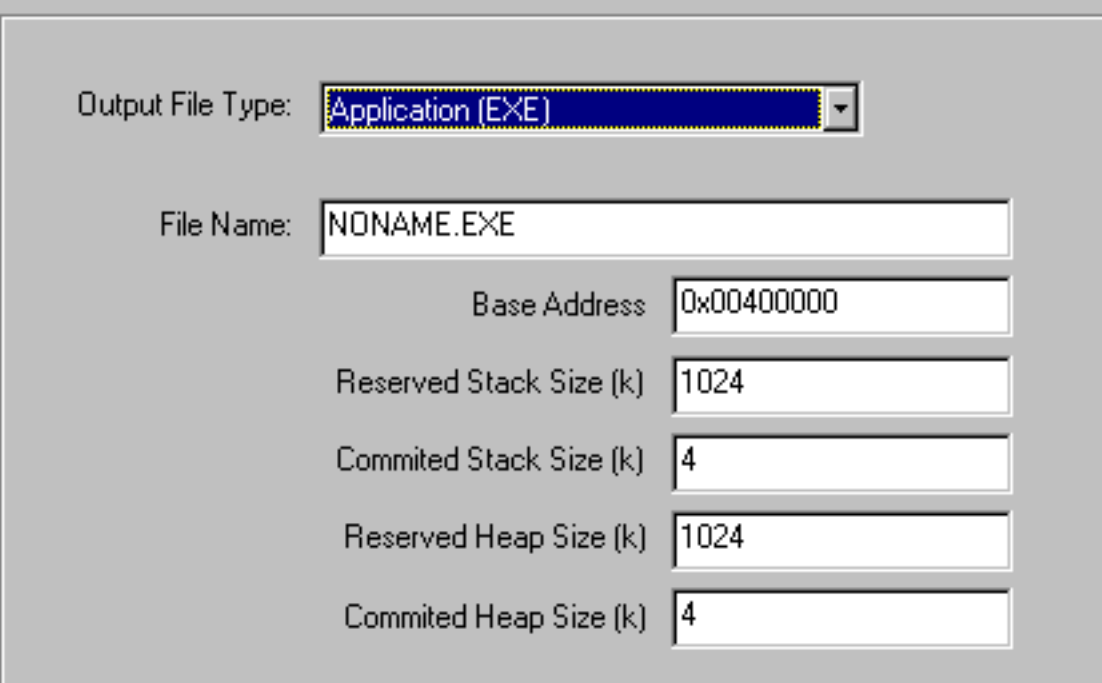
For a detailed explanation of how to configure these options, refer to the *Targeting Win32* manual on the CodeWarrior CD.

## Configuring IDE Options

### Choosing Project Settings

---

**Figure 8.19** x86 Project Options Panel



The screenshot shows a dialog box titled "x86 Project Options Panel". It contains several configuration fields:

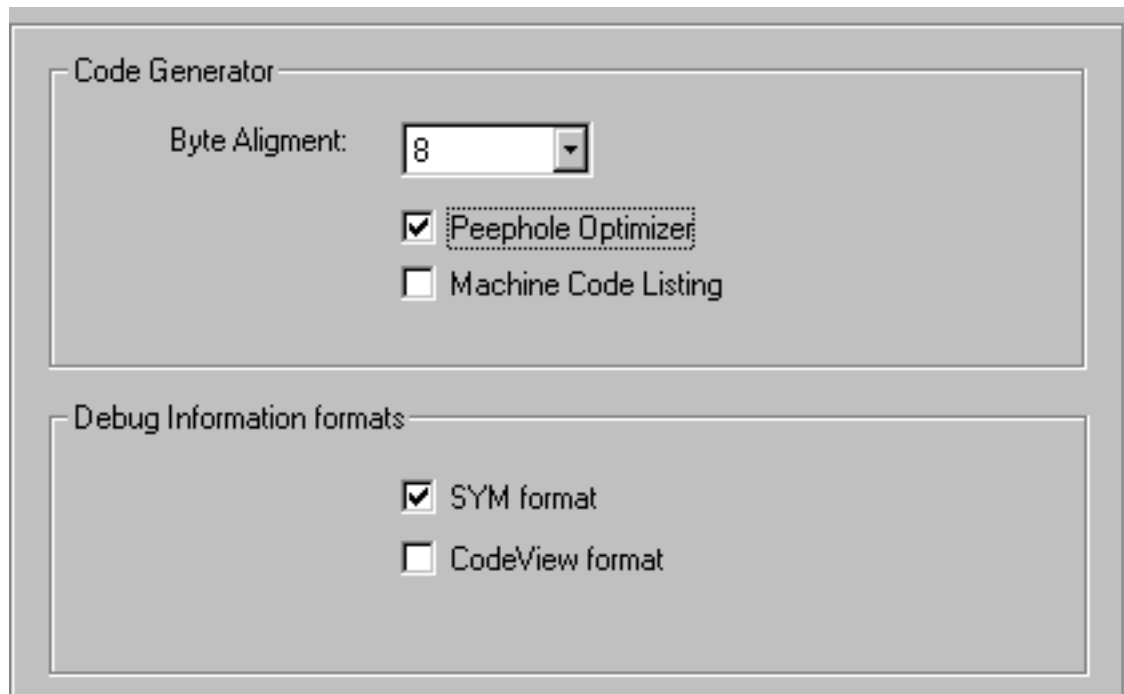
- Output File Type:** A dropdown menu with "Application (EXE)" selected.
- File Name:** A text field containing "NONAME.EXE".
- Base Address:** A text field containing "0x00400000".
- Reserved Stack Size (k):** A text field containing "1024".
- Committed Stack Size (k):** A text field containing "4".
- Reserved Heap Size (k):** A text field containing "1024".
- Committed Heap Size (k):** A text field containing "4".

### x86 CodeGen Panel

The x86 CodeGen options panel, shown in Figure 8.20, allows you to configure the code generation options for your project. For a detailed explanation of how to configure these options, refer to the *Targeting Win32* manual on the CodeWarrior CD.



**Figure 8.20** x86 CodeGen Options Panel



## **x86 Linker Panel**

The x86 Linker options panel, shown in Figure 8.21, allows you to configure the linker options for your project. For a detailed explanation of how to configure these options, refer to the *Targeting Win32* manual on the CodeWarrior CD.

## Configuring IDE Options

### Choosing Project Settings

---

**Figure 8.21** x86 Linker Options Panel

The screenshot shows the 'x86 Linker Options Panel' with the following fields and controls:

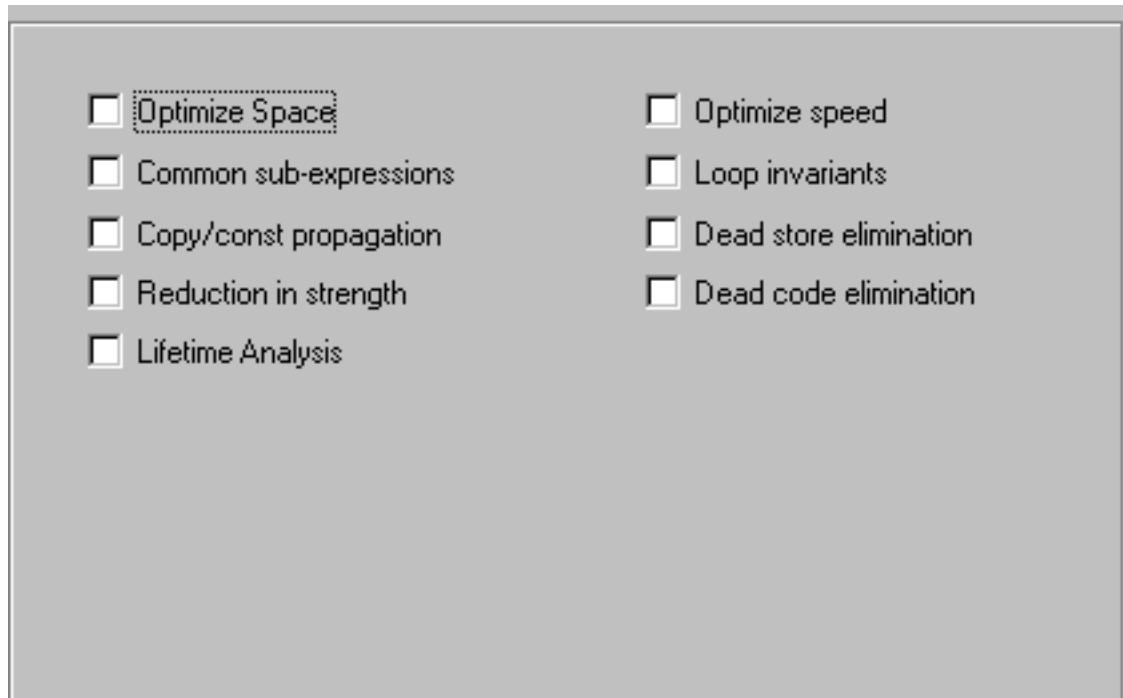
- Entry Point:** A section containing:
  - Entry Point Usage:** A dropdown menu currently set to 'Default'.
  - Name:** An empty text input field.
- Subsystem IDs:** A section containing:
  - SubSystem:** A dropdown menu currently set to 'Windows CUI'.
  - SubSystem ID:** A text input field containing '3.10'.
  - User ID:** A text input field containing '0.00'.
- Checkboxes:** Three checkboxes are located below the Subsystem IDs section:
  - ☐ **Generate Link Map** (This checkbox is highlighted with a dashed border in the image).
  - ☐ **Generate SYM File**
  - ☐ **Generate CodeView Info**
- Command File:** A text input field at the bottom of the panel.

## IR Optimizer Panel

The IR Optimizer options panel, shown in Figure 8.22, allows you to configure code optimization options for your project.

For a detailed explanation of how to configure these options, refer to the *Targeting Win32* manual on the CodeWarrior CD.

**Figure 8.22** IR Optimizer Options Panel



## WinRC Compiler

The WinRC Compiler options panel, shown in Figure 8.23, allows you to configure your options for the Win32 resource compiler.

For a detailed explanation of how to configure these options, refer to the *Targeting Win32* manual on the CodeWarrior CD.

## Configuring IDE Options

### *Choosing Project Settings*

---

**Figure 8.23** WinRC Resource Compiler





# Compiling and Linking

---

This chapter discusses how to control compilation, linking and running a CodeWarrior project.

## Compiling and Linking Projects Overview

The information in this chapter assumes you have already created a project, added the necessary files, grouped these files, and set the project's options. To learn more about how to do these things, refer to other chapters in this book, including "Projects Overview" on page 35, "Working with Files Overview" on page 69, "Source Code Editor Overview" on page 85, and "Configuring IDE Options Overview" on page 175.

You should also be familiar with features such as moving files in the project window, the project window's columns, and pop-up windows. To learn more about these things, refer to "Projects Overview" on page 35.

This chapter discusses how to compile and link a project to produce your code, and how to correct common compiler and linker errors using the Message window. It does not describe in detail the various types of programs CodeWarrior can create. For that information, please see the *CodeWarrior Targeting* manual appropriate for your platform. A table describing which guide to refer to is shown in "Targeting Guides for Various CodeWarrior Targets" on page 22.

The topics in this chapter are:

- Choosing a compiler
- Compiling and Linking a Project

## Compiling and Linking

### *Choosing a compiler*

---

- Using Precompiled or Preprocessed Headers
- Preprocessing Source Code (C/C++ only)
- Disassembling Source Code
- Guided Tour of the Message Window
- Using the Message Window

## Choosing a compiler

When you create source code files, you are using a certain programming language such as C, C++, Pascal, or another language. These languages have naming conventions for the files. For example, in the C language, a source code file ends with a .c suffix and a header file ends with a .h suffix.

This section describes how to assign a file suffix to a compiler for a given language in the CodeWarrior IDE.

### Understanding Plugin Compilers

The CodeWarrior IDE is designed to allow compilation of many different programming languages. In order to make the product modular to accept many different languages, plugin compilers are used.

Plugins are basically small loadable code modules that allow the IDE to have many different compilers at its disposal. For example, there are plugin compilers for C, C++, Pascal, Java, and assembly language.

### Setting a File Extension

To associate a plugin compiler with a given file, you set the Target Panel options. For a description of how to configure these options to set a compiler for your source code files, refer to “Target Panel” on page 192.

## Compiling and Linking a Project

The CodeWarrior IDE provides many different ways to build a project. When you build a project, you compile and link it.

All compiling and linking commands are available from the Project Menu and IDE Toolbar. Depending on your project type, a few of these commands may be disabled or renamed. For example, you cannot Run a code resource, but you can Make it. Also, a compiling or linking command may be dimmed because CodeWarrior is busy executing another command or the project is being debugged.

### Touching and Untouching Files

If you touch a file using one of the facilities in CodeWarrior, what you are doing is telling the compiler you want that file to be compiled the next time the project is built. If you untouch a file, you are telling the compiler to ignore that file the next time the project is built.

CodeWarrior compiles all touched files the next time you choose the Bring Up To Date command, the Make command, or the Run command.

To learn more information about how to select files for touching, refer to “Touching and Untouching Files” on page 62.

### Compiling Files

You can tell CodeWarrior to compile a single file, or compile certain files in your project. Of course, CodeWarrior can also compile all files in your project.

The Compile command on the Project Menu is dimmed when:

- There is no open project.
- The active Editor window does not have a source code file-name extension.
- The active window’s source code file is not included in your project.

## Compiling and Linking

### *Compiling and Linking a Project*

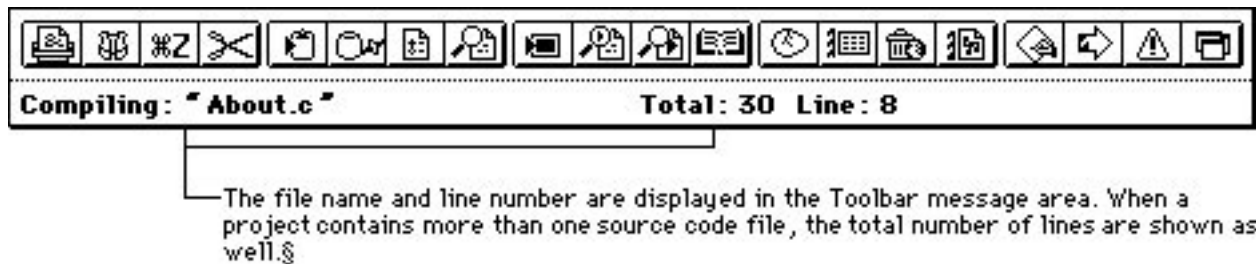
---

- The binary file, such as the application you created from the project, is running under the CodeWarrior debugger.
- An application created from the project is running.

As CodeWarrior compiles source code files and libraries in the open project, it highlights them in the Project window. The IDE Toolbar message area displays a line count and the name of the file being compiled in the message area (Figure 9.1).

A status line displaying the total number of files to be compiled and the number of the file being compiled is also provided at the bottom of the project window.

**Figure 9.1 Compiler progress**



### Compiling One File

To compile a single file in your project, select that file in your Project window and choose Compile from the Project Menu. To learn how to select several files in your project, refer to “Selecting Files and Groups” on page 52.

Alternatively, you can open the file in the CodeWarrior Editor, make the window the active window, and choose Compile from the Project Menu.

### Compiling Selected Files

You may select numerous files in your project for compilation, by selecting the files in the Project window and then choosing Compile from the Project Menu. To learn how to select several files in your project, refer to “Selecting Files and Groups” on page 52.



## **Recompiling Files**

You can force CodeWarrior to recompile a file that CodeWarrior may not recognize as changed. To do this, you must Touch the file first. To learn how to Touch a file, refer to “Touching and Untouching Files” on page 213.

After touching the file or files that you want to recompile, choose Compile or Make from the Project Menu.

## **Updating a Project**

When you have many newly added, modified, or touched files in your project, you can use the Bring Up To Date command on the Project Menu to compile all the files. The IDE Toolbar window shows status information on the compilation process, as shown in Figure 9.1 on page 214.

When using this command, the linker will not be invoked, so your project will not have its output binary produced. This command only runs the compiler.

To learn more about touching files, refer to “Touching and Untouching Files” on page 213.

## **Making a Project**

When you are ready to produce your binary file, such as an application, library, or shared library, you use the Make command on the Project Menu. This command builds the selected project type by updating the newly added, modified, and “touched” files, then linking the project.

The results of a successful build depend on the selected project type. For example, if the project type is an application, the Make command builds an application and saves it in the same folder as your project. Table 9.1 lists each project type and what is built when the Make command is executed.

**Table 9.1 Results of successful build**

Project Type	Target	Make Creates
Application	Win32	Win32 application
Library	Win32	Win32 Library (.lib)
Dynamic link library	Win32	Win32 Dynamic Link Library (.dll)

Once all the modified files and “touched” files have been compiled successfully, CodeWarrior links all the files in the project to produce your output binary. If the project has already been compiled using Bring Up To Date or another command, then the Make command only links the compiled source code files together.

## Enabling Debugging

When the Enable Debugger option is checked in the Project Menu, choosing the Debug command lets the CodeWarrior Debugger launch and debug your project. When you choose Disable Debugger from the Project Menu, choosing the Run command runs your project normally.

To learn about how to configure your project so that files in the project have debugging information generated for them, refer to “Controlling Debugging in a Project” on page 64.

To learn more about running your project, refer to “Running a Project” on page 216.

## Running a Project

When you choose the Run command from the Project Menu, CodeWarrior compiles and links (if necessary), and creates a stand-alone application, then launches that application.

When compiling and linking is successful, CodeWarrior saves a new application in the same folder as the open project. It is named according to options you set. If you would like to change these options, refer to “Choosing Project Settings” on page 191.

If the current project is for a library or shared library (DLL), the Run command is not available.

## Debugging a Project

To debug your project, there are basically two steps you need to do. Of course, you must already have your project compiled and linked with debugging information generated. To learn how to enable debugging for your project, refer to “Enabling Debugging” on page 216.

The second step is for you to start the debugger with your compiled application as the debug target. You can do this by selecting the Debug command from the Project Menu. If the Debug command is not in the menu, you do not have debugging enabled for your project. Refer to “Enabling Debugging” on page 216 to learn how to remedy this.

Once you have selected the Debug command, CodeWarrior compiles and links your project, creates a debugging information file, and then opens that debug information file with the CodeWarrior Debugger.

If the Debug command is dimmed, make sure the proper options for debugging are configured, as detailed in “Enabling Debugging” on page 216. Also, make sure that Metrowerks Debugger application is on your hard disk. If Debug is still dimmed, you are probably attempting to run a project whose project type cannot be run, such as a shared library or library.



---

**NOTE:** *The Debug command does not open any application that you may need to debug your project. If you’re debugging an Adobe Photoshop Plug-in or any other project that requires an application, you must launch the application on your own before you choose Debug.*

---

Once you are in the Metrowerks Debugger, you need to refer to the *CodeWarrior Debugger Manual* for information on how to use it.



---

**TIP:** *If you want to switch from the debugger while you are in a CodeWarrior Editor window, use the Switch to Debugger command on the File Menu.*

---

### Generating a Link Map

Metrowerks C/C++ compilers let you create a link map file that contains function and class section information on the generated object code.

Metrowerks Pascal compilers let you create a make map file that contains a list of dependencies and the compilation order.

To learn how to configure your project to create one of these files, refer to “x86 Linker Panel” on page 207. After configuring your project, you will need to Make your project. If the compile and link is successful, a link map file name after your project with the MAP extension is saved in your project folder.

### Synchronizing Modification Dates

If you want to update the modification dates stored in the project file for all files in your project, choose the Synchronize Modification Dates command from the Project Menu.

To learn more information about this topic, see “Synchronizing modification dates” on page 63.

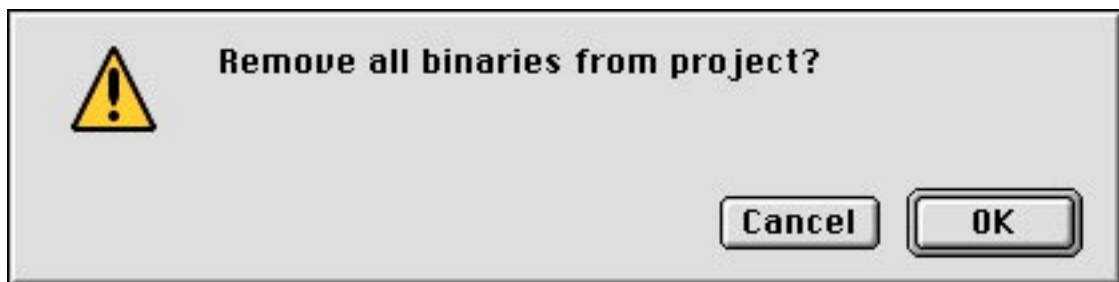
### Removing Binaries

When you compile your project in CodeWarrior, binary object code is added to the project for each file in the project. This binary object code increases the size of the project file. There are a few different commands available if you want your project file to consume less memory on the hard disk, or you want to remove all object code and start compilation over again.

## Removing Object Code

In some cases, you may wish to remove all the object code from the project and restart the compiling and linking process. To remove a project's binaries, select the Remove Binaries command from the Project Menu. The dialog box shown in Figure 9.2 appears.

**Figure 9.2** Remove Binaries dialog box



Clicking OK removes all object code data, resetting the Code and Data size of each file in the project window to zero. Cancel aborts the operation so none of your object code is removed.

## Remove Binaries & Compact command

The Remove Binaries & Compact command removes all binaries from the project and compacts it to consume the minimum amount of space on your hard disk.

Compacting the project removes all object code and debugging information stored in the project file and retains only the information about which files belong to the project and project-specific option settings.

## Advanced Compile Options

This section describes two options that either speed up your project build times or alert you when a build is completed.

## Compiling and Linking

### *Using Precompiled or Preprocessed Headers*

---

#### Alerting Yourself After a Build

You may start a project compile/link cycle in CodeWarrior, then switch to another application running on your machine. To learn how to receive notification when the build is completed, refer to “Build Extras” on page 199.

#### Speeding Up a Build by Avoiding Date Checks

To learn about how to optimize the speed of your builds in CodeWarrior, refer to “Use Modification Date Caching” on page 200.

## Using Precompiled or Preprocessed Headers

Source code files in a project typically use many header files (“.h” or “.hpp” files). Often, the same header file is included in many different source code files, forcing the compiler to (inefficiently) read the same header files many times during the compilation process.

To shorten the time spent compiling and recompiling a header file, use Precompile on the Project Menu. A precompiled header file takes the compiler significantly less time to process than an ordinary, uncompiled header file.

For instance, a header file that contains the most frequently used headers in your project could be made into one precompiled header file. Instead of having to compile the same thousands of lines of header files for each source file in your project, the compiler only has to load one precompiled header file.



---

**NOTE:** *You can only include one precompiled header in a source file. Including more than one precompiled header will result in an error.*

---



---

**TIP:** *CodeWarrior frequently changes the precompiled header format when implementing new features in CodeWarrior updates.*

---

*Therefore precompiled header formats are often incompatible between CodeWarrior updates. After installing a new CodeWarrior update, you usually need to precompile your precompiled headers to use the new format. See “Automatic updating” on page 222 for more information on updating precompiled headers.*

---

The topics in this section are:

- Creating Precompiled Headers
- Defining Symbols For C/C++
- Defining Symbols For Pascal

## Creating Precompiled Headers

To precompile a header file, you must first open a project. The option settings from this project are used when precompiling. A file to be precompiled does not have to be a header file (“.h” or “.hpp”), but it must meet these requirements:

- The source file must be a text file. You cannot precompile libraries or other files.
- The file does not have to be in a project, although a project must be open to precompile.
- It must not contain any statements that generate data or code. However, C++ source code can contain inline functions and constant variable declarations (`const`).
- Precompiled header files for different targets are not interchangeable. For example, to generate a precompiled header to use with Win32 compilers, you must use a Win32 compiler.
- A source file can include only one precompiled header file using the `#include` directive.

Create a source code file using New on the File Menu. In that file, put your `#include` directives. For example, if you wanted to create a precompiled header file of the files `string.h` and `stdio.h`, just put the following in your source code file:

## Compiling and Linking

### *Using Precompiled or Preprocessed Headers*

---

---

```
#include <stdio.h>
#include <string.h>
```

---

Then, save your source code file using the Save command on the File Menu. Use a name ending in the .pch extension, such as My-Header.pch, then use the Precompile command on the Project Menu to precompile your file. By default, CodeWarrior precompiles the source code, then saves it as the same name as your .pch file, without the .pch extension. For example, myHeader.pch would be precompiled and saved as myHeader on your hard disk.

To specify the precompiled header filename in source code instead, add

```
#pragma precompile_target "name"
```

as the first line in the source code file. This pragma tells the compiler to save the precompiled header as *name*. With this pragma as the first line in the source code, you can rename your file whatever you want, bypassing the default.

### **Precompile command**

To precompile a file, choose Precompile from the Project Menu. This command precompiles the source code file in the active window, creating a precompiled header file. The progress of this operation is displayed in the IDE Toolbar. If compiler errors are detected, a Message window appears.

To learn more about the Message window and correcting compiler errors, consult “Correcting Compiler Errors and Warnings” on page 238.

### **Automatic updating**

During a Make or Bring Up To Date operation, CodeWarrior automatically updates a precompiled header if the source has been modified.



If CodeWarrior encounters a .pch or .pch++ file in the project that was modified since it was last precompiled, CodeWarrior precompiles it again to ensure that the resulting precompiled header is up to date.

To create a precompiled header file that is automatically updated, open the project that will use the precompiled header. Then create a source code file that will be used as the precompiled header's source file.

To read about the requirements for a recompiled header source file, refer to "Creating Precompiled Headers" on page 221.

In the first line of the source code file, add the line

```
#pragma precompile_target "name"
```

This pragma tells the compiler to create a precompiled header with the filename of *name*.

Save the source file with a .pch filename extension.

Now add the source file to the open project with the Add Window command in the Project Menu.

Whenever the .pch file is modified, the CodeWarrior project manager will automatically update it by precompiling it.

To include the precompiled header in a project source code file, add this line as the first #include directive in the file:

```
#include "name"
```

Alternatively, you may also specify the use of a precompiled header file in your project settings. To learn about how to do this, refer to "C/C++ Compiler Panel" on page 201.



---

**NOTE:** *Do not use the .pch source file in #include directives; use the name of the resulting precompiled header file. Although using the .pch file is legal and will not affect the final binary, you won't be taking advantage of the precompiled header's speed.*

---

## Defining Symbols For C/C++

To automatically update and add predefined symbols and other preprocessor directives, you can create a precompiled header file, add it to your project, and place its name in the appropriate C/C++ Compiler Panel option.

First, open your project and create a new source code file with the New command on the File Menu.

This new text file will contain your preprocessor directives. You'll use this file as a precompiled header file that you will add to your project.

Choose Project Settings from the Edit Menu, and select the C/C++ Compiler Panel settings panel.

If there is a filename in the Prefix File box, Copy it into the Clipboard, then click OK to close the dialog box.

In the new Editor window, paste the filename used in Prefix File in an include directive. Make sure this is the *first* directive in the file.

For example, if the Prefix file is MyHeaders, then the first directive in the editor window is

```
#include <MyHeaders>
```

Include the `#pragma precompile_target` statement. This statement lets you name the precompiled file. For example, to create a file named MyPrecomp, use this statement

```
#pragma precompile_target "MyPrecomp"
```

Type in all your own `#define`, `#include`, and other preprocessor directives.

Save this file with a `.pch` or filename extension.

Use `.pch` as a filename extension. For example, save this file as "MyPrecomp.pch".

Choose Add Window from the Project Menu to add this precompilable header file to your project.

Choose Project Settings from the Edit Menu and select the C/C++ Compiler Panel settings panel.

In the Prefix File field, enter your precompiled file's name, in this example "MyPrecomp". Click OK to save your changes.

Whenever your project is built, the CodeWarrior project manager updates your precompiled header and automatically includes it in each source code file.

## Defining Symbols For Pascal

Although the Pascal preprocessor is not as powerful as the C/C++ preprocessor, you can still create files that can automatically insert your own preprocessor symbols and compiler directives into your project. For more information on the Pascal compiler directives, see the Pascal Language Manual on the CodeWarrior Reference CD.

Open your project and create a new source code file with the New command. This new text file will contain your compiler directives.

Choose Project Settings from the Edit Menu, and select the Pascal Compiler settings panel.

If there is a filename in the Prefix File box, Copy it into the Clipboard, then click OK.

In the new editor window, paste the filename used in Prefix File in an include directive, `{I}`. Make sure this is the *first* directive in the file.

For example, if the Prefix file is OtherDefs.p, then the first directive in the editor window is

```
{I OtherDefs.p}
```

Type in all your own `$(SETC)`, `{I}`, and other preprocessor directives. This file cannot contain any source code that generates data or executable code.

## Compiling and Linking

### *Preprocessing Source Code (C/C++ only)*

---

Save this file as an ordinary Pascal source code file in the same folder as your project. For example, save this file as “MyPre-comp.pas”.

Choose Project Settings from the Edit Menu, and select the Pascal Compiler settings panel.

In the Prefix File field, enter your file's name, in this example “MyPrecomp”, then click OK to save your changes.

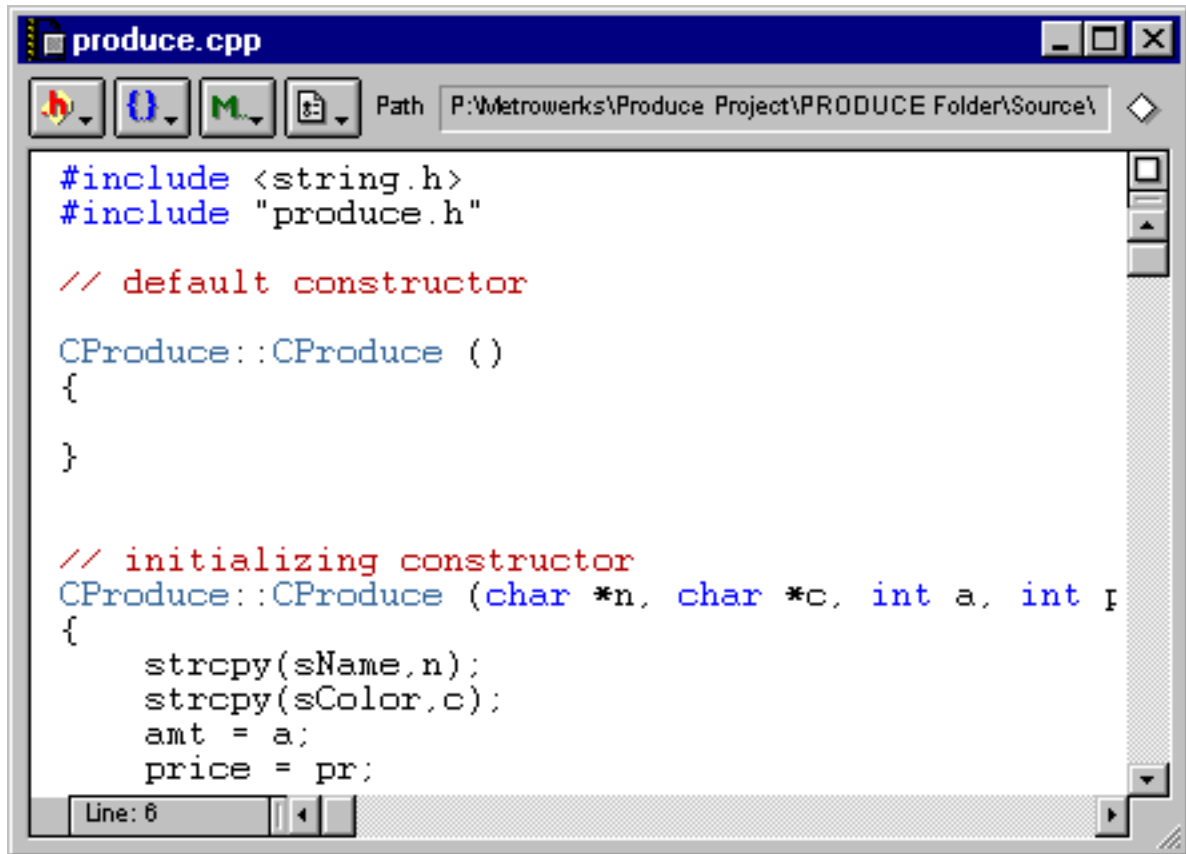
Whenever your project is built, the CodeWarrior project manager automatically includes it in each source code file.

## Preprocessing Source Code (C/C++ only)

The C/C++ preprocessor prepares source code for the C/C++ compiler. It interprets directives beginning with the “#” symbol (such as `#define` and `#ifdef`), removes extra spaces and blank lines, and removes comments (`/*...*/` and `//`). You might want to preprocess a file if you want to see what the code looks like just before compilation.

For example, Figure 9.3 shows a source code file before using the Preprocess command on the Project Menu.

**Figure 9.3** Source code before Preprocess



Open a file that you want to preprocess, or select a file in your currently-open Project window. To preprocess a file, select the Preprocess command on the Project Menu. The results of the Preprocess command are stored in a new file named after the source code file that was preprocessed and beginning with the “#” character (Figure 9.4).

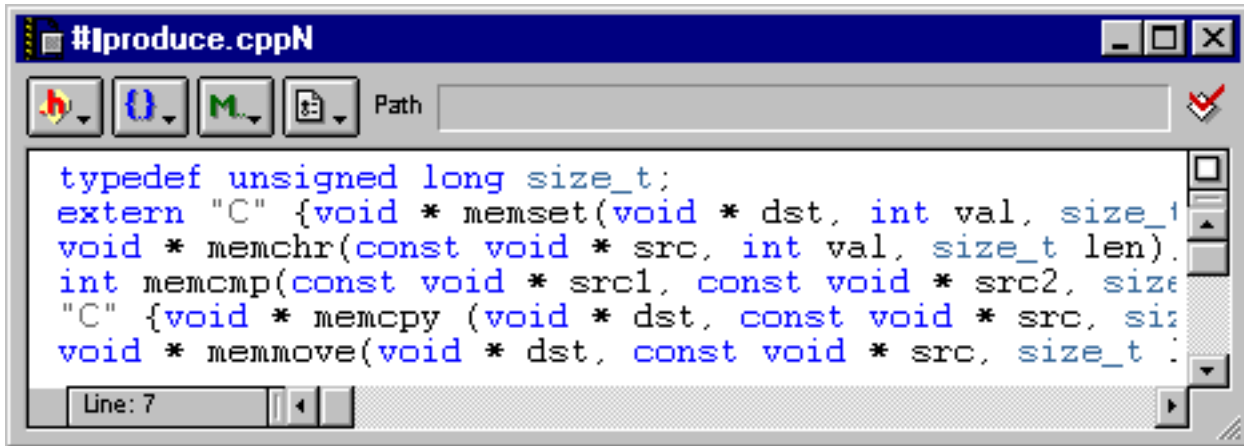
To save the contents of the new window, choose one of the save commands in the File Menu.

## Compiling and Linking

### Disassembling Source Code

---

Figure 9.4 Preprocessor output



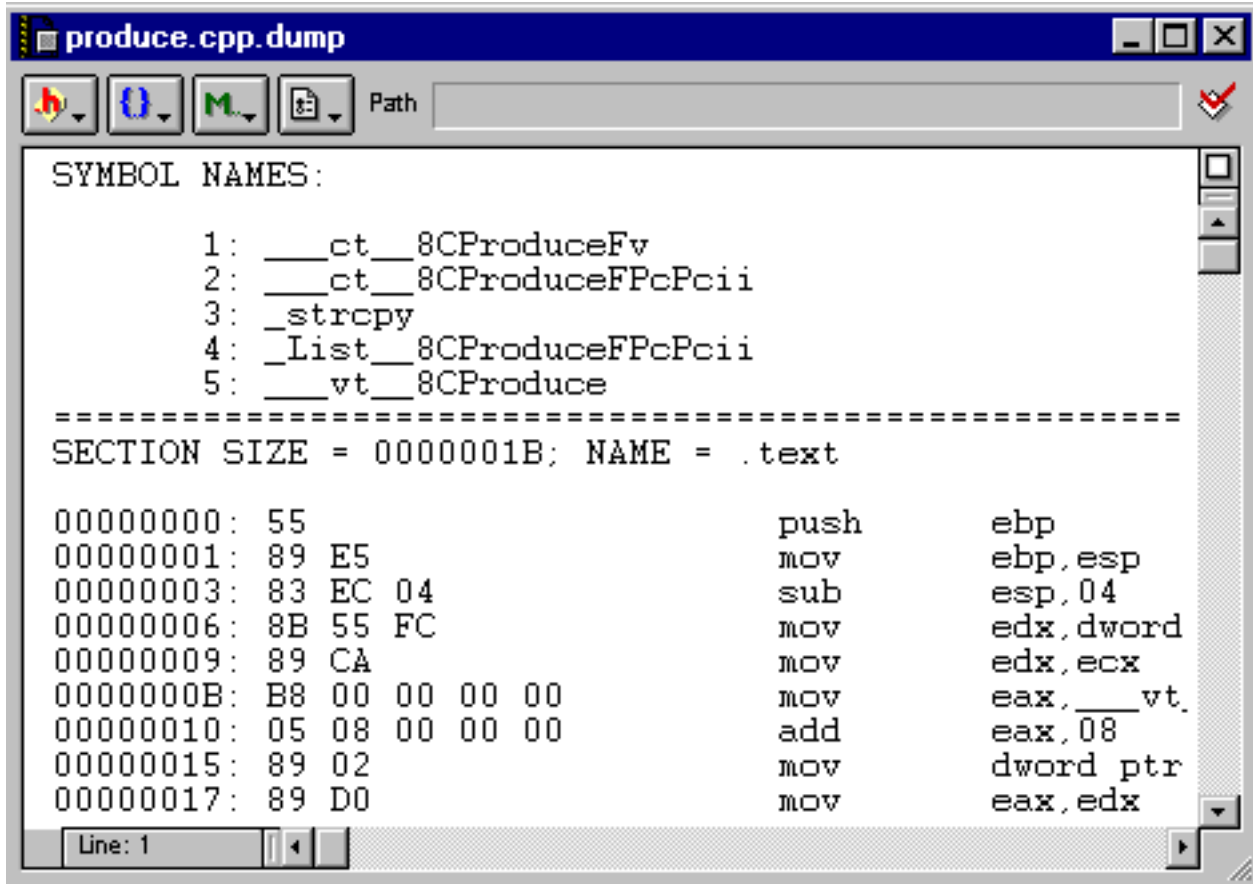
## Disassembling Source Code

If you wanted to see the code that would be generated for your file you could disassemble the file. Disassembling is useful if you want to know the machine level calls that are being made when your source code is executed. In addition, the disassembled code can be a model for writing your own assembly routines.

The Disassemble command on the Project Menu disassembles the compiled source code file selected in the project window and displays its assembly-language code in a new window. The title of the new window consists of the name of the source code file with the extension “.dump” (Figure 9.5).

To save the contents of the “.dump” window, choose one of the save commands in the File Menu.

Figure 9.5 Disassembling a selected source code file



The screenshot shows a window titled 'produce.cpp.dump'. It contains a list of symbol names and a disassembly of a section named '.text'. The symbol names are: 1: \_\_ct\_\_8CProduceFv, 2: \_\_ct\_\_8CProduceFPcPcii, 3: \_strcpy, 4: \_List\_\_8CProduceFPcPcii, and 5: \_\_vt\_\_8CProduce. The disassembly section shows instructions for the '.text' section, starting at address 00000000. The instructions are: push ebp, mov ebp, esp, sub esp, 04, mov edx, dword, mov edx, ecx, mov eax, \_\_vt., add eax, 08, mov dword ptr, and mov eax, edx.

```
produce.cpp.dump
Symbol Names:
1: __ct__8CProduceFv
2: __ct__8CProduceFPcPcii
3: _strcpy
4: _List__8CProduceFPcPcii
5: __vt__8CProduce

=====
SECTION SIZE = 0000001B; NAME = .text

00000000: 55                push     ebp
00000001: 89 E5            mov     ebp, esp
00000003: 83 EC 04         sub     esp, 04
00000006: 8B 55 FC         mov     edx, dword
00000009: 89 CA            mov     edx, ecx
0000000B: B8 00 00 00 00   mov     eax, __vt.
00000010: 05 08 00 00 00   add     eax, 08
00000015: 89 02            mov     dword ptr
00000017: 89 D0            mov     eax, edx
```

If the file being disassembled has not been compiled, the disassemble command will compile the file before disassembling it.

## Guided Tour of the Message Window

The Message window is used to display messages about events that have occurred when compiling, linking, or searching files. There are a number of elements in the window that are useful for accomplishing certain tasks, such as navigating to error locations and scrolling to see all messages for a project.

The topics in this section include:

## Compiling and Linking

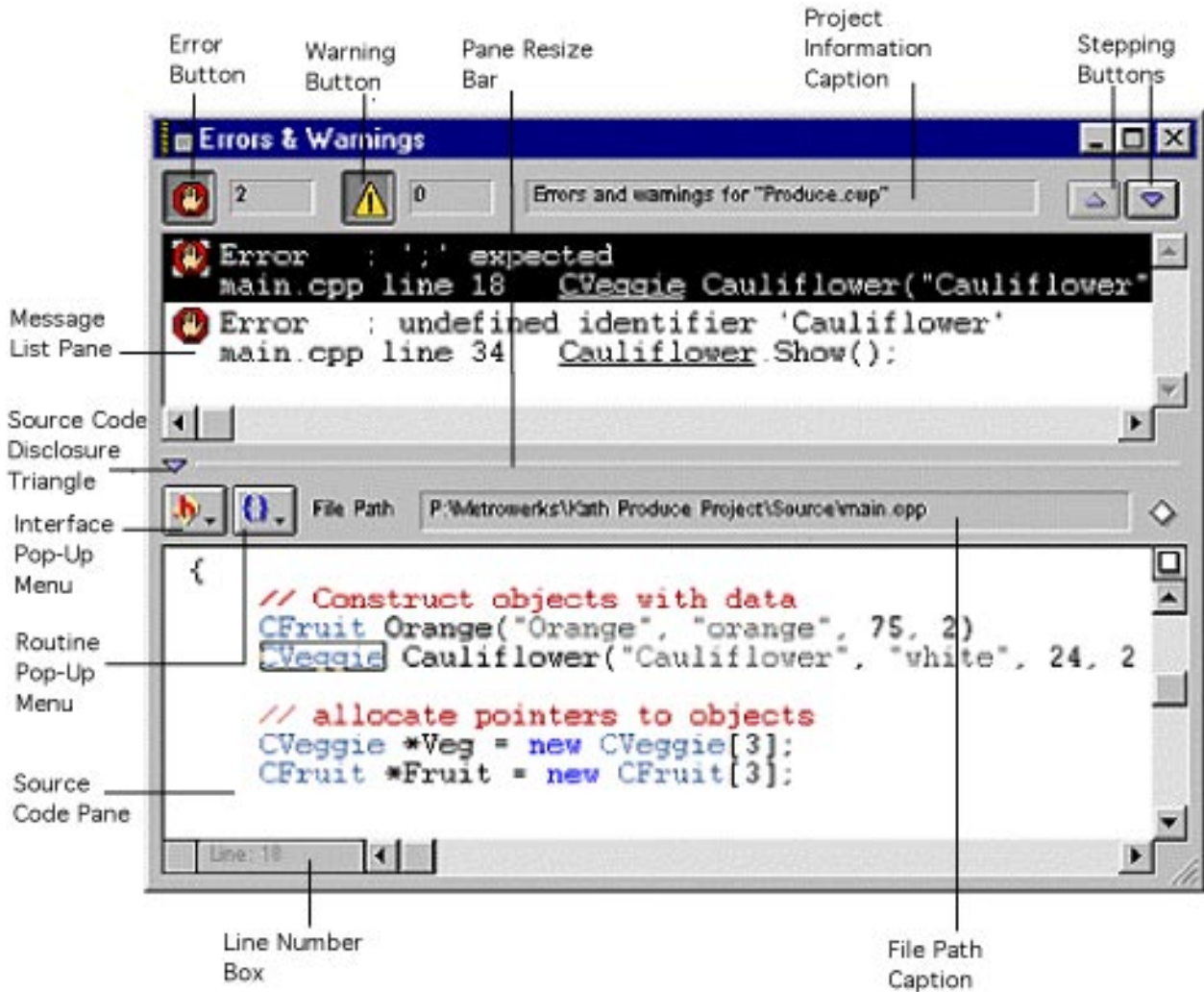
### *Guided Tour of the Message Window*

---

- Error Button
- Warning Button
- Project Information Caption
- Stepping Buttons
- Message List Pane
- Source Code Disclosure Triangle
- Source Code Pane
- Pane Resize Bar



**Figure 9.6** The CodeWarrior Message Window



## Error Button



The Error Button in the Message window, shown in Figure 9.6, toggles the view of error messages on and off. This is useful if you have changed the view of the window to something else and want to get back to viewing the error messages.

To learn more about seeing error messages in the Message window, refer to "Seeing Errors and Warnings" on page 235.

### Warning Button



The Warning Button in the Message window, shown in Figure 9.6 on page 231, toggles the view of warning messages on and off

To learn more about seeing error messages in the Message window, refer to “Seeing Errors and Warnings” on page 235.

### Project Information Caption

The Project Information Caption, shown in Figure 9.6 on page 231, gives a short description of the view you are looking at in the Message window. Your project name will appear here.

### Stepping Buttons

The Stepping Buttons, shown in Figure 9.6 on page 231, allow you to step up or down through your messages in the window.

To learn more about stepping through messages in the Message window, refer to “Stepping Through Messages” on page 236.

### Message List Pane

The Message List Pane, shown in Figure 9.6 on page 231, allows you to view your messages.

To learn more about seeing messages in the Message window, refer to “Seeing Errors and Warnings” on page 235.

### Source Code Disclosure Triangle

The Source Code Disclosure Triangle, shown in Figure 9.6 on page 231, allows you hide the Source Code Pane of the Message window.

## **Source Code Pane**

The Source Code Pane of the Message window allows you to view the source code at the location where a message is referring. To learn more information about the view in this window, refer to “Seeing Errors and Warnings” on page 235.

## **Pane Resize Bar**

The Pane Resize Bar allows you to reallocate the amount of space in the Message window given to the Source Code Pane and Message List Pane. By clicking and dragging this bar up or down you will change the amount of space on your computer screen that is allocated to these panes.

## **Pop-Up Menu Disclosure Button**

To learn more about the Pop-up Menu Disclosure Button, refer to the discussion of the CodeWarrior Editor in “Pop-Up Menu Disclosure Button” on page 93.

## **Interface Pop-Up Menu**

To learn more about the Interface Pop-up Menu, refer to the discussion of the CodeWarrior Editor in “Interface Pop-Up Menu” on page 87.

## **Routine Pop-Up Menu**

To learn more about the Routine Pop-up Menu, refer to the discussion of the CodeWarrior Editor in “Routine Pop-Up Menu” on page 88.

## **File Path Caption**

To learn more about the File Path Caption, refer to the discussion of the CodeWarrior Editor in “File Path Caption” on page 91.

## Line Number Button

To learn more about the Line Number Button, refer to the discussion of the CodeWarrior Editor in “Line Number Button” on page 92.

## Using the Message Window




While compiling your project, CodeWarrior may detect a syntax error or other type of compiler error in one of your project’s source code files. If this happens, the Message window displays the total number of errors and warnings, as well as information about each one.

In this section, you will learn how to interpret, navigate, and use the information that appears in the Message window. The topics in this section include:

- Seeing Errors and Warnings
- Stepping Through Messages
- Correcting Compiler Errors and Warnings
- Correcting Linker Errors
- Correcting Pascal Circular References
- Saving and Printing the Message Window
- Locating Errors in Modified Files

## Seeing Errors and Warnings

The Message window displays several types of messages:

Select this ...	To display this...
 Errors	Either compiler or linker errors. Both types of errors prevent the compiler and linker from creating a final binary.
 Warnings	Either compiler or linker warnings. Neither type prevents CodeWarrior from creating a binary. However, they indicate potential problems during runtime. You can specify which conditions lead to warning messages or you can upgrade all warnings to errors.
 Notes	All other types of messages issued in the Message window. For example, results of a batch find are notes messages.

To close the Message window, click its close box or select Close in the File Menu while the Message window is the active window. If you close the message window and want to see it again, choose the Errors & Warnings Window command from the Window Menu to reopen it.

To see only error messages in the Message List Pane, click on the Error Button and turn off the Warning Button.

To see only warnings in the Message List Pane, click the Warning Button and turn off the Error Button.

To see both errors and warnings in the Message List Pane, click both buttons. Notes do not appear in the Errors & Warnings window.

You'll also see other types of messages from time to time in a Message window, such as:

- During Add Window or Add Files when a file being added does not reside on an existing access path.
- During linking when a project contains conflicting resources.

## Compiling and Linking

### *Using the Message Window*

---

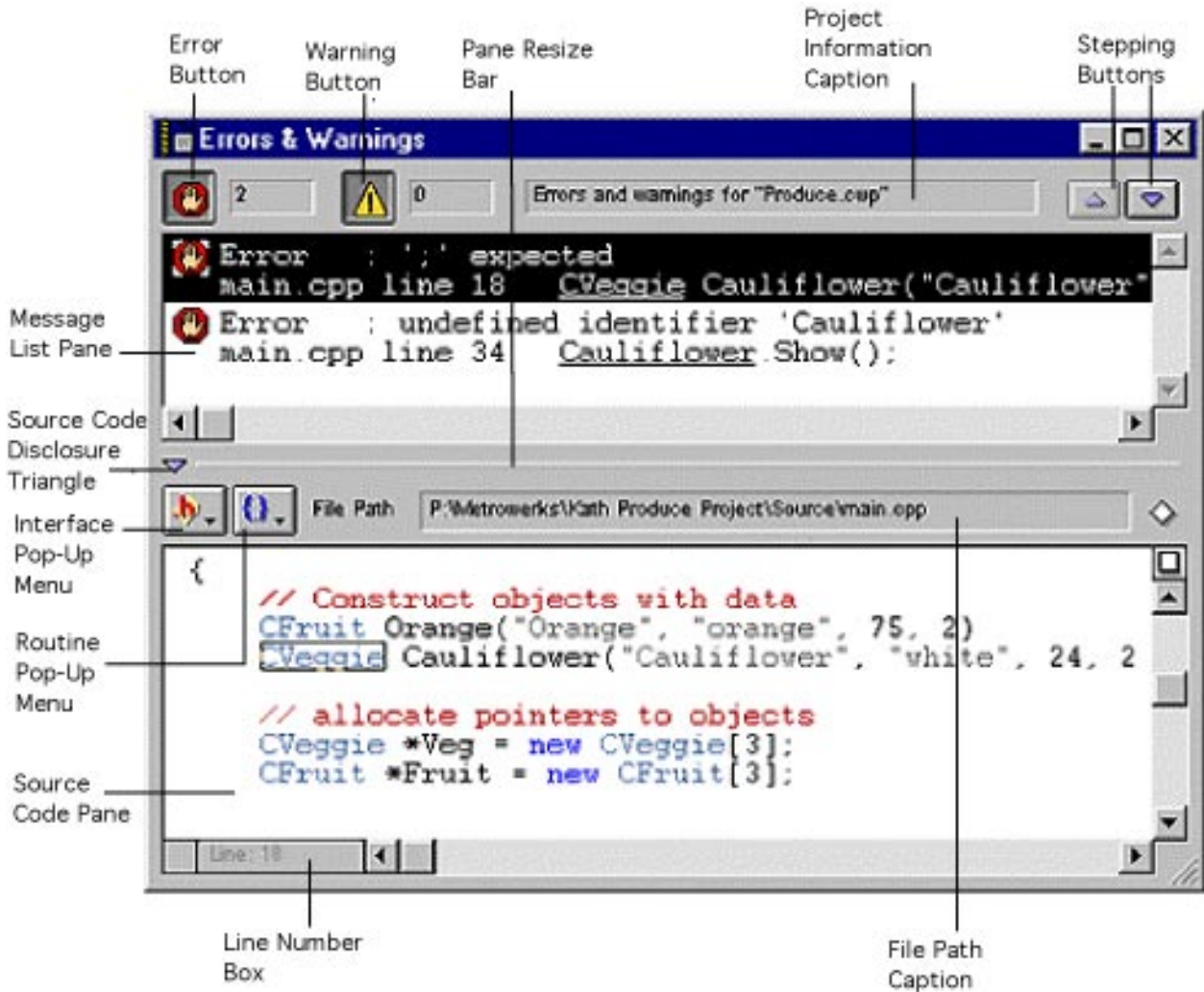
- During a Find when the Batch Check Box is selected in the Find dialog box.

## Stepping Through Messages

When the compiler finds errors during a build, or the CodeWarrior search engine finds text you asked it to look for when Using Batch Searches, you'll see the message window, as shown in Figure 9.7. The window is divided into two panes:

- Message List Pane, which lists the messages
- Source Code Pane, which displays the source code for the selected message

**Figure 9.7 Errors and Warnings Message Window**



To step through the list of messages, click the up or down Stepping Buttons or click the error message you are interested in.

To navigate your source code that is shown in the Source Code Pane for a given message, you use the Interface Pop-Up Menu, Routine Pop-Up Menu, or the Line Number Button. To learn about how to use these navigational features, refer to "Guided Tour of the Editor Window" on page 85.

## Correcting Compiler Errors and Warnings

When an error occurs during compilation, the Message window will show you the error message in the Message List Pane. The location in the source code that the message refers to will be shown in the Source Code Pane. You can navigate to the spot in your source code where the message refers to, and inspect or correct your code.

For a complete list of compiler errors and their possible causes, consult the *Error Reference* documentation on your CodeWarrior CD.

### Correcting Errors in the Source Code Pane

To correct a compiler error or warning, you must first find the cause. First, make sure that the Source Code Pane of the Message window is visible. If it isn't visible, refer to "Source Code Disclosure Triangle" on page 232 to learn how to make it visible.

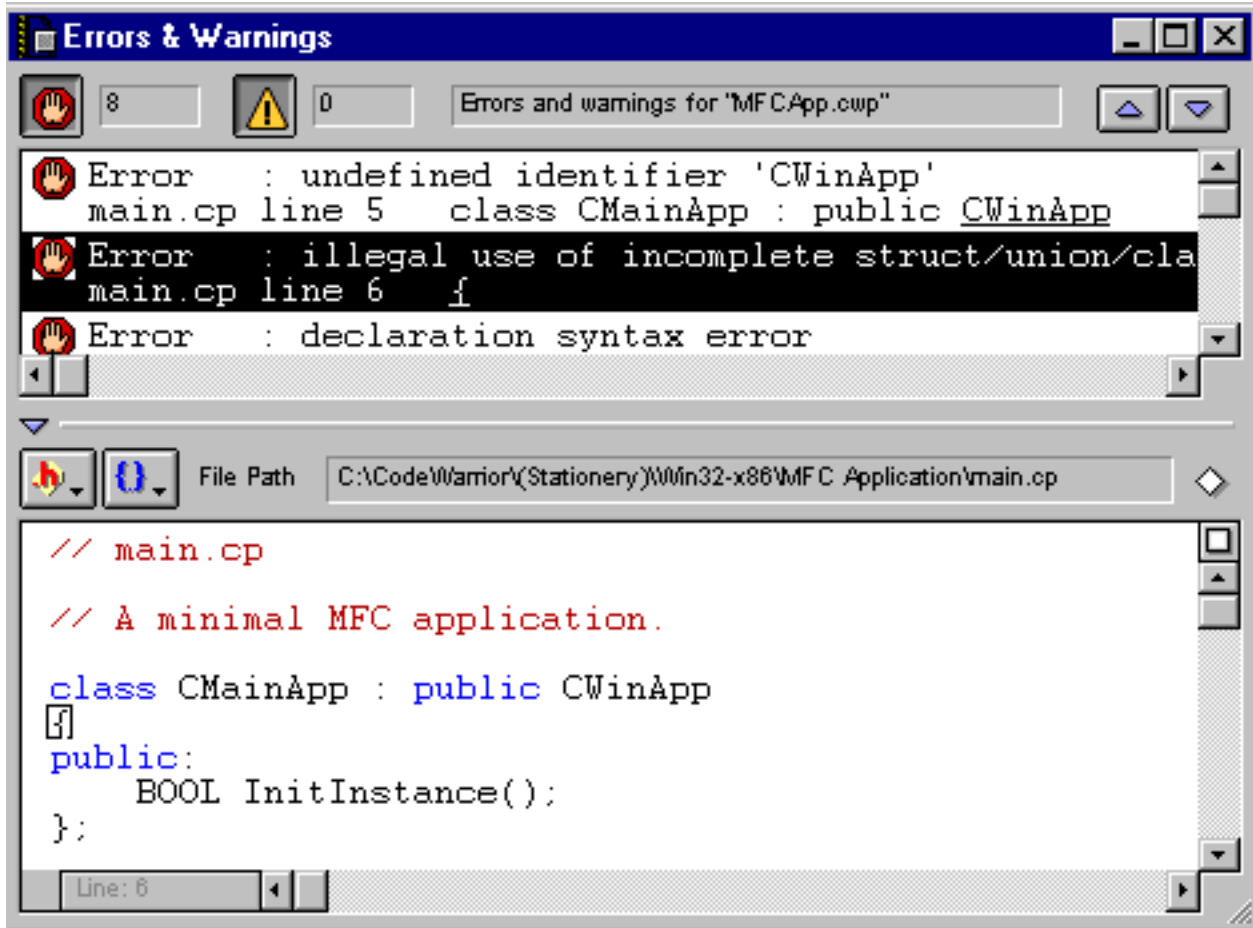
To view the statement that the compiler believes has caused the error or warning, select the message in the Message List Pane of the Message window, and notice that the Source Code Pane view is now showing the source code that corresponds to the message.

When you do, the file containing the error is brought to the front. The corresponding error is selected in the Source Code Pane, as shown in Figure 9.8.

You can use the Interface Pop-Up Menu, Routine Pop-Up Menu, or the Line Number Button in the Source Code Pane to navigate your code or open interface files. To learn about how to use these navigational features, refer to "Guided Tour of the Editor Window" on page 85.



Figure 9.8 Source Code Pane Compilation Error Location



### Opening the File for the Corresponding Message

To open a source code file that corresponds to a given message, select the message in the Message List Pane and press Return. You may also double-click the message in the Message List Pane to open the relevant file.

### Correcting Linker Errors

If the linker encounters any errors while linking your project, the Message window appears indicating these errors (Figure 9.9). This window can be scrolled through using the scroll bar or arrow keys.

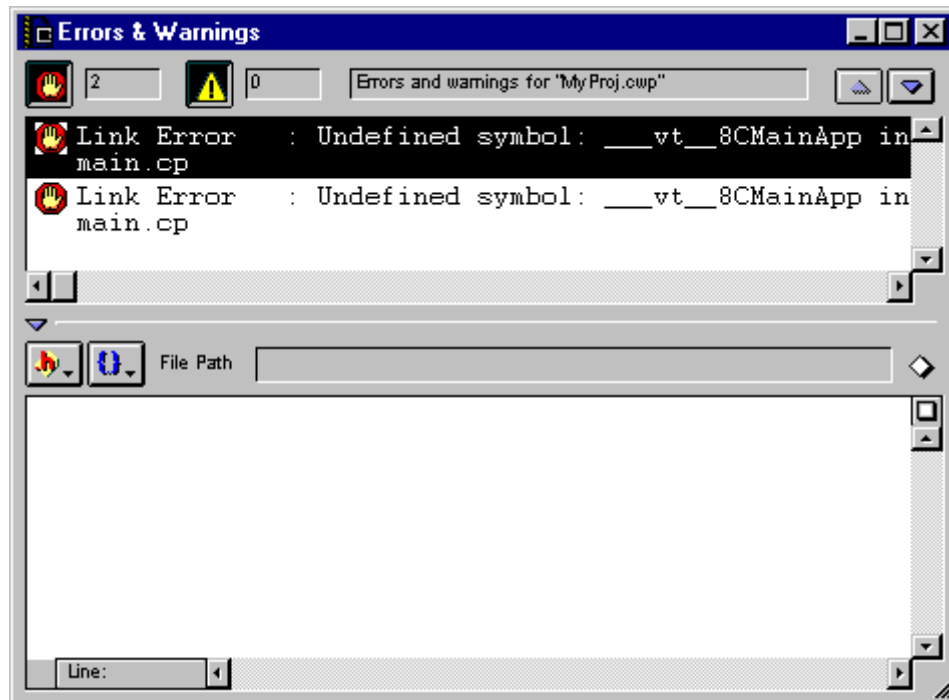
## Compiling and Linking

### Using the Message Window

---

To learn about how to scroll through messages in the Message window, refer to “Stepping Buttons” on page 232. To learn about changing the view of messages in the Message window, refer to “Seeing Errors and Warnings” on page 235.

**Figure 9.9** Linker errors message window



Since Linker errors are a result of problems in the object code, CodeWarrior cannot show their corresponding errors in the project's source code files.

Linker errors are usually the result of one of the following circumstances:

- Your project is missing the necessary libraries. To find out which libraries or shared libraries should be added to your project, refer to the *CodeWarrior Targeting* manual appropriate for your platform, as described in Table 1.2 on page 22. Linker error messages of this type occur when the project is missing a library.

- You have misspelled the name of a library routine. This means that the routine that the Linker is searching for does not exist. Check the name of the routine to make sure it is spelled correctly.

## Correcting Pascal Circular References

The Make and Run command for the Metrowerks Pascal compiler builds your project by examining every Pascal file in your project file. As this examination is performed, a tree of dependencies is built for the interfaces of your units and for their implementations.

A circular reference occurs when a unit declares something that is used in another unit and that same unit declares something used by the former. To break this loop, the Pascal compiler does not allow such things among the interface parts of units, but it is permitted for implementations.

### Listing 9.1    **A valid example of circular referencing**

---

UNIT A;	UNIT B;	UNIT C;
INTERFACE	INTERFACE	INTERFACE
USES C;	USES C;	
TYPE	TYPE	TYPE
A_type = ...	B_type = ....	C_type = ...
IMPLEMENTATION	IMPLEMENTATION	IMPLEMENTATION
USES B;	USES A;	USES A, B;
....	...	...

---

The example in Listing 9.1 is perfectly valid, since both A's and B's interfaces depend on C's, but are independent from one another. Knowing everything that was declared, A's implementation depends on all interfaces, the same is true for B's and C's. For this example, the make utility will ask the compiler to compile Listing 9.1 in the following order:

1. **C's interface is compiled.**

## Compiling and Linking

### *Using the Message Window*

---

2. **B's interface is compiled.**
3. **All of unit A is compiled (unit and implementation),**
4. **B's implementation is compiled.**
5. **C's implementation.**

After an interface compilation, the compiler writes a binary symbol table, containing all the declarations of the interface, in a 'sbmf' resource in the project file. This information is read back when the unit's name is encountered in a USES clause for another compilation. A unit is recompiled only when one of the following conditions occur:

- The source was modified,
- The source is currently open and edited, or,
- A unit on which the source depends was recompiled.

## Saving and Printing the Message Window

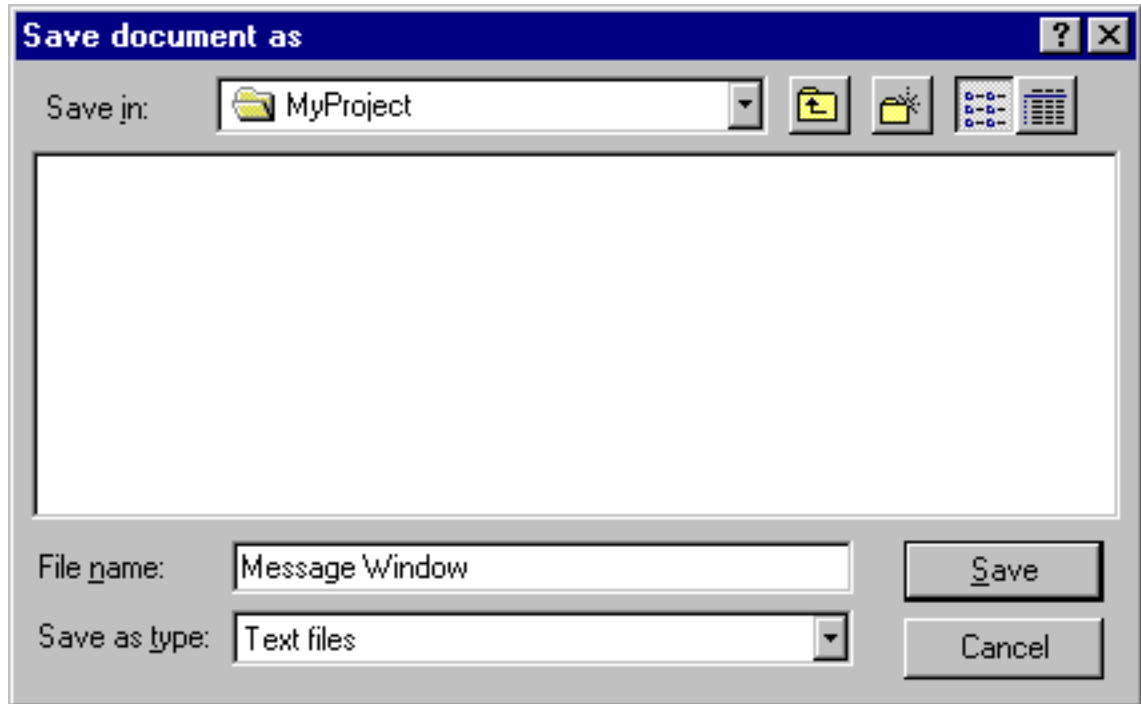
To save or print the contents of the Message window, just follow these steps.

First, make the message window active. To accomplish this, either click on the deactivated message window, or select Errors & Warnings Window from the Window Menu.

Next, select the Save A Copy As or Print command from the File Menu.

The Save A Copy As command will display the following dialog (Figure 9.10)

**Figure 9.10**    **Saving the Message window**



Specify the name and the location, as shown in Figure 9.10. A text file will be saved containing all the errors, warnings, and messages listed in the message window.

If you choose the Print command on the File Menu to print the message window, the print dialog box appears instead. Specify printing options and click OK. All the errors, warnings, and messages will be printed.

## **Locating Errors in Modified Files**

If an error is corrected or the source code is changed, the compiler may not be able to find other errors in the source code file. This may result in an alert telling the user that the position of the error could not be found. When this happens, recompile your project to update the list of errors in the Message window.

## Compiling and Linking

*Using the Message Window*

---



# IDE Menu Reference

---

This chapter describes each command on each CodeWarrior IDE menu.

## IDE Menu Reference Overview

There are several menus in the CodeWarrior IDE menubar:

- File Menu
- Edit Menu
- Search Menu
- Project Menu
- Tools Menu
- Window Menu
- Help Menu

The File Menu, Edit Menu, Search Menu, Project Menu, Tools Menu, Window Menu, and Help Menu are visible at all times.

## File Menu

The File Menu contains commands you use when opening, creating, saving, closing, and printing existing or new source code files and projects. The File Menu also provides a few different methods of saving edited files.

### New

Creates a new editable text file.

To learn more about this command, refer to “Creating a New File” on page 69 for more information.

#### **New Project**

Creates a new project file.

To learn more about this command, refer to “Creating a Project” on page 36.

#### **Open**

Allows you to open an existing text file.

To learn more about this command, refer to “Opening Files with the File Menu” on page 70.

#### **Open Selection**

This menu item is enabled in the menu when text is selected in the active Editor window.

To learn more about this command, refer to “Opening Files with the File Menu” on page 70.

#### **Open File**

This menu item opens a text or project file.

To learn more about this command, refer to “Opening Files with the File Menu” on page 70.

#### **Close**

Closes the frontmost window whether it is the Project window, the Message Window, or a source code window.

See “Closing a File” on page 79 for more information.

#### **Close All**

Closes all open Editor windows.

To learn more about this topic, refer to “Closing All Files” on page 80.



**Switch to Debugger**

Gives control to the debugger. The line the text insertion point is on in the CodeWarrior Editor is displayed by the debugger. This command is dimmed if no source code window is active, or the debugger is not running.

To learn more about this topic, refer to “Debugging a Project” on page 217. You can also refer to the *Metrowerks Debugger Manual*.

**Save**

Saves the contents of the active Editor window to disk.

For more information, see “Saving One File” on page 76.

**Save All**

Saves all Editor files that are currently open.

For more information, see “Saving All Files” on page 76.

**Save As**

Saves the contents of the active window to disk under another name of your choosing.

For more information, see “Renaming and Saving a File” on page 77.

**Save A Copy As**

Saves the active Editor window or Project window in a separate file. This command operates in two different ways, depending on whether a source code file or the Project window is active.

For more information, see “Backing Up Files” on page 78.

**Revert**

Use the Revert command to revert the active Editor window to its last saved version.

To learn more about how to revert to the previous version of a file, refer to “Reverting to a Previously-Saved File” on page 83.

#### **Print Setup**

Sets the options used when printing files from CodeWarrior.

For more information about Print Setup, see “Setting Print Options” on page 81.

#### **Print**

Prints files from CodeWarrior on your printer.

For more information on printing files, see “Printing a Window” on page 82, or read the documentation that came with your printer.

#### **Exit**

Quits CodeWarrior immediately provided one of the following conditions have been met:

- All changes to the open Editor files have already been saved, or
- The open Editor files have not been changed.

If a Project window is open, the Exit command saves all changes to the project file before the environment quits. If an Editor window is open and changes have not been saved, CodeWarrior asks if you want to save the changes before quitting.

## **Edit Menu**

The Edit menu contains all the customary editing commands and some CodeWarrior additions, including the commands that open the Preferences and Project Settings dialogs.

#### **Undo**

The text of this menu command varies depending on the most recent action, and your Editor options settings.

Undo reverses the effect of your last action. The name of the Undo command varies depending on the type of operation you last exe-

cuted. For example, if you have just typed in an open Editor window, the Undo command is renamed Undo Typing. Choosing the Undo Typing command will remove the text you have just typed.

To learn more about this topic, refer to “Undoing the last edit” on page 107, and “Undoing and redoing multiple edits” on page 107.

If you don’t have Use Multiple Undo turned on in the Editor options panel, Undo toggles between Undo and Redo. To learn more about how to configure this option, refer to “Editor Settings Panel” on page 180.

## **Redo**

### **Multiple Undo**

### **Multiple Redo**

Once an operation has been undone, it may be redone. For example, if you select the Undo Typing command, the command is changed to Redo Typing. Choosing this command overrides the undo.

If you have Use Multiple Undo turned on in the Editor Settings Panel, you have more flexibility with regard to Undo and Redo operations. Choose Undo multiple times to undo multiple actions. Choose Redo multiple times to redo multiple actions.

To learn more about undo operations, refer to “Undoing the last edit” on page 107, and “Undoing and redoing multiple edits” on page 107.

To learn about how to configure multiple undo, refer to “Editor Settings Panel” on page 180.

## **Cut**

Deletes the selected text and puts it in the Clipboard, replacing the contents of the Clipboard.

#### **Copy**

Copies the selected text in the active Editor window onto the system Clipboard. If the Message Window is active, the Copy command copies all the text in the Message Window onto the Clipboard.

#### **Paste**

Pastes the contents of the Clipboard into the active Editor window.

The Paste command replaces the selected text with the contents of the Clipboard. If no text is selected, the Clipboard contents are placed after the text insertion point.

If the active window is the Message Window, the Paste command is dimmed and cannot be executed.

#### **Clear**

Deletes the selected text without placing it in the Clipboard. The Clear command is equivalent to pressing the Delete or Backspace key.

#### **Select All**

Selects all the text in the active window. This command is usually used in conjunction with other Edit menu commands such as Cut, Copy, and Clear.

To learn more about selecting text, refer to “Selecting Text” on page 101.

#### **Balance**

Selects the text enclosed in either parentheses (), brackets [], or braces {}. For a complete procedure on how to use this command and how to balance while typing, consult “Balancing Punctuation” on page 106.

**Shift Left**

Shifts the selected source code one tab size to the left. The tab size is specified in the Preferences dialog box.

To learn more about this feature, refer to “Shifting Text Left and Right” on page 106.

**Shift Right**

Shifts the selected source code one tab size to the right.

To learn more about this feature, refer to “Shifting Text Left and Right” on page 106.

**Insert Reference Template**

This command is used in the Macintosh-hosted CodeWarrior product, and should be ignored on Windows.

**Preferences**

Use this command to change the global preferences for the CodeWarrior IDE.

To learn more about configuring preferences, refer to “Choosing Preferences” on page 180.

**Project Settings**

Use this command to change settings for the active project.

To learn more about configuring Project Settings, refer to “Choosing Project Settings” on page 191.

## **Search Menu**

This menu contains all the commands used to find and replace text. There are also some commands for code navigation.

#### **Find**

Opens the Find dialog box which is used to find and/or replace the occurrences of a specific string in one or many files.

To learn more about the Find window and its capabilities, refer to “Guided Tour of the Find Window” on page 115.

#### **Find Next**

Finds the next occurrence of the Find Text Box string in the active window. This is an alternative to clicking the Find button in the Find dialog box.

To learn more about this feature, refer to “Finding Search Text” on page 127.

#### **Find Previous**

Find Previous operates the same way as Find Next, except that it finds the previous occurrence of the Find Text Box string.

To learn more about this feature, refer to “Finding Search Text” on page 127.

#### **Find in Next File**

Finds the next occurrence of the Find Text Box string in the next file listed in the Multi-File Search portion of the Find window (as exposed by the Multi-File Search Disclosure Triangle in the Find window). This is an alternative to using the Find window. If the Multi-File Search Button is not enabled as shown in Figure 6.3 on page 121, this command is dimmed.

To learn more about this feature, refer to “Searching and Replacing Text in Multiple Files” on page 133.

#### **Find in Previous File**

This command operates the same way as Find in Next File. Beginning at the end of the previous file in the file list, it searches for the next occurrence of the Find Text Box string.

To learn more about this feature, refer to “Searching and Replacing Text in Multiple Files” on page 133.

**Enter ‘Find’ String**

This command copies the selected text in the active window into the Find Text Box, making it the search target string. This is an alternative to copying text and pasting it into the Find window.

To learn how to select text, refer to “Selecting Text” on page 101.

**Enter ‘Replace’ String**

This command copies the selected text in the active window into the Replace Text Box, making it the replacement string.

To learn more about replacing text, refer to “Replacing Found Text” on page 130.

**Find Selection**

Finds the next occurrence of the selected text in the active text editor window. If you hold down the Shift key, this command becomes Find Previous Selection.

To learn more about this feature, refer to “Finding Search Text” on page 127.

**Find Previous Selection**

Find Previous Selection finds the previous occurrence of the selected text in the active text editor window.

To learn more about this feature, refer to “Finding Search Text” on page 127.

To learn how to select text, refer to “Selecting Text” on page 101.

#### **Replace**

This command replaces the selected text in the active window with the text string in the Replace Text Box of the Find window. If no text is selected in the active editor window, this command is dimmed.

This command is useful if you wish to replace one instance of a text string without having to open the Find window. For example, say that you have just replaced all the occurrences of the variable “icount” with “jcount”. While scrolling through your source code, you notice one instance of the variable “icount” is misspelled as “icont”. To replace this variable with “jcount”, select “icont” and choose the Replace command from the Search Menu.

To learn more about replacing text, refer to “Replacing Found Text” on page 130.

To learn how to select text, refer to “Selecting Text” on page 101.

#### **Replace & Find Next**

This command replaces the selected text with the text in the Replace Text Box string of the Find window, and then performs a Find Next. If no text is selected in the active editor window and there is no text in the Find Text Box string field of the Find window, this command is dimmed.

To learn more about replacing text, refer to “Replacing Found Text” on page 130.

To learn how to select text, refer to “Selecting Text” on page 101.

#### **Replace & Find Previous**

This command operates the same way as Replace & Find Next except that it performs a Find Previous.

#### **Replace All**

Finds all the occurrences of the Find string and replaces them with the Replace string. If no text is selected in the active editor window



and there is no text in the Find string field in the Find dialog box, this command is dimmed.

### **Find Definition & Reference**

This command searches for the definition of the routine name selected in the active window. Searching starts with the source files belonging to the open project. If no definition is found in the project files, the on-line reference databases are searched.

If no definition is found, a system beep sounds.

This feature may not be implemented in time for the DR1 release of the CodeWarrior Windows tools.

### **Find Reference**

This command searches for the definition of the routine name selected in the active window. Searching will occur only in the on-line databases provided with CodeWarrior.

If no definition is found, a system beep sounds.

This feature may not be implemented in time for the DR1 release of the CodeWarrior Windows tools.

### **Find Definition**

This command searches for the definition of the routine name selected in the active window. Searching is done only on the source files belonging to the open project.

If no definition is found, a system beep sounds.

This feature may not be implemented in time for the DR1 release of the CodeWarrior Windows tools.

### **Go Back**

This command returns you to the next previous view in the Browser.

To learn more about this feature, refer to “Go Back and Go Forward” on page 169.

#### **Go Forward**

This command moves you to the next view in the Browser (after you have used the Go Back command to return to a previous view.

“Go Back and Go Forward” on page 169.

#### **Go To Line**

Opens a dialog (in which you enter the line number) and then moves the text insertion point to the line.

For more information about this feature, refer to “Going to a Particular Line” on page 114.

## **Project Menu**

The Project menu lets you add and remove files and libraries from your project. It also lets you compile, build, and link your project. All of these commands are covered in this section.

#### **Add Window**

This command adds the file in the active Editor window to the open project.

To learn more about this feature, refer to “Using the Add Window Command” on page 57.

#### **Add Files**

This command adds files to the Project window.

To learn more about this feature, refer to “Using the Add Files Command” on page 56.

## **Remove Files**

This command removes the selected files and segments/groups from the project window. Removing all the files from a segment/groups removes the segment/group and renumbers the remaining project segments.

To learn more about removing files from projects, consult “Removing Files and Groups” on page 60.



---

**WARNING!** *This command cannot be undone.*

---

## **Reset File Paths**

This command resets the project’s cache of access paths for all files belonging to the open project. This command is useful if, for example, you move one of the project’s files to a different location on your drive.

## **Synchronize Modification Dates**

This command updates the modification dates stored in the project file. It checks the modification date for each file in the project, and if the file has been modified since it was last compiled, CodeWarrior marks it for recompilation.

To learn more about this topic, refer to “Synchronizing modification dates” on page 63.

## **Check Syntax**

This command checks the syntax of the source code file in the active Editor window or the selected file(s) in the open Project window. If the active Editor window is empty, or no project is open, this command is dimmed.

Check Syntax does not generate object code. This command only checks the source code for syntax errors. The progress of this operation is tracked in the Toolbar’s message area. To abort this command at any time, press the Escape key.

If one or more errors are detected, the Message window appears. For information on how to correct compiler errors, consult “Correcting Compiler Errors and Warnings” on page 238.

### **Preprocess**

This command performs C/C++ preprocessing on selected files.

To learn more about this command, refer to “Preprocessing Source Code (C/C++ only)” on page 226.

### **Precompile**

This command precompiles the source code file in the active Editor window into a precompiled header file.

To learn more about this topic, refer to “Using Precompiled or Preprocessed Headers” on page 220.

### **Compile**

This command compiles selected files. If the project window is active, the selected files and segments/groups are compiled. If a source code file in an Editor window is active, the source code file is compiled. The source code file must be in the open project.

To learn more about this topic, refer to “Compiling and Linking a Project” on page 213.

### **Disassemble**

This command disassembles the compiled source code files selected in the project window, and displays object code in new windows with the title of the source code file and the extension `.dump`.

To learn more about this feature, refer to “Disassembling Source Code” on page 228.

### **Remove Binaries**

This command removes all compiled source code binaries from the open project. The numbers in the Code Column and Data Column of each file are reset to zero.

To learn more about this topic, refer to “Removing Binaries” on page 218.

### **Remove Binaries & Compact**

This command removes all binaries from the project file and compacts it to save disk space.

Compacting the project removes all binary and debugging information stored in the project file, and retains only the information regarding which files belong to the project and project settings.

To learn more about this topic, refer to “Removing Binaries” on page 218.

### **Bring Up To Date**

This command updates the open project by compiling all of its modified and “touched” files.

To learn more about this topic, refer to “Updating a Project” on page 215.

### **Make**

This command builds the selected project by compiling and linking the modified and “touched” files in the open project. The results of a successful build depend on the selected project type.

To learn more about this topic, refer to “Making a Project” on page 215.

#### **Enable Debugger**

This command sets preferences to allow your project to be debugged. With this command, the debugger can be launched to debug your project.

To learn more about this topic, refer to “Controlling Debugging in a Project” on page 64.

#### **Disable Debugger**

Launches your project directly instead of using the debugger. This command does not reset preferences set by the Enable Debugger-command.

This command is dimmed if the active project does not create an application.

To learn more about this topic, refer to “Controlling Debugging in a Project” on page 64.

#### **Run**

This command compiles, links, creates a stand-alone application, and launches that application. If the project type is set as a library or a shared library, then the Run command is dimmed.

To learn more about this topic, refer to “Running a Project” on page 216.

#### **Debug**

This command compiles and links your project and then opens the project’s debugger file with the Metrowerks Debugger. This command runs the Metrowerks Debugger for any project that the debugger can work with.

To learn more about the CodeWarrior Debugger, refer to the *CodeWarrior Debugger Manual*.

## Tools Menu

The Tools menu contains all the commands used to move and customize the Toolbar.

### Show Toolbar

This command shows the IDE Toolbar if it is hidden.

To learn more about the IDE Toolbar, refer to “IDE Toolbar” on page 31.

### Hide Toolbar

If the IDE Toolbar is showing, this command hides it.

To learn more about the IDE Toolbar, refer to “IDE Toolbar” on page 31.

### Anchor Toolbar

Anchors or “floats” the IDE Toolbar depending on the current status of the Toolbar.

By default, the Toolbar is anchored which means that it cannot be moved. If you wish to move the Toolbar, choose the Unanchor Toolbar command. The Toolbar is drawn with a small window title bar and close box, which can be used to move and hide the Toolbar respectively.

To learn more about the IDE Toolbar, refer to “IDE Toolbar” on page 31.

### Unanchor Toolbar

If the Toolbar has been unanchored and moved, the Anchor Toolbar command will move the Toolbar back to its default position directly underneath the menu bar. Subsequently unanchoring the Toolbar moves it back to its last unanchored position.

To learn more about the IDE Toolbar, refer to “IDE Toolbar” on page 31.

#### **Reset Toolbar**

Resets the Toolbar to its default condition.

To learn more about the IDE Toolbar, refer to “IDE Toolbar” on page 31.

#### **Clear Toolbar**

This command removes all command icons from the Toolbar.

To learn more about the IDE Toolbar, refer to “IDE Toolbar” on page 31.

## **Window Menu**

The Window menu includes commands that tile open editor windows, switch between windows, and reopen previously opened projects.

#### **Stack**

This command opens all Editor windows to their full screen size and stacks them one on top of another, with their window titles showing. This command is dimmed when the active window is the Project window or Message window.

#### **Tile**

This command arranges all Editor windows so that none overlap. This command is dimmed when the active window is the Project window or Message window.

#### **Tile Vertical**

This command arranges all the Editor windows vertically.

This command is dimmed when the active window is the Project window or Message window.



**Zoom Window**

This command expands the active window to the largest possible size. If you choose it again, it returns the window to its original size.

**Save Default Window**

This command saves the settings of the active window, so that the next time you open a window of that type, CodeWarrior opens it with the saved settings.

To learn more about this command, refer to ,“Saving Window Settings” on page 97, or “Saving a Default Browser” on page 174.

**Show Catalog Window**

This command displays the Browser Catalog Window. This menu command is dimmed when the Browser is not activated.

To learn more about this feature, refer to “Catalog Window” on page 151. To learn how to activate the Browser, refer to “Activating the Browser” on page 146.

**Show Hierarchy Window**

This command displays the Browser Multi-Class Hierarchy Window. This menu command is dimmed when the Browser is not activated.

To learn more about this feature, refer to “Multi-Class Hierarchy Window” on page 161.

To learn how to activate the Browser, refer to “Activating the Browser” on page 146.

**New Class Browser**

This command displays the Browser’s Multi-Class Browser Window. This menu command is dimmed when the Browser is not activated.

To learn more about this feature, refer to “Multi-Class Browser Window” on page 153.

To learn how to activate the Browser, refer to “Activating the Browser” on page 146.

#### **Project Switch List**

This command allows you to reopen a project that was previously opened then closed. Using this command closes your current project before reopening the newly-selected project.

To learn more about this feature, refer to “Using the Project Switch List” on page 44.

#### **Errors & Warnings Window**

This command opens and brings the Errors and Warnings window to the front.

To learn more about this window, refer to “Guided Tour of the Message Window” on page 229. Also, refer to “Using Batch Searches” on page 132.

#### **Other Window Menu Items**

The other Window Menu items depend solely on which project, source files, header files, and other windows you have open.

All of the open files are shown in this menu and the first nine files (1 through 9) are given key equivalents. The current project is always given the Ctrl-0 (zero) combination. A checkmark is placed beside the file in the active window. A file whose modifications have not been saved is underlined.

A checkmark may also be placed beside the Message window if it is the active window.

To make one of your open CodeWarrior files active and bring its window to the front, you can do one of the following:

- Click in its window.
- Select it in the Window Menu.
- Use the key equivalent shown in the Window Menu.

## Help Menu

The Help menu contains commands to help you look up on-line help information, learn about how to use on-line help, and view the Metrowerks About Box.

### **Contents**

This menu command displays the CodeWarrior help files.

### **Keys**

This menu command displays the topic index for the CodeWarrior on-line help files.

### **How to Use Help**

This menu command displays the on-line help that tells how to use the help facilities.

### **About Metrowerks**

Displays the Metrowerks About Box.

## IDE Menu Reference

### *Help Menu*

---

# Index

---

## Symbols

#pragma precompile\_target 222  
.cwp 42  
.dump 228

## Numerics

68K 17

## A

About Metrowerks command 265  
Add Default button 197  
Add File command 256  
Add Window command 256  
Alert Yourself After Build option 220  
Anchor Toolbar command 261  
Apply Button 179  
Arrow Keys 100  
Automatic updating 222

## B

Background Color 184  
backup files 78  
Balance command 106, 250  
Balance While Typing option 181  
Balancing punctuation 106  
Batch search 132  
BeOS 17, 22  
Bring Up To Date command 220, 259  
browser 145–174  
    activating 146, 200  
    analyzing inheritance 172  
    base classes in hierarchy 163  
    base classes in single-class 160  
    catalog view 147, 151  
    classes pane 156  
    customizing windows 174  
    data members pane 157  
    declaration button 160  
    editing code 172  
    file button 156  
    finding function overrides 173  
    hierarchy view 149

    identifier icon 158  
    interface 150–167  
    lines in hierarchy 162  
    member functions pane 157  
    multi-class 153–158  
    multi-class hierarchy 161–163  
    navigating code with 166  
    opening source file 171  
    orientation button 155  
    resize bar 158  
    saving windows 174  
    seeing declaration 171  
    seeing function definition 171  
    seeing PowerPlant in 173  
    show hierarchy button 160  
    showing data in single-class 160  
    showing subclasses in hierarchy 162  
    single-class 159  
    single-class hierarchy 163  
    source pane 158  
    strategy 146–149  
    symbol window 164  
    synchronized class selection 157, 162  
    using 168–174  
    viewing options 146

browser view 148

## C

C 18  
C++ 18  
C/C++ preprocessor 226  
catalog view 147  
catalog window  
    browser 151  
category pop-up menu, in category window 152  
Check Syntax command 257  
classes pane in browser 156  
Clear command 105, 250  
Clear Toolbar command 262  
Close All command 80, 246  
Close command 246  
code navigation pop-up menu 166  
CodeWarrior IDE 20  
comments, coloring 186

## Index

---

- compacting projects 219
- Compile column 62
- Compile command 258
- Compiler 194
- Compilers 18
- Compiling 211–243
  - project 213
- compiling
  - source files 215
- Compiling and Linking
  - choosing a compiler 212
  - compiling files 213–215
  - debugging 217
  - disassembling 228
  - guided tour 229–234
  - link map 218
  - making a project 215
  - options 219
  - overview 211
  - plugin compilers 212
  - precompiling headers 220–226
  - preprocessing 226
  - removing binaries 218, 219
  - removing object code 219
  - running 216
  - setting file extension 212
  - speeding 220
  - Synchronizing Modification Dates 218
  - touch and untouch 213
- compiling one file 214
- compiling selected files 214
- Contents command 265
- Context Popup Delay option 182
- Copy command 105, 250
- custom keyword, coloring 187, 188
- Cut command 105, 249

## D

- data members pane, in browser 157
- date caching 220
- Debugging configuration 216
- Defining Symbols for C/C++ 224
- Defining Symbols for Pascal 225
- Disable debugging command 260
- Disassemble command 258

- Disassembling source code 228
- documentation
  - viewers 21
- DOS text files 79
- Drag & Drop editing support 183
- Drag and Drop text 102–105
- Dynamic Scroll option 181

## E

- Edit Menu 26, 248–251
  - Balance command 250
  - Clear command 250
  - Copy command 250
  - Cut command 249
  - Insert Reference Template command 251
  - Multiple Redo 249
  - Multiple Undo 249
  - Paste command 250
  - Preferences command 251
  - Project Settings command 251
  - Redo command 249
  - Select All command 250
  - Shift Left command 251
  - Shift Right command 251
  - Undo command 248
- editing
  - in browser 172
- editing, redoing 107
- editing, undoing 107
- Editor 85–114
  - adding text 100
  - balancing punctuation 106
  - color syntax 108
  - configuration 93–98
  - deleting text 100
  - drag and drop 183
  - finding a routine in 109
  - font 93
  - font preferences 183
  - Go Back and Go Forward 114
  - go to line number 114
  - guided tour 85–93
  - markers 109–112
  - moving text 102–105
  - navigating text 108–114
  - opening related file 113

- overview 85
- panes 95–97
- saving window settings 97
- selecting text 101
- text editing 98–108
- text size 93
- undoing changes 107–108
- user interface elements 85
- Enable Debugging command 260
- End key 100
- Enter ‘Find’ String command 126, 253
- Enter ‘Replace’ String command 253
- Enter ‘Find’ String command 253
- Enter ‘Replace’ String command 253
- Entire Word Check Box 130
- Error Button 231
- Error Messages 235
- error messages
  - compiler 236
- Exit command 248
- extension 193

## F

- file button, in browser 156
- File extension 193
- File Menu 26, 245–248
  - Close All command 246
  - Close command 246
  - Exit command 248
  - New command 245
  - New Project command 246
  - Open command 246
  - Open File command 246
  - Open Selection command 246
  - Print command 248
  - Print Setup command 248
  - Revert command 247
  - Save A Copy As command 247
  - Save All command 247
  - Save As command 247
  - Save command 247
  - Switch to Debugger command 247
- file name suffix 193
- File Path Caption 91, 233
- File Privileges Icons 92
- File Sets List 123

- File Sets Pop-up Menu 122
- Files 69–84
  - closing 79–81
  - creating 69
  - opening 113
  - opening existing 70–75
  - overview 69
  - printing 81–82
  - reverting to saved 83
  - saving 75–79
- Files Sets
  - saving 137
- Find Definition & Reference command 255
- Find Definition command 255
- Find in Next File command 252
- Find in Previous File command 252
- Find Next command 126, 252
- Find Previous command 126, 128, 252
- Find Previous Selection command 253
- Find Reference command 255
- Find Selection command 253
- Find Window
  - guided tour 115–125
- finding all implementations of function 164
- Flashing Delay option 182
- Font Preferences 183
- function, finding all 164

## G

- General Magic’s Magic Cap 22
- Generate Make Map File option 200
- Go Back command 255
- Go Forward command 256
- Go To Line command 256
- grep 140–144
- Groups 58–62

## H

- header files
  - opening 75
  - precompiling 220–223
- Help Menu 31, 265
  - About Metrowerks command 265
  - Contents command 265
  - How to Use Help command 265

## Index

---

- Keys command 265
- Hide Toolbar command 261
- hierarchy view
  - browser 149
- Home key 100
- How to Use Help command 265

### I

- IDE 17, 20
  - guided tour 24–32
  - installation 24
  - Menus 25
  - Toolbar 24, 31
- identifier icon in browser 158
- Ignore Case Check Box 129
- Ignored by Make option 195
- Insert Reference Template command 251
- Installation 24
- Integrated Development Environment 17
- Interface Pop-up Menu 87, 233

### J

- Java 17, 18, 22

### K

- Keys command 265
- keywords, coloring 187

### L

- Launchable option 194
- line button in hierarchy browser 162
- Line Number Button 92, 234
- line number, going to 114
- Linkers 18
- Linking 211–243
- list button, in browser
  - browser
  - list button 156

### M

- Mac OS 22
- Macintosh 17
- Magic Cap 17

- Main Text Color 184
- Make 215
- Make command 220, 259
- Marker Pop-up Menu 90
- marker, adding to text 90
- Marking files for compilation 63
- member functions pane, in browser 157
- Message List Pane 232
- Message Window
  - command 264
  - correcting compiler errors 238
  - error and warning messages 235
  - stepping through messages 236
  - using 234–243
- Modification dates, synchronizing 63
- multi-class browser 153–158
- multi-class hierarchy, browser 161–163
- Multi-file searches 121–125
- multiple redo 249
- multiple undo 249
- multiple Undo command 107

### N

- New Class Browser command 263
- New command 245
- New Project command 246
- number, going to line 114

### O

- Open command 246
- Open File command 246
- Open Selection command 74, 246
- opening file with browser 171
- Options 175–209
  - Access Paths 195–199
  - advanced compile options 219
  - browser coloring 190
  - Build Extras 199–200
  - C/C++ Compiler panel 201
  - C/C++ Warnings Panel 202
  - Custom Keywords panel 200
  - Editor settings 180
  - fonts and tabs 184
  - guided tour 176
  - IR Optimizer panel 208



- overview 175
- Pascal Compiler panel 203
- Pascal Warnings panel 204
- preferences 178, 180
- project settings 176, 191–210
- syntax coloring 185–190
- Target options 192
- Win32 resource compiler 209
- x86 Code Generation panel 206
- x86 Linker panel 207
- x86 Project panel 205
- Options Pop-up Menu 90
- orientation button
  - in browser 155
- Others Button 124

## P

- Page Down key 100
- Page Up key 100
- Palm OS 18, 22
- Pane Resize Bar 233
- Pane Splitter Controls 92
- Panes
  - in editor window 95–97
- Pascal 18
- Paste command 105, 250
- PlayStation 18, 22
- Pop-up Menu Disclosure Button 93, 233
- Post Linker option 193
- Post Linkers 18
- PowerTV 22
- PowerTV OS 18
- Precompile command 222, 222–223, 258
- `precompile_target`, `#pragma` 222
- Precompiled 194
- Precompiled headers
  - automatic updating 222
  - creating 221
- precompiled headers
  - automatic updating of 222
- precompiling 220–223
- Preference Panels 18
- Preferences 178
  - choosing 180
- Preferences command 251

- Preprocess command 258
- Preprocessing code 226
- Print command 248
- Print Setup command 248
- Project Headers Check Box 124
- Project Information Caption 232
- Project Menu 28, 256–260
  - Add File command 256
  - Add Window command 256
  - Bring Up To Date command 259
  - Check Syntax command 257
  - Compile command 258
  - Disable debugging command 260
  - Disassemble command 258
  - Enable Debugging command 260
  - Make command 259
  - Precompile command 258
  - Preprocess command 258
  - Remove Binaries & Compact command 259
  - Remove Binaries command 259
  - Remove Files command 257
  - Reset File Paths command 257
  - Run command 260
  - Synchronize      Modification      Dates  
command 257
- Project Settings 176
- Project Settings command 251
- Project stationery
  - creating 41
- Project Switch List 44
- Project Switch List submenu 44, 264
- Project Window
  - guided tour 47–51
  - navigating in 47
  - user interface items 47
- Projector Aware 183
- Projects 35–67
  - adding files 55–58
  - adding preprocessor symbols to 66
  - building 41, 213
  - choosing stationery 37
  - closing 46
  - compiling 213
  - creating 36–43
  - creating groups 59
  - creating stationery for 41
  - debug enabling 216

## Index

---

- debug setup 64–66
- debugging 217
- Expanding and Collapsing Groups 54
- expanding and collapsing groups 54
- groups and segments 51
- Items Saved with 46
- making 215
- managing files in 51–64
- modifying 40
- moving 64
- moving around 47
- moving files and groups 58
- naming 38
- naming groups 61
- new 37
- open command 44
- opening existing 43–45
- opening with Project Switch List 44
- removing files and groups 60
- running 216
- save as type options 46
- saving 45–46
- selecting files and groups 52–54
- settings 191–210
- stationery folder 41
- stationery, about 36
- touching and untouching files 62
- updating 215
- using stationery 37
- projects
  - building 220
  - compacting 219
  - compiling 215, 220

## Q

QuickStart 21

## R

- Recent Strings pop-up menu 127, 131
- recompiling 215
- recompiling files 215
- Redo command 107, 249
- regular expressions 140–144
- Release notes 19
- Remove a file set command 138
- Remove Binaries & Compact command 259

- Remove Binaries command 259
- Remove Files command 61, 257
- Replace & Find Next command 254
- Replace & Find Previous command 254
- Replace All command 130, 254
- Replace and Find Previous command 131
- Replace command 254
- replacing
  - in multiple files 133–140
- Reset File Paths command 257
- Reset Toolbar command 262
- resize bar, in browser 158
- Resource file option 194
- Revert command 247
- Routine Pop-up Menu 109, 233
- routine pop-Up menu 88
- Run command 260

## S

- Save A Copy As command 247
- Save All Before "Update" option 182
- Save All command 247
- Save As command 77, 247
- Save command 247
- Save this File Set command 137
- Search Menu 27, 251–256
  - Enter 'Find' String command 253
  - Enter 'Replace' String command 253
  - Find command 252
  - Find Definition & Reference 255
  - Find Definition command 255
  - Find in Next File command 252
  - Find in Previous File command 252
  - Find Next command 252
  - Find Previous command 252
  - Find Previous Selection command 253
  - Find Reference command 255
  - Find Selection command 253
  - Go Back command 255
  - Go Forward command 256
  - Go To Line command 256
  - Replace & Find Next command 254
  - Replace & Find Previous command 254
  - Replace All command 254
  - Replace command 254

---

Search Menu Find Selection command 253  
searching 125  
    for selection 126  
    multi-file 121–125  
    multiple files 133–140  
    selected text 125–126  
Select All command 250  
selected text search 125  
selection  
    finding 126  
Selection Position 183  
Shift Left command 106, 251  
Shift Right command 106, 251  
Show Catalog Window command 263  
Show Toolbar command 261  
single-class browser 159  
single-class hierarchy, in browser 163  
Sort Function Pop-Up 183  
Source Code Disclosure Triangle 232  
Source Code Pane 233  
source files  
    compiling 215  
    precompiling 220–223  
source pane, in browser 158  
Sources Check Box 124  
Stack command 262  
Stepping Buttons 232  
Stop at EOF option 123, 124, 139  
Store Analysis Results option 200  
Switch to Debugger command 247  
symbol window, in browser 164  
Synchronize Modification Dates command 257  
Synchronizing modification dates 63  
Syntax coloring, table of 186  
System Headers Check Box 124  
System Include Path Pane 196  
System Requirements 23

## **T**

Target option 193  
Targets  
    documentation 21  
    IDE 22  
Text  
    drag and drop of 102–105

Text Editing Area 87  
text replace  
    multiple file 133–140  
    Replace All 130  
    replacing found text 130  
    selective replace 130  
    single file 126–133  
text search 115–144  
    activating multi-file 133  
    Batch search 132  
    choosing file sets 137  
    choosing files 134  
    controlling range 129  
    controlling search parameters 129  
    controlling search range 139  
    finding selection 126  
    finding text 127  
    for selection 126  
    multi-file 121–125  
    multiple file 133–140  
    overview 115  
    regular expressions 140–144  
    removing file sets 138  
    saving file sets 137  
    selected text search 125  
    single file 126–133  
Tile command 262  
Tile Vertical command 262  
Tools Menu 29, 261–262  
    Anchor Toolbar command 261  
    Clear Toolbar command 262  
    Hide Toolbar command 261  
    Reset Toolbar command 262  
    Show Toolbar command 261  
    Unanchor Toolbar command 261  
Touch command 62  
Treat #include as #include "..." option 196  
Turbo Pascal 18  
Tutorial Resources 21

## **U**

Unanchor Toolbar command 261  
Undo command 248, 249  
UNIX text files 79  
updating projects 215, 220  
Use Modification Dates Caching option 200

## Index

---

Use Multiple Undo option 182  
User Include Path Pane 196

### V

viewers 21

### W

Warning Button 232  
Warning Messages 235  
wildcard searching 140–144

Win32/x86 17, 22  
Window Menu 30, 262–264  
    New Class Browser command 263  
    Project Switch List submenu 264  
    Show Catalog Window command 263  
    Stack command 262  
    Tile command 262  
    Tile Vertical command 262  
Window Position and Size 183  
Windows 95 17, 23  
Windows NT 17, 23

# About CodeWarrior Documentation

*Information about the people who worked on this  
documentation and references to other documentation  
you'll find useful.*

## About CodeWarrior IDE User's Guide

**writing lead:** "BitHead" Magnuson

**other writing:** Jeff Mattson, Sarah Markey, Jim Trudeau

**engineering:** Berardino Baratta, Kevin Bell, Jesse  
Donaldson, Matt Henderson, Glenn  
Meter, Dan Podwall, Dieter Shirley, Cam  
Vien, Bob Kushlis, Eric Cloninger,  
Andrew Southwick, HongGang Zhang,  
Kurt Ostfeld, Joel Sumner

**frontline warriors:** John Roseborough, Marc Paquette



## Guide to Other CodeWarrior Documentation

<b>If you need information about...</b>	<b>See this</b>
<i>Installing updates to CodeWarrior</i>	<i>QuickStart Guide</i>
<i>Getting started using CodeWarrior</i>	<i>QuickStart Guide;</i> <i>Tutorials (Apple Guide)</i>
<i>Using CodeWarrior IDE (Integrated Development Environment)</i>	<i>IDE User's Guide</i>
<i>Important, last minute, information on new features and changes</i>	<i>Release Notes folder</i>
<i>Creating Macintosh and Power Macintosh software</i>	<i>Targeting Mac OS;</i> <i>Mac OS folder</i>
<i>Creating Microsoft Win32/x86 software</i>	<i>Targeting Win32;</i> <i>Win32/x86 folder</i>
<i>Creating Magic Cap software</i>	<i>Targeting Magic Cap;</i> <i>Magic Cap folder</i>
<i>Using the ToolServer with the CodeWarrior editor</i>	<i>Targeting Mac OS</i>
<i>Controlling CodeWarrior through AppleScript</i>	<i>IDE User's Guide</i>
<i>Using CodeWarrior to program in MPW</i>	<i>Command Line Tools Manual</i>
<i>C, C++, or 68K assembly language programming</i>	<i>C, C++, and Assembly Reference;</i> <i>MSL C Reference;</i> <i>MSL C++ Reference</i>
<i>Pascal or Object Pascal programming</i>	<i>Pascal Language Manual;</i> <i>Pascal Library Reference</i>
<i>Fixing compiler and linker errors</i>	<i>Errors Reference</i>
<i>Debugging</i>	<i>Debugger Manual</i>
<i>Fixing memory bugs</i>	<i>ZoneRanger Manual</i>
<i>Speeding up your programs</i>	<i>Profiler Manual</i>
<i>PowerPlant</i>	<i>The PowerPlant Book;</i> <i>PowerPlant Advanced Topics;</i> <i>PowerPlant reference documents</i>
<i>Creating a PowerPlant visual interface</i>	<i>Constructor Manual</i>
<i>Learning how to program for the Mac OS</i>	<i>Discover Programming for Macin-</i> <i>tosh</i>
<i>Learning how to program in Java</i>	<i>Discover Programming for Java</i>
<i>Contacting Metrowerks about registration, sales, and licensing</i>	<i>Quick Start Guide</i>
<i>Contacting Metrowerks about problems and suggestions using CodeWarrior software</i>	<i>email Report Forms in the Release</i> <i>Notes folder</i>
<i>Sample programs and examples</i>	<i>CodeWarrior Examples folder;</i> <i>The PowerPlant Book;</i> <i>PowerPlant Advanced Topics;</i> <i>Tutorials (Apple Guide)</i>
<i>Problems other CodeWarrior users have solved</i>	<i>Internet newsgroup [docs] folder</i>
<i>Getting more information from CodeWarrior documentation</i>	<i>CodeWarrior Cross-Reference</i>

IDE 96/11/18