

# CodeWarrior® IDE User Guide



Because of last-minute changes to CodeWarrior, some of the information in this manual may be inaccurate. Please read the Release Notes on the CodeWarrior CD for the latest up-to-date information.

**Metrowerks CodeWarrior copyright ©1993–1997 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>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>CompuServe</b>	goto Metrowerks

---

# Table of Contents

---

<b>1 Introduction</b>	<b>19</b>
Introduction	19
Read the Release Notes!	21
Typographical Conventions	21
Notes, Warnings, Tips, and Beginner's Hints	21
Typeface Conventions	21
IDE User Guide Overview	22
About the CodeWarrior IDE	23
Where to Go From Here	23
QuickStart and Tutorial Resources	24
Operating System Targeting Documentation	24
What's New in This Release	25
Threaded Execution	26
Multiple Open Projects	27
Multiple Targets Per Project	27
Subprojects	28
Mac OS Merge Linker	29
New Toolbars	30
Customizable User Interface Key Bindings	30
Improved Drag and Drop Support	30
Improved Project File Management	30
<b>2 Getting Started</b>	<b>31</b>
Getting Started Overview	31
System Requirements	31
CodeWarrior IDE Installation	32
Programming Concepts	32
Creating Input Files	33
Source Code File	33
Resource File	33
Interface or Header File	33
Project File	34
Generating the Software	34
Compilers	34

## Table of Contents

---

Assemblers . . . . .	34
Linkers . . . . .	34
Output Files (Product) . . . . .	34
Debugging and Refining . . . . .	35
CodeWarrior IDE Guided Tour . . . . .	35
Initial View of the IDE . . . . .	35
IDE Menus . . . . .	36
File Menu . . . . .	37
Edit Menu . . . . .	38
Search Menu . . . . .	39
Project Menu . . . . .	40
Window Menu . . . . .	41
Help Menu . . . . .	42
Global Toolbar . . . . .	42
Global Toolbar Overview . . . . .	42
Customizing Toolbar Commands . . . . .	43
Customizing the Global Toolbar . . . . .	43
Other IDE Components . . . . .	43
<b>3 Working with Projects . . . . .</b>	<b>45</b>
Projects Overview . . . . .	45
Creating a Simple Project . . . . .	46
About Project Stationery . . . . .	46
Using Stationery Projects . . . . .	47
Choosing the New Project Stationery File . . . . .	47
Naming Your New Project . . . . .	49
Modifying Your New Project . . . . .	51
Building Your New Project . . . . .	52
About the Project Stationery Folder . . . . .	52
Creating Your Own Project Stationery . . . . .	52
Opening an Existing Project . . . . .	55
Using the Open Command . . . . .	55
Using the Open Recent Command . . . . .	56
Using the Project Window to Open Subprojects. . . . .	56
Saving a Project . . . . .	56
Items Saved with Your Project . . . . .	57

## Table of Contents

---

Saving a Copy of Your Project . . . . .	57
Closing a Project . . . . .	57
Choosing a Default Project . . . . .	58
Guided Tour of the Project Window . . . . .	58
Navigating the Project Window . . . . .	59
Project Window User Interface Items . . . . .	59
Category Tabs . . . . .	60
Toolbar . . . . .	61
Group Organization . . . . .	61
File Column . . . . .	62
Code Column . . . . .	62
Data Column . . . . .	63
Debug Column . . . . .	63
Touch Column . . . . .	63
Interfaces Pop-up . . . . .	64
Group Pop-up . . . . .	64
Checkout Status Column . . . . .	64
Managing Files in a Project . . . . .	65
About Groups and Segments . . . . .	65
Expanding and Collapsing Groups . . . . .	66
Selecting Files and Groups . . . . .	67
Selection by mouse-clicking . . . . .	68
Selection by keyboard . . . . .	69
Adding Files . . . . .	69
Where Files Appear . . . . .	70
Using the Add Files Command . . . . .	70
Using Drag and Drop . . . . .	72
Using the Add Window Command . . . . .	73
Moving Files and Groups . . . . .	73
Creating Groups . . . . .	74
Removing Files and Groups . . . . .	75
Renaming Groups . . . . .	76
Touching and Untouching Files . . . . .	76
Synchronizing modification dates . . . . .	79
Creating Complex Projects . . . . .	79
What is a Target? . . . . .	80

## Table of Contents

---

What is a Subproject? . . . . .	81
Strategy for Creating Complex Projects . . . . .	83
Creating a New Target . . . . .	84
Changing a Target Name . . . . .	87
Changing the Target Settings. . . . .	87
Setting the Current Build Target . . . . .	88
Creating Target Dependencies . . . . .	88
Assigning Files to Targets . . . . .	91
Creating Subprojects Within Projects . . . . .	93
Examining Project Information . . . . .	94
Moving a Project . . . . .	96
Controlling Debugging in a Project . . . . .	96
Activating Debugging for a Project . . . . .	97
Activating Debugging for a File . . . . .	97
Debug Info Marker for Groups . . . . .	98
Adding Preprocessor Symbols to a Project . . . . .	99

## **4 Working with Files . . . . . 101**

Working with Files Overview . . . . .	101
Creating a New File. . . . .	101
Opening an Existing File. . . . .	102
Opening Files with the File Menu . . . . .	102
Project File . . . . .	102
Text File . . . . .	103
Opening Files from the Project Window . . . . .	104
File Column. . . . .	104
Group Pop-up Menu . . . . .	104
Interfaces Pop-up Menu . . . . .	105
Opening Files from an Editor Window . . . . .	106
Opening a Related File . . . . .	107
Saving a File . . . . .	107
Saving One File . . . . .	108
Saving Files Automatically . . . . .	109
Renaming and Saving a File . . . . .	109
Backing Up Files. . . . .	110
Saving as a UNIX or DOS text file . . . . .	111

Closing a File . . . . .	112
Closing One File . . . . .	112
Closing All Files . . . . .	113
Printing a File . . . . .	113
Setting Print Options . . . . .	114
Printing a Window . . . . .	114
Reverting to a Previously-Saved File . . . . .	115
<b>5 Editing Source Code . . . . .</b>	<b>117</b>
Source Code Editor Overview . . . . .	117
Guided Tour of the Editor Window . . . . .	117
Text Editing Area. . . . .	119
Interface Pop-Up Menu . . . . .	119
Routine Pop-Up Menu . . . . .	121
Marker Pop-Up Menu . . . . .	122
Options Pop-Up Menu . . . . .	123
File Path Caption. . . . .	124
Permissions Pop-up Menu. . . . .	125
Line Number Button . . . . .	125
Pane Splitter Controls. . . . .	126
Pop-Up Menu Disclosure Button . . . . .	126
Dirty File Marker. . . . .	126
Editor Window Configuration . . . . .	126
Setting Text Size and Font . . . . .	127
Seeing Window Controls . . . . .	127
Splitting the Window into Panes . . . . .	129
Creating a new pane . . . . .	130
Resizing a pane . . . . .	131
Removing a pane . . . . .	131
Saving Window Settings . . . . .	131
Customizing the Toolbar . . . . .	132
Basic Text Editing. . . . .	132
Basic Editor Window Navigation. . . . .	132
Scroll bar navigation . . . . .	133
Keyboard navigation . . . . .	133
Adding Text . . . . .	134

## Table of Contents

---

Deleting Text . . . . .	135
Selecting Text . . . . .	135
Moving Text (Drag and Drop) . . . . .	136
Using Cut, Copy, Paste, and Clear . . . . .	140
Balancing Punctuation . . . . .	141
Using automatic balancing . . . . .	141
Shifting Text Left and Right . . . . .	141
Undoing Changes . . . . .	142
Undoing the last edit . . . . .	142
Undoing and redoing multiple edits . . . . .	142
Reverting to the last saved version of a file . . . . .	143
Controlling Color . . . . .	143
Navigating in Text . . . . .	143
Finding a Routine . . . . .	144
Adding, Removing, and Selecting a Marker . . . . .	144
Adding a Marker . . . . .	144
Removing a Marker . . . . .	147
Jumping to a Marker . . . . .	147
Opening a Related File . . . . .	147
Going to a Particular Line . . . . .	149
Using Go Back and Go Forward . . . . .	149
Configuring Editor Commands . . . . .	150

## **6 Searching and Replacing Text . . . . . 151**

Searching and Replacing Text Overview . . . . .	151
Guided Tour of the Find Dialog Box. . . . .	151
Search and Replace Section . . . . .	152
Find Text Box . . . . .	153
Replace Text Box. . . . .	153
Recent Strings Pop-Up Menu . . . . .	154
Find Button . . . . .	154
Replace Button . . . . .	155
Replace & Find Button . . . . .	155
Replace All Button . . . . .	155
Batch Checkbox . . . . .	155
Wrap Check Box . . . . .	156



---

Ignore Case Check Box . . . . .	156
Entire Word Check Box . . . . .	156
Regexp Check Box . . . . .	156
Multi-File Search Disclosure Triangle . . . . .	157
Multi-File Search Button . . . . .	157
Multi-File Search Section . . . . .	158
File Sets Pop-Up Menu . . . . .	159
File Sets List. . . . .	160
Project Pop-up Menu . . . . .	160
Stop at End of File Check Box . . . . .	161
Sources Check Box . . . . .	162
System Headers Check Box . . . . .	162
Project Headers Check Box . . . . .	162
Others Button . . . . .	162
Searching for Selected Text. . . . .	163
Finding text in the active editor window . . . . .	163
Finding text in another window . . . . .	164
Searching and Replacing Text in a Single File . . . . .	164
Finding Search Text. . . . .	165
Controlling Search Range . . . . .	166
Controlling Search Parameters . . . . .	167
Ignore Case Check Box . . . . .	167
Entire Word Check Box . . . . .	167
Replacing Found Text. . . . .	168
Replace All . . . . .	168
Selective Replace. . . . .	168
Using Batch Searches . . . . .	169
Searching and Replacing Text in Multiple Files . . . . .	171
Activating Multi-File Search . . . . .	171
Choosing Files to be Searched . . . . .	172
Adding project source files . . . . .	172
Adding project header files . . . . .	173
Adding system header files . . . . .	173
Adding and removing arbitrary files . . . . .	174
Choosing a file set . . . . .	175
Saving a File Set . . . . .	176

## Table of Contents

---

Removing a File Set. . . . .	177
Controlling Search Range . . . . .	177
Using Regular Expressions (grep). . . . .	178
Matching simple expressions . . . . .	179
Matching any character . . . . .	179
Repeating expressions . . . . .	179
Grouping expressions . . . . .	180
Choosing one character from many. . . . .	180
Matching the beginning or end of a line. . . . .	181
Using the Find string in the Replace string . . . . .	181
Remembering sub-expressions. . . . .	182
<b>7 Browsing Source Code . . . . .</b>	<b>183</b>
Browser Overview . . . . .	183
Activating the Browser . . . . .	184
Understanding the Browser Strategy . . . . .	184
Catalog View . . . . .	185
Browser View . . . . .	186
Hierarchy View . . . . .	187
Guided Tour of the Browser . . . . .	188
Catalog Window . . . . .	189
Category pop-up menu. . . . .	190
Symbols pane . . . . .	191
Multi-Class Browser Window . . . . .	191
Orientation button . . . . .	193
List button . . . . .	194
Open File button. . . . .	194
Classes pane . . . . .	195
Member Functions pane . . . . .	195
Data Members pane . . . . .	195
Source pane . . . . .	196
Resize bar. . . . .	196
Identifier icon . . . . .	196
Single-Class Browser Window . . . . .	197
Bases text field. . . . .	199
Show checkboxes . . . . .	199

Show Declaration button . . . . .	199
Show Hierarchy button . . . . .	199
Multi-Class Hierarchy Window . . . . .	199
Line button . . . . .	201
Hierarchy expansion triangle . . . . .	201
Ancestor Class pop-up menu . . . . .	202
Single-Class Hierarchy Window . . . . .	202
Symbol Window . . . . .	204
Routine Symbols pane . . . . .	205
Context Pop-Up Menu . . . . .	205
Using the Browser . . . . .	207
Setting Browser Options . . . . .	207
Navigating Code in the Browser . . . . .	208
Using the Context Pop-Up Menu. . . . .	208
Go Back and Go Forward . . . . .	208
Opening a Source File. . . . .	210
Seeing a Declaration . . . . .	211
Seeing a Routine Definition . . . . .	211
Editing Code in the Browser . . . . .	212
Analyzing Inheritance . . . . .	212
Finding Functions That Are Overrides . . . . .	212
Seeing MFC Classes . . . . .	213
Saving a Default Browser . . . . .	213
Customizing the Browser . . . . .	214
<b>8 Configuring IDE Options . . . . .</b>	<b>215</b>
Configuring IDE Options Overview . . . . .	215
Option Dialog Boxes Guided Tour . . . . .	216
Options Panels. . . . .	216
Dialog Box Buttons . . . . .	218
Discarding Changes . . . . .	219
Factory Settings Button . . . . .	219
Revert Panel Button . . . . .	219
Save Button . . . . .	220
Choosing Preferences . . . . .	220
Editor Preferences . . . . .	220

## Table of Contents

---

General Preferences . . . . .	221
Editor Settings . . . . .	221
Dynamic Scroll . . . . .	222
Balance While Typing . . . . .	222
Save All Before “Update” . . . . .	223
Relaxed C Popup Parsing . . . . .	223
Use Multiple Undo. . . . .	224
Flashing Delay . . . . .	224
Context Popup Delay. . . . .	224
Drag and Drop Editing . . . . .	225
Sort Function Popup . . . . .	225
Main Text Color . . . . .	225
Background Color . . . . .	225
Fonts and Tabs . . . . .	225
Syntax Coloring . . . . .	226
Changing syntax highlighting colors . . . . .	227
Controlling syntax highlighting within a window. . . . .	228
Using color for custom keywords . . . . .	228
Importing or exporting custom keywords . . . . .	230
Browser Coloring . . . . .	230
Enable Automatic Toolbar Help . . . . .	231
Key Bindings . . . . .	231
Restrictions for Choosing Key Bindings. . . . .	232
What is a Prefix Key?. . . . .	233
What is a Quote Key?. . . . .	233
Modifying Key Bindings . . . . .	233
Setting the Prefix Key Timeout. . . . .	236
Exporting Key Bindings . . . . .	237
Importing Key Bindings . . . . .	237
Configuring Misc Bindings . . . . .	238
Configuring Editor Commands Bindings . . . . .	239
Configuring Prefix Keys Bindings . . . . .	243
Configuring the Quote Key Binding . . . . .	243
Choosing Target Settings . . . . .	244
Editor . . . . .	244
Target . . . . .	244

## Table of Contents

---

Language Settings . . . . .	244
Code Generation. . . . .	245
Linker . . . . .	245
Custom Keywords . . . . .	245
68K Target. . . . .	246
Access Paths. . . . .	247
Always Search User Paths. . . . .	248
User Include Path Pane . . . . .	248
System Include Path Pane. . . . .	248
Add Default. . . . .	249
Add . . . . .	249
Change. . . . .	250
Remove . . . . .	251
Build Extras . . . . .	251
Use Modification Date Caching . . . . .	251
Activate Browser . . . . .	252
Java Project . . . . .	252
PPC Target . . . . .	253
File Mappings . . . . .	253
File Mappings List . . . . .	254
Extension . . . . .	255
Compiler . . . . .	255
Precompiled . . . . .	255
Launchable . . . . .	255
Resource File . . . . .	255
Ignored by Make. . . . .	255
Target Settings . . . . .	255
Linker and Post Linker . . . . .	256
Target Name . . . . .	256
Output Directory . . . . .	256
x86 Target . . . . .	257
C/C++ Compiler. . . . .	258
C/C++ Warnings. . . . .	259
Pascal Compiler . . . . .	260
Pascal Warnings . . . . .	260
PPCAsm . . . . .	261

## Table of Contents

---

Rez . . . . .	261
WinRC Resource Compiler . . . . .	261
68K Processor . . . . .	261
68K Disassembler . . . . .	262
IR Optimizer . . . . .	263
Java VM . . . . .	264
PPC Processor . . . . .	264
PPC Disassembler . . . . .	264
x86 CodeGen . . . . .	264
68K Linker . . . . .	265
CFM68K Linker . . . . .	266
Java Linker . . . . .	267
PPC Linker . . . . .	267
PPC PEF . . . . .	268
x86 Linker. . . . .	268
Toolbar Customization . . . . .	269

## **9 Compiling and Linking . . . . . 271**

Compiling and Linking Overview . . . . .	271
Choosing a Compiler . . . . .	272
Understanding Plugin Compilers. . . . .	272
Setting a File Extension . . . . .	273
Compiling and Linking a Project . . . . .	273
Touching and Untouching Files . . . . .	274
Compiling Files . . . . .	274
Compiling One File . . . . .	275
Compiling Selected Files . . . . .	275
Recompiling Files . . . . .	276
Updating a Project . . . . .	276
Making a Project . . . . .	276
Enabling Debugging . . . . .	277
Running a Project . . . . .	278
Debugging a Project . . . . .	278
Generating a Link Map . . . . .	279
Synchronizing Modification Dates . . . . .	279
Removing Objects . . . . .	280

## Table of Contents

---

Removing Object Code . . . . .	280
Advanced Compile Options . . . . .	280
Alerting Yourself After a Build. . . . .	281
Speeding Up a Build by Avoiding Date Checks . . . . .	281
Using Precompiled or Preprocessed Headers . . . . .	281
Creating Precompiled Headers. . . . .	282
Precompile command . . . . .	283
Automatic updating . . . . .	284
Defining Symbols For C/C++ . . . . .	285
Defining Symbols For Pascal. . . . .	286
Preprocessing Source Code. . . . .	287
Disassembling Source Code . . . . .	289
Guided Tour of the Message Window . . . . .	290
Error Button . . . . .	292
Warning Button . . . . .	293
Project Information Caption . . . . .	293
Stepping Buttons. . . . .	293
Message List Pane . . . . .	293
Source Code Disclosure Triangle . . . . .	294
Source Code Pane . . . . .	294
Pane Resize Bar . . . . .	294
Pop-Up Menu Disclosure Button . . . . .	294
Interface Pop-Up Menu . . . . .	294
Routine Pop-Up Menu . . . . .	294
File Path Caption. . . . .	295
Line Number Button . . . . .	295
Using the Message Window . . . . .	295
Seeing Errors and Warnings . . . . .	296
Stepping Through Messages . . . . .	297
Correcting Compiler Errors and Warnings. . . . .	299
Correcting Errors in the Source Code Pane . . . . .	299
Opening the File for the Corresponding Message. . . . .	300
Correcting Linker Errors . . . . .	300
Viewing Linker Errors . . . . .	301
Why Linker Errors Occur . . . . .	301
Correcting Pascal Circular References . . . . .	301

## Table of Contents

---

Saving and Printing the Message Window . . . . .	303
Locating Errors in Modified Files . . . . .	304
<b>10 Configuring Version Control Software . . . . .</b>	<b>305</b>
Version Control System Overview . . . . .	305
Using Source Code Control with Files . . . . .	305
<b>11 IDE Menu Reference . . . . .</b>	<b>307</b>
IDE Menu Reference Overview . . . . .	307
File Menu . . . . .	307
New . . . . .	308
New Project . . . . .	308
Open . . . . .	309
Open Recent . . . . .	309
Open File . . . . .	309
Open Selection . . . . .	309
Close . . . . .	309
Switch to MW Debugger . . . . .	310
Save . . . . .	310
Save As . . . . .	310
Save A Copy As . . . . .	310
Revert . . . . .	310
Print Setup . . . . .	311
Print . . . . .	311
Exit . . . . .	311
Edit Menu . . . . .	312
Undo . . . . .	312
Redo, Multiple Undo, and Multiple Redo . . . . .	313
Cut . . . . .	313
Copy . . . . .	313
Paste . . . . .	313
Clear . . . . .	314
Select All . . . . .	314
Balance . . . . .	314
Shift Left . . . . .	314
Shift Right . . . . .	314



## Table of Contents

---

Insert Reference Template . . . . .	315
Preferences . . . . .	315
Target Settings. . . . .	315
Search Menu . . . . .	315
Find . . . . .	316
Find Next. . . . .	317
Find Previous . . . . .	317
Find in Next File. . . . .	317
Enter 'Find' String . . . . .	317
Enter 'Replace' String. . . . .	317
Find Selection . . . . .	318
Replace. . . . .	318
Replace & Find Next . . . . .	318
Replace All . . . . .	319
Find Definition . . . . .	319
Find Definition and Reference . . . . .	319
Go Back . . . . .	319
Go Forward . . . . .	319
Go To Line . . . . .	320
Project Menu. . . . .	321
Add Window . . . . .	322
Add Files . . . . .	322
Create New Group . . . . .	322
Create New Target . . . . .	322
Create New Segment . . . . .	322
Remove Selected Items . . . . .	323
Check Syntax . . . . .	323
Preprocess . . . . .	323
Precompile . . . . .	323
Compile . . . . .	324
Disassemble. . . . .	324
Bring Up To Date . . . . .	324
Make. . . . .	324
Remove Object Code . . . . .	325
Reset File Paths . . . . .	325
Synchronize Modification Dates . . . . .	325

## Table of Contents

---

Enable Debugger . . . . .	325
Disable Debugger . . . . .	325
Run . . . . .	326
Debug . . . . .	326
Set Default Project . . . . .	326
Set Current Target . . . . .	326
Window Menu . . . . .	326
Stack . . . . .	327
Tile . . . . .	328
Tile Vertical . . . . .	328
Zoom Window . . . . .	328
Save Default Window . . . . .	328
Toolbar . . . . .	328
Show Catalog Window . . . . .	328
Show Hierarchy Window . . . . .	329
New Class Browser . . . . .	329
Build Progress Window . . . . .	329
Errors & Warnings Window . . . . .	329
Project Inspector . . . . .	329
Other Window Menu Items . . . . .	330
Toolbar Submenu . . . . .	330
Toolbar Elements Window . . . . .	331
Show Window Toolbar and Hide Window Toolbar . . . . .	331
Reset Window Toolbar . . . . .	332
Clear Window Toolbar . . . . .	332
Show Global Toolbar and Hide Global Toolbar . . . . .	332
Reset Floating Toolbar . . . . .	332
Clear Floating Toolbar . . . . .	332
Help Menu . . . . .	333
Contents . . . . .	333
Keys . . . . .	333
How to Use Help . . . . .	333
About Metrowerks . . . . .	333

<b>Index . . . . .</b>	<b>335</b>
------------------------	------------



# Introduction

---

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

## Introduction

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

- [Read the Release Notes!](#)
- [Typographical Conventions](#)
- [IDE User Guide Overview](#)
- [About the CodeWarrior IDE](#)
- [Where to Go From Here](#)
- [What's New in This Release](#)

The IDE is a collection of development tools for creating and generating code for the systems listed in [Table 1.1](#). To learn which manual to read for a given target that you are interested in, refer to [Table 1.2 on page 25](#).

**Table 1.1 CodeWarrior IDE Targets**

Target	Description
68K Mac OS	Any Mac OS computer that uses the Motorola 680x0 family of microprocessors
Power Mac OS	Any Mac OS computer that uses a PowerPC microprocessor

## Introduction

### Introduction

---

Target	Description
Win32/x86	The Win32 model for Windows NT 4.0 and Windows 95 on 80x86-class processors
Java	The Java virtual machine
BeOS	The Be operating system
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
OS-9	A real-time operating system for software running on embedded systems

---

**NOTE:** Your version of the CodeWarrior product 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 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.
- Assembly Language, a low-level machine code language that the processor understands.

---

**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 IDE 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 the CodeWarrior CDs, in the Release Notes folder.

## Typographical Conventions

When reading the manual, there are a few style rules that are observed, in order to make the format of the manual consistent.

### Notes, Warnings, Tips, and Beginner's Hints

An advisory statement or **NOTE** may restate an important fact, or call your attention to a fact which may not be obvious.

A **WARNING** given in the text may call attention to something such as an operation that, if performed, could be irreversible, or flag a possible error that may occur.

A **TIP** can help you become more productive with the CodeWarrior IDE. Impress your friends with your knowledge of little-known facts that can only be learned by actually reading the fabulous manual!

A **For Beginners** note may help you better understand the terminology or concepts if you are new to programming.

### Typeface Conventions

If you see some text that suddenly appears in a different typeface (as the word *different* does in this sentence), you are reading

“Computer Voice” text. This is used to denote file or directory names, code, keyboard input, or other items that you see on the computer’s hard disk.

## 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 IDE Project Window, shows how to setup, configure, and work with projects
- [Working with Files Overview](#)— introduces the concepts behind working with files in the CodeWarrior IDE
- [Source Code Editor Overview](#)— explains how to use the CodeWarrior IDE text editor
- [Searching and Replacing Text Overview](#)— explains how to use the CodeWarrior IDE 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 the CodeWarrior IDE’s Preferences and Project Settings dialog boxes
- [Compiling and Linking Overview](#)— discusses how to control compilation, linking, and running a CodeWarrior project
- [Version Control System Overview](#)—discusses how to use revision control systems with 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 the CodeWarrior IDE. 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

If you are an experienced CodeWarrior IDE user, review [“What’s New in This Release” on page 25](#) for an overview of the new features in this release.

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

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 24](#).

## Introduction

### Where to Go From Here

---

To get started quickly with a new target operating system, see [“Operating System Targeting Documentation” on page 24](#).

The following sections will give you a better idea of how to get help for your next step with the CodeWarrior product.

- [QuickStart and Tutorial Resources](#)
- [Operating System Targeting Documentation](#)

## 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 the CodeWarrior IDE, check out these resources:

- The *CodeWarrior QuickStart Guide* for an overview of the CodeWarrior IDE, 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 35](#) provides a quick overview of the CodeWarrior IDE user interface.
- The CW Tutorials Code folder on the CodeWarrior CD contains some sample projects that will help you become productive quickly with the CodeWarrior IDE.
- 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 the CodeWarrior IDE, 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](#).



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

<b>Target</b>	<b>Targeting Manual</b>
Mac OS	<i>Targeting Mac OS</i>
Win32/x86	<i>Targeting Win32</i>
Java	<i>Targeting Java</i>
BeOS	<i>CodeWarrior BeIDE User's Guide</i> , and the manuals supplied by Be, Inc.
PlayStation	<i>Targeting PlayStation OS</i>
PowerTV OS	<i>Targeting PowerTV</i>
Palm OS	<i>Targeting Palm OS</i>
OS-9	<i>Targeting OS-9</i>

## What's New in This Release

The Metrowerks IDE is now at version 2.0, and with the change in version number there are many significant new features. In addition, the build system and project manager portions of the CodeWarrior IDE have been rewritten to incorporate new features, and provide improved performance.

Among the new features, the following are probably the most notable:

- [Threaded Execution](#)
- [Multiple Open Projects](#)
- [Multiple Targets Per Project](#)
- [Subprojects](#)
- [Mac OS Merge Linker](#)
- [New Toolbars](#)
- [Customizable User Interface Key Bindings](#)
- [Improved Drag and Drop Support](#)

## Introduction

### *What's New in This Release*

---

- [Improved Project File Management](#)

#### **Threaded Execution**

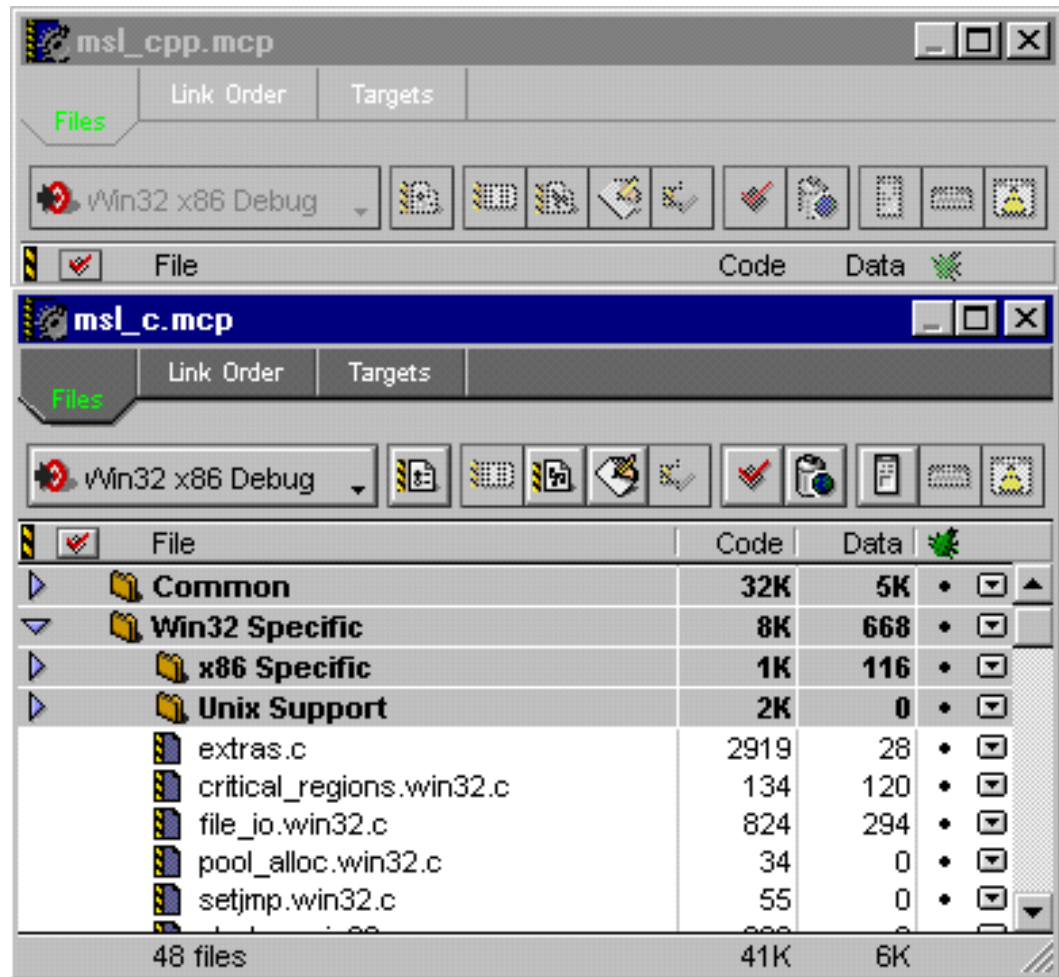
Compilation now occurs in a separate run time task. This means you can use the Editor to edit files at the same time your project is being compiled in the IDE. Other things you can do include using the Browser to browse your code, or using the text search features in the IDE.

Certain operations that would interfere with compilation may be disabled when compilation is in progress, such as changing preferences, target settings, or adding and removing files for the project.

## Multiple Open Projects

The CodeWarrior IDE can now have multiple projects open at a time, as shown in [Figure 1.1](#).

**Figure 1.1** Multiple Open Projects



## Multiple Targets Per Project

It is now possible to have your project contain multiple targets, as shown in [Figure 1.2](#). As a result, you can use the same project to build different versions of your code.

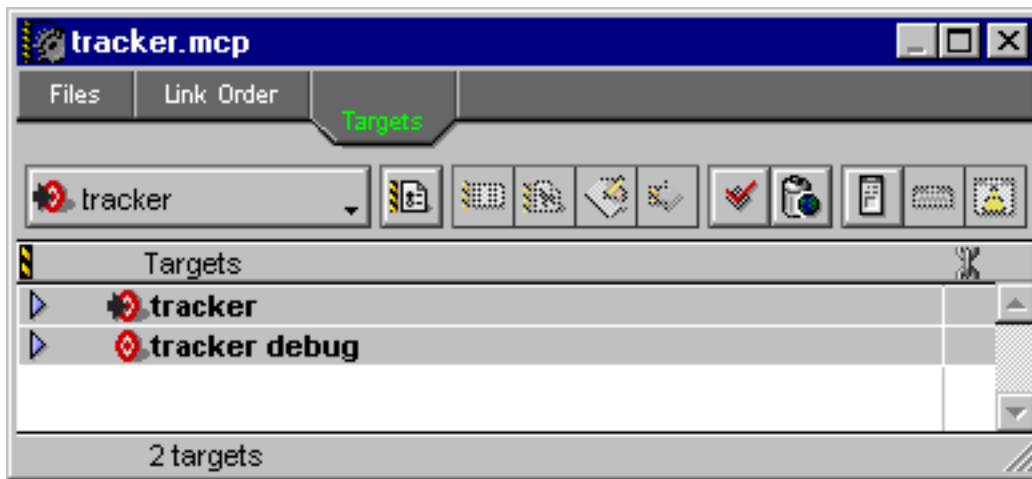
## Introduction

### *What's New in This Release*

---

To learn more information about using multiple targets in your projects, refer to [“Creating Complex Projects” on page 79](#).

**Figure 1.2 Multiple Targets in a Project**

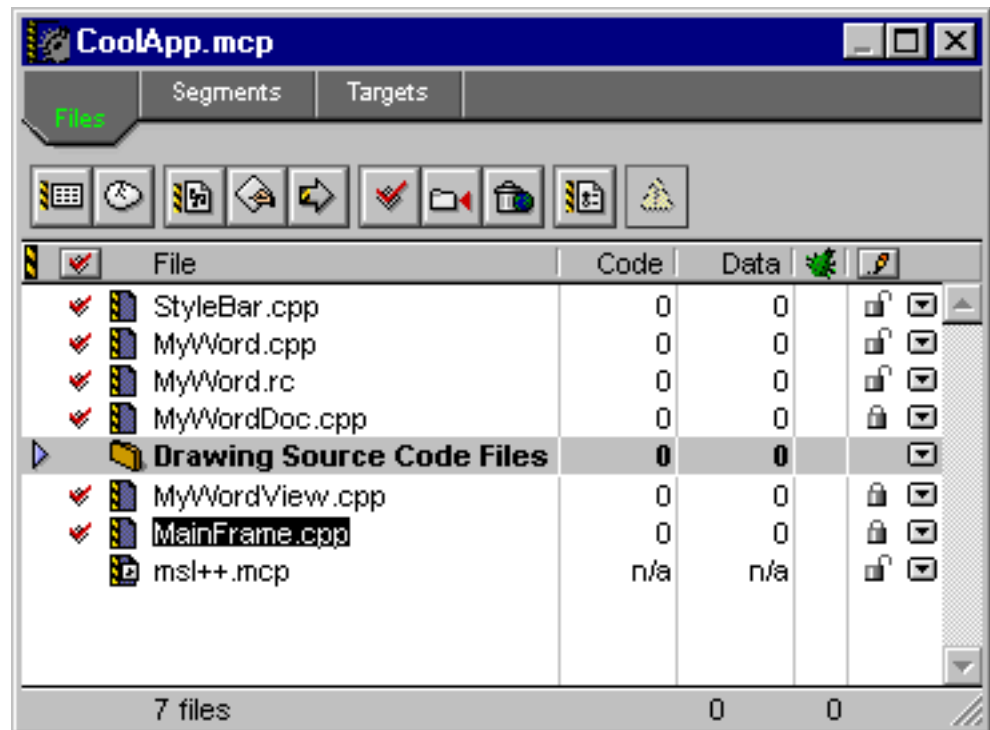


### Subprojects

The CodeWarrior IDE now allows you to use your projects nested within other projects. These nested projects are called subprojects. A subproject file is identical to a project file, except that it is nested inside a project. You add a subproject to your project window just like any other file. [Figure 1.3](#) shows you a subproject named `Cool-App.mcp` as part of the main project. Using this project-embedding technique, you can create a subproject that builds all shared libraries in your main project, then add the subproject to the main project. When you build your main project, the CodeWarrior IDE scans all dependencies of the subproject and builds it automatically if necessary.

To learn more about subprojects, refer to [“Creating Complex Projects” on page 79](#).

**Figure 1.3 Subproject within a Project**



### Mac OS Merge Linker

The new CodeWarrior tools include a new merging linker that can merge Mac OS 68K and PowerPC executables into one application. This allows you to build a “fat” application of both 68K and PowerPC Mac OS binaries in one output file.

To learn more about how to use the merge linker with your Mac OS projects, refer to the *Targeting Mac OS* manual on your CodeWarrior CD.

## Introduction

### *What's New in This Release*

---

#### **New Toolbars**

Many CodeWarrior IDE windows now have configurable toolbars. To learn more about customizing the icons on the toolbars, refer to [“Toolbar Customization” on page 269.](#)

#### **Customizable User Interface Key Bindings**

The CodeWarrior IDE now has customizable key bindings for all menu, keyboard, and editor commands. You can configure the IDE to suit your personal working style using this feature. To learn more about how to use this feature, refer to [“Key Bindings” on page 231.](#)

#### **Improved Drag and Drop Support**

Drag and drop support has been improved. You can add files to a project by dragging files and folders into the project window from the Windows Explorer shell.

To learn more about this feature, refer to [“Adding Files” on page 69.](#)

#### **Improved Project File Management**

With this release, the Resource.frk restriction for project files is gone. The CodeWarrior IDE has been redesigned to remove the need for it.

Storage of the data for the IDE's internal settings for a project is factored among different files, only one of which is required to create the project again.

To learn more about the internal workings of the project file storage, refer to [“Moving a Project” on page 96.](#)



# Getting Started

---

This chapter gives you the information you need to get started with the CodeWarrior IDE. 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](#)
- [Programming Concepts](#)
- [CodeWarrior IDE Guided Tour](#)

---

**TIP:** For a quick look at the CodeWarrior IDE user interface, go to [“CodeWarrior IDE Guided Tour” on page 35](#). The tour gives you your first glimpse of the CodeWarrior IDE interface, particularly the Global 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 IDE product, read the QuickStart guide on the CodeWarrior CD. When you install the CodeWarrior software, you install the CodeWarrior Integrated Development Environment (IDE), including the CodeWarrior tools, languages, and debugger.

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

## Programming Concepts

If you are new to programming computers, this section should help define some of the terms used in this manual. In this section, we'll define terms such as compilers, linkers, and resource files, and give you some information that may help you digest the material in this manual more quickly.

There are four categories of items that you will encounter when creating a computer program:

- [Creating Input Files](#)
- [Generating the Software](#)
- [Output Files \(Product\)](#)
- [Debugging and Refining](#)

As an overview to the process of creating an executable piece of computer software, think of this process as being four steps.

First, you create files that contain computer code statements using a computer language such as C, C++, Java, assembly language (machine code), or Pascal. You might also create files that contain resource descriptions of objects your program will need, such as windows, dialog boxes, menus, and other user interface items.

Next, you apply build tools to help turn your "source" materials into a product that can execute on your intended target computer.



These tools include compilers, linkers, and maybe even assembly-language assemblers. With these build tools you create a software product that can run on your target computer.

You can create many different types of software products, depending on your tools and intended target computer. Some of the products you can create include applications (or executables), shared libraries, and static libraries.

Once you have created a software product, you can use a debugger to run the software and see if it behaves correctly. If it doesn't, you go back to the beginning and make changes to your source material so that the software behaves the way you want it to.

## **Creating Input Files**

The types of files you create include Source Code files, Resource files, a Project file, and Interface or Header Files.

### **Source Code File**

A Source Code file is a text format computer language file, containing program statements, and written in a language such as C, C++, Java, assembly language, or Pascal.

### **Resource File**

A Resource file contains descriptions of items that your software may need, such as window definitions, dialog box layouts, and other items. A resource file may be a binary file linked into your software product, or it may be a text file that is compiled with a special resource compiler before being linked.

### **Interface or Header File**

An Interface file (as it is called in Pascal) or a Header or Include file (as it is called in most other languages) is a text format file that works in conjunction with your Source Code files. Generally, these files contain data and parameter definitions used by your Source Code files.

### Project File

A Project file is a file that describes which Source Code and Resource files to include in your finished software product, and also describes the actions that should be taken when building your product.

## Generating the Software

The types of tools you use to build your software product include Compilers, Assemblers, and Linkers.

### Compilers

A Compiler is a tool that takes a computer language Source Code file, such as a C, C++, Pascal, or Java file, and compiles the file into binary machine code (also called Object Code) that will be used when the final build stage is invoked. A Compiler is one of the first tools that is invoked to build your program.

### Assemblers

An Assembler is a tool that takes an assembly-language Source Code file and produces Object Code.

### Linkers

A Linker is a tool that takes all the Object Code produced by the compilation and assembly stage of building your project, and links all the Object Code together, producing a software product.

## Output Files (Product)

There are many different types of software products. An application, or executable as it may be called, is a piece of code that can run on a computer. It may make use of other software products called libraries. A library is a binary file that can be linked with other Object Code files and other library files to produce a piece of software.

For in-depth information about the kinds of software products you can develop for your target, 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 25.](#)

## Debugging and Refining

A Debugger is a tool that you can use to monitor the execution of your program. With it, you can stop at certain points in your program’s execution and see what the contents of variables are. You can step one line of code at a time and watch your program execute in “slow motion.”

All you need to do to enable the Debugger is produce special format files that can be read by the Debugger when the code is executing. These files are often called debugging info files, or simply referred to by their format (CodeView or SYM files, depending on the platform you are working on).

The CodeWarrior Debugger is discussed in the *CodeWarrior Debugger Manual* on the CodeWarrior Reference CD. Other useful tools for refining your code that complement the debugger can be found on the CodeWarrior Tools CD. ZoneRanger and the Profiler are especially useful.

## 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](#)
- [Global Toolbar](#)
- [Other IDE Components](#)

### Initial View of the IDE

When you launch CodeWarrior, you see the menu bar at the top of the CodeWarrior window, with the Global Toolbar below it, as

## Getting Started

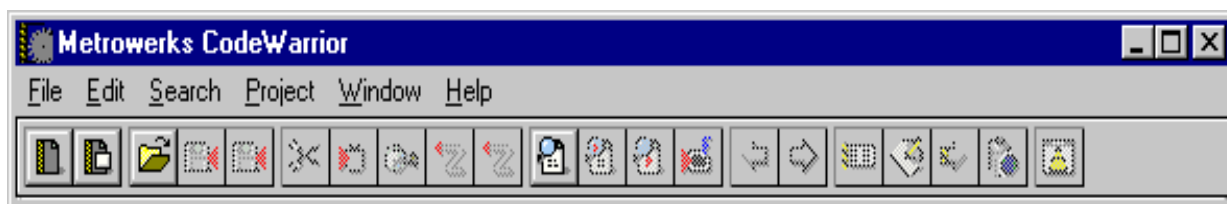
### CodeWarrior IDE Guided Tour

---

shown in [Figure 2.1](#). To learn how to hide the iconic toolbar, refer to [“Show Global Toolbar and Hide Global Toolbar” on page 332](#).

You use the menus and icons on this main menu and Global Toolbar to access the CodeWarrior IDE tools and commands.

**Figure 2.1 The CodeWarrior Menubar**



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 307](#).

The section titled [“Toolbar Customization” on page 269](#) describes how to customize the icons on the CodeWarrior IDE toolbars.

## IDE Menus

The CodeWarrior IDE menus allow you to perform a large number of 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 file.

---

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 307](#).

## File Menu

The [File Menu](#), shown in [Figure 2.2](#), 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 showing a file in the debugger and printing files.

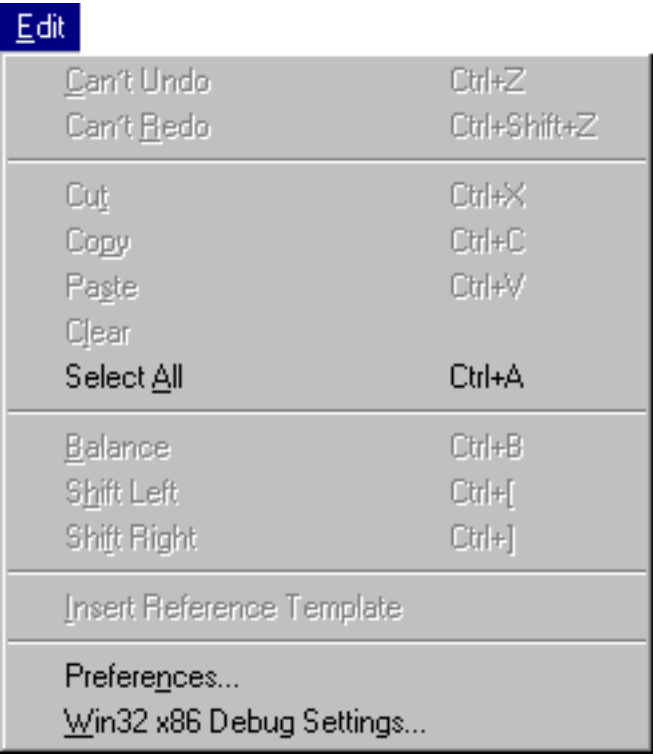
**Figure 2.2** The File menu



**Edit Menu**

The [Edit Menu](#), shown in [Figure 2.3](#), 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. The section titled [“Configuring IDE Options Overview” on page 215](#) describes configuration of the [Preferences](#) and [Target Settings](#).

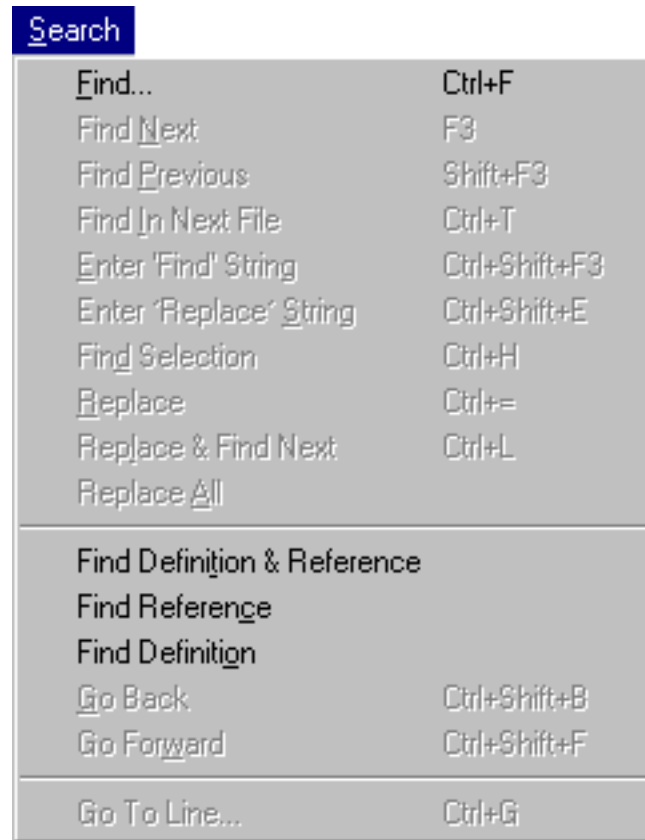
**Figure 2.3    The Edit menu**



## Search Menu

The [Search Menu](#), shown in [Figure 2.4](#), contains all the necessary commands used to find text, replace text, and to find the definitions of routines in your source code.

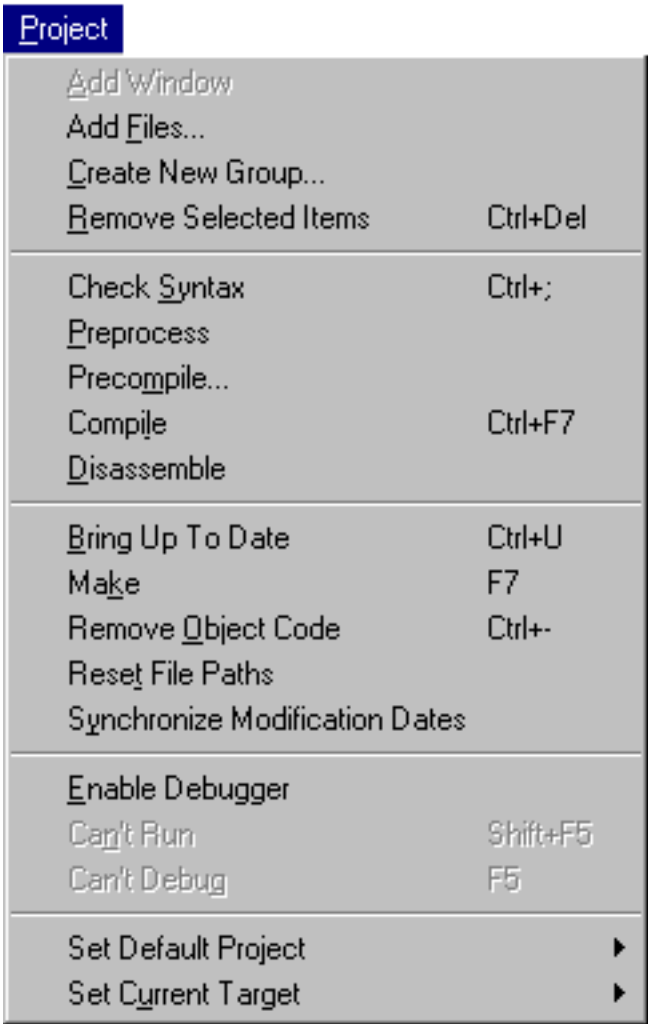
**Figure 2.4**    **The Search menu**



**Project Menu**

Use the commands on the [Project Menu](#), shown in [Figure 2.5](#), 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, disassembling source files, and setting the target and project are also on this menu. In addition, the Debugger may be enabled using this menu.

**Figure 2.5    The Project menu**

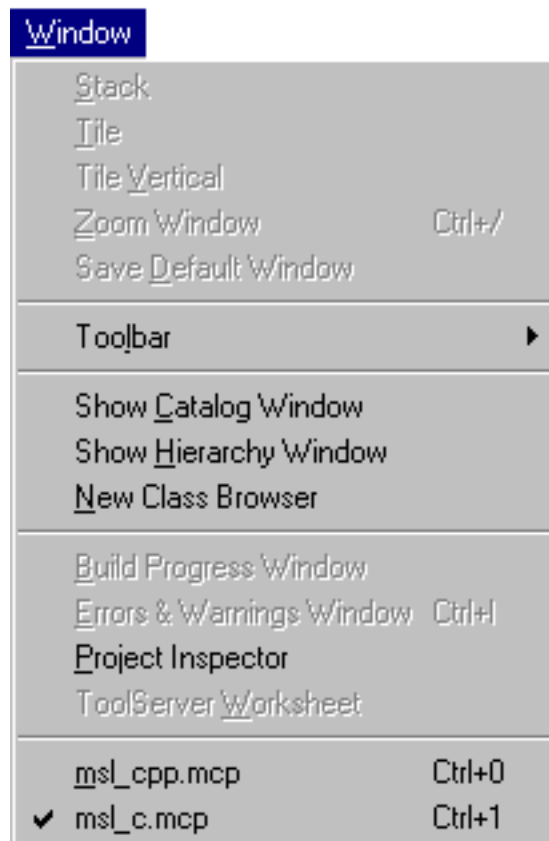




## Window Menu

The [Window Menu](#), shown in [Figure 2.6](#), 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. In addition, there are commands for configuring the toolbars in the IDE, including the toolbars for the project, browser, and editor windows.

**Figure 2.6**    **The Window menu**



## Getting Started

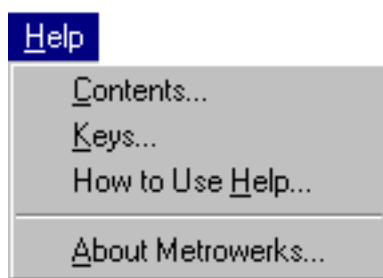
### CodeWarrior IDE Guided Tour

---

#### Help Menu

The [Help Menu](#), as shown in [Figure 2.7](#), 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 2.7** Help Menu



#### Global Toolbar

The Global Toolbar is a palette of icons ([Figure 2.8](#)) that execute a corresponding menu command.

The topics in this section are:

- [Global Toolbar Overview](#)
- [Customizing Toolbar Commands](#)
- [Customizing the Global Toolbar](#)

#### Global Toolbar Overview

The Global Toolbar contains a row of icons that represent commands also listed in the CodeWarrior menus, such as [Add Files](#), [Make](#), or [Run](#).

**Figure 2.8    The Global Toolbar**



### **Customizing Toolbar Commands**

Some of the menus have additional commands accessed by keyboard / mouse combinations. If a command you use frequently is not on the Global Toolbar, it may allow you to save time if you add it to the toolbar.

To learn about how to do this, refer to [“Toolbar Customization” on page 269.](#)

### **Customizing the Global Toolbar**

To learn how to customize the location of the Global Toolbar, or change the icons on it, refer to [“Toolbar Customization” on page 269.](#)

## **Other IDE Components**

This chapter discussed the CodeWarrior IDE menus and the Global Toolbar. 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 58.](#)

For a tour of the Editor Window, see [“Guided Tour of the Editor Window” on page 117.](#)

For a tour of the Browser Windows, see [“Guided Tour of the Browser” on page 188.](#)

## **Getting Started**

*CodeWarrior IDE Guided Tour*

---



# Working with Projects

---

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

## Projects Overview

A project is a collection of files that the CodeWarrior 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 CodeWarrior 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, and saving projects. It also describes operations such as moving files in the project window, marking files for debugging, creating nested projects and targets, and dividing the project window into segments or groups of files.

The topics in this chapter are:

- [Creating a Simple Project](#)
- [Opening an Existing Project](#)
- [Saving a Project](#)
- [Closing a Project](#)
- [Choosing a Default Project](#)
- [Guided Tour of the Project Window](#)
- [Managing Files in a Project](#)

- [Creating Complex Projects](#)
- [Guided Tour of the Project Window](#)
- [Moving a Project](#)
- [Controlling Debugging in a Project](#)

## Creating a Simple Project

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

CodeWarrior projects can be configured to have multiple targets, and may also contain subprojects. To learn more about these topics, refer to [“Creating Complex Projects” on page 79](#) after reading the material in this section.

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, the CodeWarrior IDE creates a new project and, optionally, a new project folder. It then copies all the stationery information files 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 projects only) and names.

If you use a stationery 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 the CodeWarrior IDE.

## 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](#).

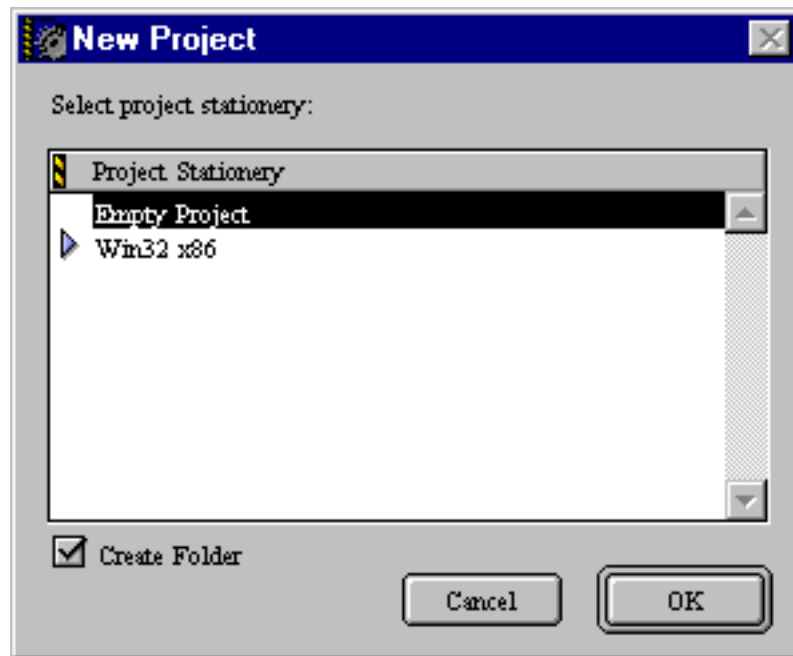
## Working with Projects

### *Creating a Simple Project*

---

The CodeWarrior IDE displays the [New Project](#) window, shown in [Figure 3.1](#), from which you choose your project stationery.

**Figure 3.1** New Project window

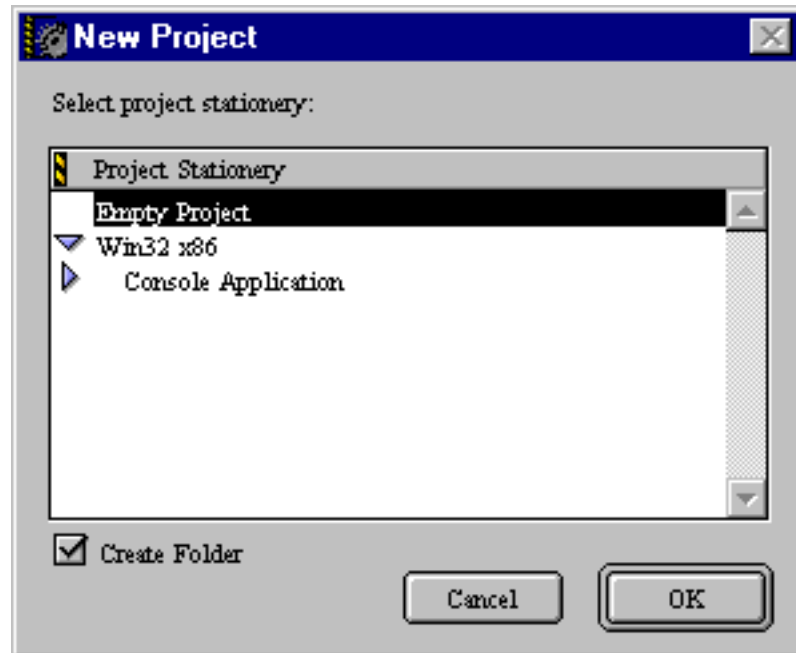


Click on a disclosure triangle in the window (the triangles on the left side of the list) for the type of code that you are interested in. You then see the project stationery available for that target. [Figure 3.2](#) shows several levels expanded. It may be necessary for you to click on more disclosure triangles to show the stationery of interest to you.



If you want to create an empty project that contains no libraries or other support files, you may choose the Empty Project option from the list.

**Figure 3.2** Choosing project stationery



Click one of the project stationery items that corresponds to your intended target code type. If you want to create a new folder containing all the new project files, make sure the Create Folder checkbox has a check in it by clicking on it.

### **Naming Your New Project**

Click the OK button and 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 box 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 to disk.

---

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

## Working with Projects

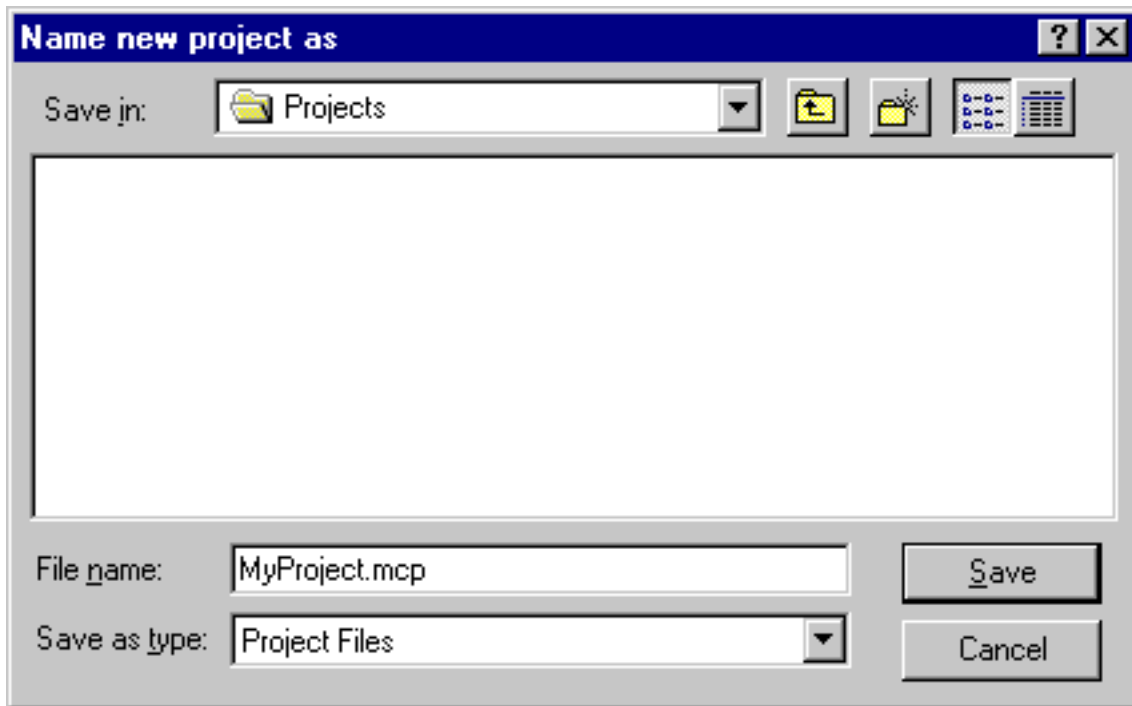
### *Creating a Simple Project*

---

and is the convention that the CodeWarrior IDE uses for project files.

---

**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.

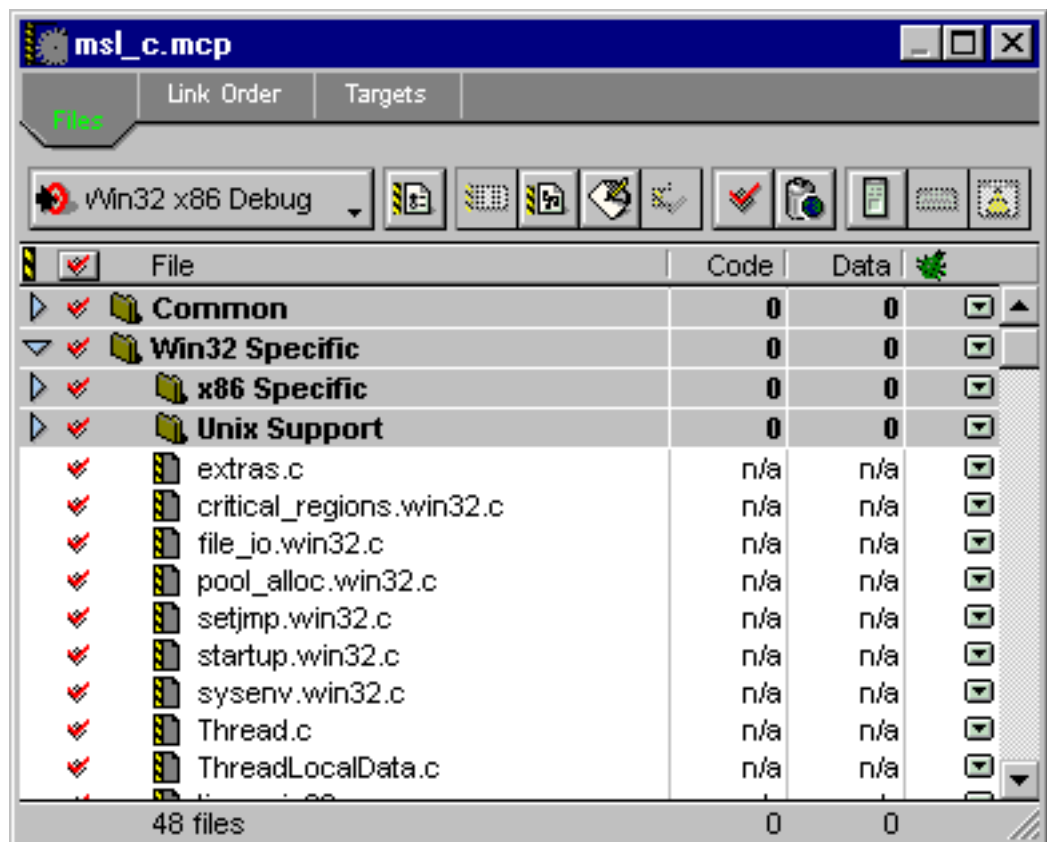
---

When you choose Save, the CodeWarrior IDE 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 49](#)), minus the file extension. The new folder contains your new project file for the stationery you chose.

- Setting [Target Settings](#) and [Preferences](#) to be the same as the settings stored in the chosen stationery.
- Opening the project window. The new project contains libraries, source code placeholders, and resource file placeholders. If you chose to create an empty project, there will not be any files or libraries in your new project window.

**Figure 3.4** A project window



### Modifying Your New Project

Most new projects created from stationery contain source code files that are basically empty placeholders. You can remove these and substitute your own files. Refer to the section titled [“Managing Files in a Project” on page 65](#) for more information about removing and adding files in a project.

## Working with Projects

### *Creating a Simple Project*

---

You may also want to add additional libraries to your project file. To learn about which libraries to include, refer to the targeting manual of interest to you. A table of all the targeting manuals for the CodeWarrior product can be found as [Table 1.2 on page 25](#).

### **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 creating. To learn how to build a project, refer to the discussion in [“Compiling and Linking Overview” on page 271](#).

### **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 inside 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
- Project stationery files

Refer to the ReadMe file in the (Stationery) folder for more information about the stationery.

### **Creating Your Own Project Stationery**

You can create a unique stationery or “template” project file that includes the files and options you want to have for a starter project. This stationery project can be reused whenever you are creating a new project, so that you always start from your own customized settings.

To create your own project stationery file, create a new project or open an existing project. Then, change the settings using the [Preferences](#) command on the [Edit Menu](#). You should then save your new stationery project in the appropriate subfolder in the (Stationery) folder.

To learn how to configure [Preferences](#) settings, refer to [“Choosing Preferences” on page 220](#). You can also customize [Target Settings](#) for your custom stationery in a similar way. To learn more about how to do this, refer to [“Choosing Target Settings” on page 244](#).

Next, add the appropriate files as desired for your stationery project. You can add your own “placeholder” files that will always appear in your stationery projects. To do this, copy the “placeholder” files you want to appear in your stationery into your new subfolder inside the (Stationery) folder on your hard disk. You should have one folder per stationery project that you create.

For information about adding or changing files in the project, see [“Managing Files in a Project” on page 65](#).

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 box that appears. If you would like to learn more information about this process, refer to [“Backing Up Files” on page 110](#) 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 inside one of the subdirectories inside the (Stationery) folder, which is located in your CodeWarrior folder.

Give the project a name and click the Save button.

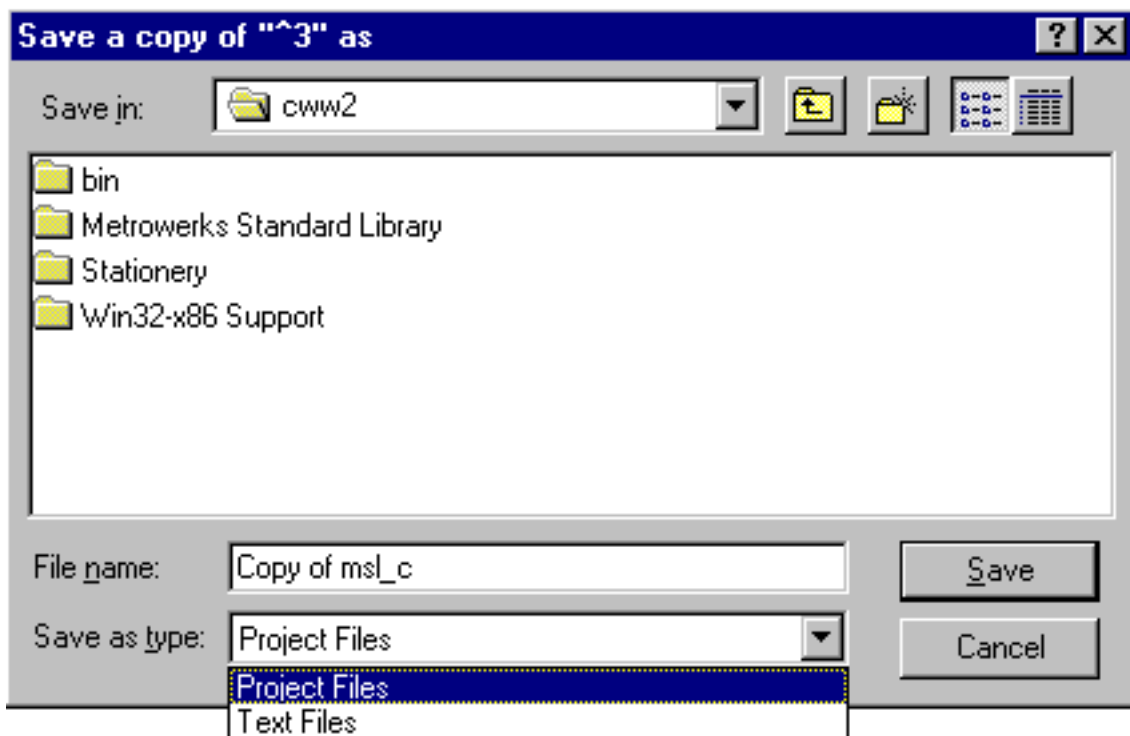
## Working with Projects

### Creating a Simple Project

---

By saving your stationery in one of the subdirectories, you will have your stationery available the next time you use the [New Project](#) command.

**Figure 3.5** Save A Copy As Dialog Box



---

**TIP:** We suggest naming your stationery project ending with a .mcp. This makes your project easier to identify on your hard disk and obeys the conventional naming that the CodeWarrior IDE expects for a project file.

---

The reason you want to save the stationery project before doing a lot of work with it is because you will have a “starter” or “template” project file on your hard disk. You can make copies of the “starter” project to get new projects quickly started, with all your favorite settings, by using the [New Project](#) command.

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 for new projects, you just create a new stationery project. Just make sure that you have a project window open, then configure your options, then save your new stationery project in the appropriate stationery folder.

## Opening an Existing Project

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

Note that you may have many different projects open at a time, not just one project.

The current project window you have open can be made the active window using the [Window Menu](#) ([Figure 11.5 on page 327](#)). To switch to one of these opened projects, just choose a project from this menu. You can also click in the window to make it the active window.

The topics in this section are:

- [Using the Open Command](#)
- [Using the Open Recent Command](#)
- [Using the Project Window to Open Subprojects](#)

### Using the Open Command

Information on how to open a project file using the [Open](#) command on the [File Menu](#) is found in the section of the manual titled [“Opening Files with the File Menu” on page 102](#).

You can have more than one project open at a time in the CodeWarrior IDE.

## Using the Open Recent Command

The CodeWarrior IDE maintains some of the projects you have opened recently in a list under the [File Menu](#). As a convenience, you may use the [Open Recent](#) menu command to reopen one of these projects.

## Using the Project Window to Open Subprojects

If your project contains subprojects, and you want to open one of those subprojects, you can double-click on the project file icon in the Project window to open it.

To learn more about subprojects, and how to add them to your Project window, refer to [“Creating Complex Projects” on page 79](#).

## Saving a Project

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

Some of the times when your settings get saved are when you:

- Close the project
- Change [Preferences](#) or [Target Settings](#) for the project
- Add or delete files for the project
- Compile any file in the project
- Edit Groups or Segments in the project
- Remove Object Code from the project
- Quit the CodeWarrior IDE

You never have to manually save your project unless you want to create a copy of it, since the project is automatically saved each time it is closed, and also when other common actions are performed.



## Items Saved with Your Project

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

- The names of the files added to your project and their locations
- All configuration options
- Dependency information (the touch state and interface file lists)
- Browser information
- The object code of any compiled source code files

## Saving a Copy of Your Project

The CodeWarrior IDE lets you save a copy of the project under a new name. To learn about using this feature, see the discussion in [“Backing Up Files” on page 110](#).

## 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 IDE application to work on something else. To read about how to close your project, refer to [“Closing One File” on page 112](#).

You don’t have to close your project before quitting the CodeWarrior IDE application, since your project settings are automatically saved. To learn more details about saved projects, refer to [“Saving a Project” on page 56](#).

The CodeWarrior IDE will allow you to have more than one project open at a time, so you don’t have to close each project you are finished with before switching to another project. Just open your new project and begin working with it.

## Working with Projects

### *Choosing a Default Project*

---

---

**TIP:** Having multiple projects open at a time consumes more memory on your computer, and also causes project opening times to lengthen slightly.

---

## Choosing a Default Project

Since the CodeWarrior IDE permits multiple open projects, it is sometimes ambiguous as to what project is used when you perform a [Make](#), [Run](#), or other operation on your project. If the active window is a project window, that project will be used for any builds that are started. Source code files can be in more than one open project. So you can set which project is the default project for builds using the [Set Default Project](#) menu command on the [Project Menu](#). In any ambiguous case such as this, when a source code file may belong to one or more open projects, the CodeWarrior IDE will operate on the default project you have chosen using the [Set Default Project](#) menu command.

The first project you open becomes the default project. If you close the default project, the default project will then be set to the Project window that is closest to the top.

## Guided Tour of the Project Window

The project window displays the project information you are working with, showing the project's files, grouping, 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 3.6 on page 60](#) for a detailed look at the project window.

To learn more about debugging information, see [“Controlling Debugging in a Project” on page 96](#).

## Navigating the Project Window

To navigate the project window, use the scroll bar 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, if available, to jump from the first file in the first segment or group (Home Key) to the last file in the last segment (End Key).

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

To learn about a technique for selecting files as you type, refer to [“Selection by keyboard” on page 69.](#)

## Project Window User Interface Items

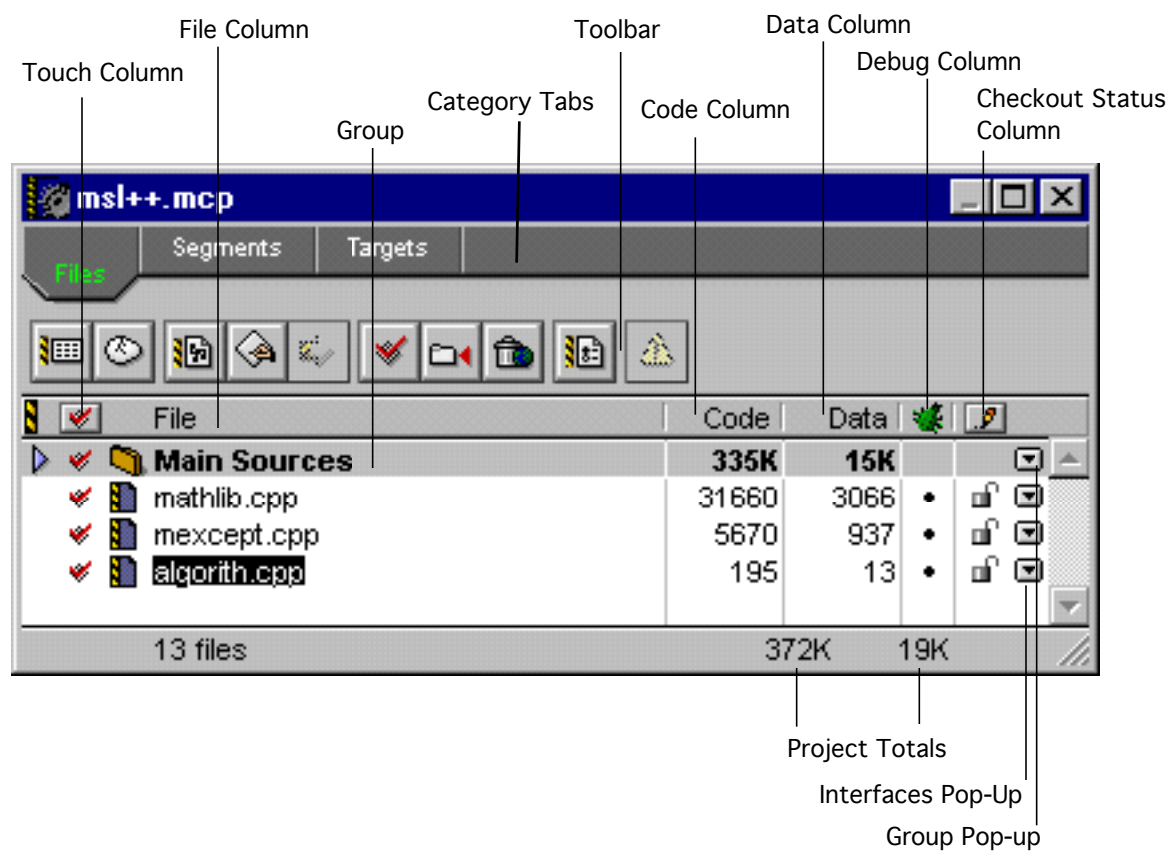
This section will take you on a tour of the various user interface items that you can use to alter your project view or project settings. The project window contains many pop-up menus and columns that indicate segmentation/grouping, debugging, access path, and interface file information. This section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

The topics in this section are:

- [Category Tabs](#)
- [Toolbar](#)
- [Group Organization](#)
- [File Column](#)
- [Code Column](#)
- [Data Column](#)
- [Debug Column](#)
- [Touch Column](#)
- [Interfaces Pop-up](#)
- [Group Pop-up](#)

- [Checkout Status Column](#)

**Figure 3.6 The Project Window**



### Category Tabs

You click on one of the three Category Tabs, shown in [Figure 3.6](#), to change the view of your project window. The Category Tabs are Files, Segments or Link Order, and Targets. The Files tab allows you to inspect some information for files in your project. The Link Order or Segments tab (name depends on your target type) allows you to group your files for segmenting purposes. The Targets view allows you to configure what targets your project will build code for, and in what order the targets are built.

To learn more about targets, refer to [“What is a Target?” on page 80](#). To learn more about working with files in the project window, refer

to [“Managing Files in a Project” on page 65](#). To learn more about segmenting files, refer to [“About Groups and Segments” on page 65](#).

## **Toolbar**

The Toolbar in the project window allows you to perform shortcuts for commands to boost your productivity. You may configure which icons are on the Toolbar, and the order in which they are displayed. You can even configure whether the toolbar is visible. To learn more about toolbars in the CodeWarrior IDE and how to configure them, refer to [“Toolbar Customization” on page 269](#).

## **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 at the top left edge of each group, and entire groups can be moved around in the window. [Figure 3.6](#) shows an example of file groups in the Files category of the project window.

The files shown in the Files view are all the files in all targets of the project, while the files shown in the Segments or Link Order view show the link order for the current target.

Items shown in the Files view can be grouped together with an arbitrary hierarchy, and rearranging items has no effect on the final binary code that is produced by your project. By default, the CodeWarrior IDE compiles files in the specified order, but will depart from this to follow dependencies where necessary. This is because there could be files the IDE couldn't build right the first time because the dependencies were not known until after the first build. If the files are in an unsympathetic dependency order, the build could fail. By putting the files in a correct order in the Link Order or Segments view the user can allow the first build to succeed.

Items in the Link Order or Segments view can only be nested one level deep. The files in this view are arranged in the order they will be compiled. Changing their order affects the final binary code that is produced by your project file.

## Working with Projects

### *Guided Tour of the Project Window*

---

While this file grouping capability is definitely useful from an organizational standpoint, it also has significance for some particular targets you may be building. Using the Segments category, some targets use the grouping facility to produce code segments at link time. These segments of code break the entire program code into pieces, or segments. Sometimes this code segmenting is required by the target computer's hardware architecture. For more discussion about groups, refer to [“Managing Files in a Project” on page 65](#).

### **File Column**

The File Column of the project window, shown in [Figure 3.6](#), shows which files are in your project and the names of the groups which contain those files. Files can be added, moved around, opened, or removed from within groups.

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. For information on opening files from the File Column, see [“Opening Files from the Project Window” on page 104](#).

For information on adding, moving, or removing files, see [“Managing Files in a Project” on page 65](#).

### **Code Column**

The Code Column of the project window, shown in [Figure 3.6 on page 60](#), 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 may only use the data and code that are referenced by other files in the project.

The numeric values shown are only for items in the current target. Values for items not in the current target are shown in gray.

For more information on how the linker works, see [“Compiling and Linking a Project” on page 273](#).

### **Data Column**

The Data Column of the project window, shown in [Figure 3.6 on page 60](#), shows the size, in bytes or kilobytes, of the non-executable data area for a corresponding file. If zero is displayed, it means that 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 necessarily use all a project file’s data. Instead, the linker may only use the parts necessary to completely link the project.

The numeric values shown are only for items in the current target. Values for items not in the current target are shown in gray.

For more information on how the linker works, see [“Compiling and Linking a Project” on page 273](#).

### **Debug Column**

The Debug Column of the project window, shown in [Figure 3.6 on page 60](#), 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 97](#).

### **Touch Column**

The Touch Column of the project window, shown in [Figure 3.6 on page 60](#), indicates whether a file needs to be compiled the next time a project is built.

When clicked, the Touch icon at the top of the Touch column causes all files in the project to have their modification dates synchronized. To learn more about this feature, refer to [“Synchronizing modification dates” on page 79](#).

## Working with Projects

### *Guided Tour of the Project Window*

---

When clicking in the Touch Column next to the segment/group names, all files in that group/segment will be touched.

For more information about touching and untouching files, see [“Touching and Untouching Files” on page 274.](#)

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

The Interfaces Pop-up in the project window, shown in [Figure 3.6 on page 60](#), 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 Pop-up Menu” on page 105.](#)

For more information about touching and untouching files, see [“Touching and Untouching Files” on page 274.](#)

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

The Group Pop-up of the project window, shown in [Figure 3.6 on page 60](#), 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 Pop-up](#) since it is in the same row as the name of a Group.

For more information about opening files this way, refer to [“Opening Files from the Project Window” on page 104.](#)

#### **Checkout Status Column**

The Checkout Status Column is not yet available for use in this release of the CodeWarrior product.



## 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 in this section are:

- [About Groups and Segments](#)
- [Expanding and Collapsing Groups](#)
- [Selecting Files and 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. Segments provide a similar function that is different in a subtle way. Some target linkers break executable code into segments. These segments of executable code are swapped in and out of memory while the program is running. Creating groups of these files tells the linker which files to put in one segment.

In other words, you can use Groups to group your files together in the Files view of the project window. In the Segments view of the project window, other targets that require segmenting use the file grouping to create code segments of related code.

For example, in the Segments tab view, by grouping all the files that contain sound playback functionality into a group named “Sound,” you would be creating a segment of code that is loaded when your program’s execution calls for playing sound. When another segment needs room in memory, and no sound is being played, your “Sound” segment would be swapped out of memory by the operating system to free space for the new segment to load.

## Working with Projects

### *Managing Files in a Project*

---

Now, in the Files view, you can still group all the “Sound” files together in a group, but these files would not necessarily be put into the same segment, unless the `#pragma segment` directive were used in the source code.

Motorola 68K projects use segments. PowerPC, Win 32/x86, CFM68K, Java, and Be projects do not use segmenting.

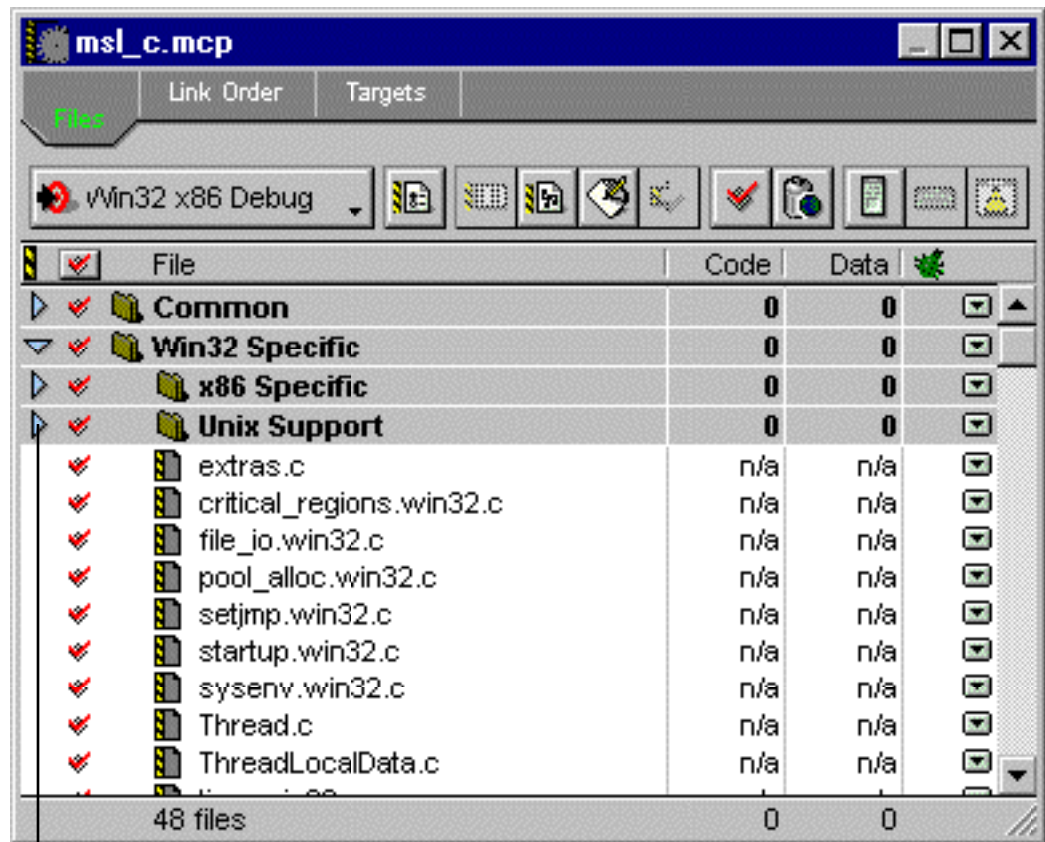
In summary, Groups allow you to organize your source code files into logical categories to help you keep track of where you put your files. For some targets, Segments serve the additional purpose of assigning code to segments that the linker will create.

## Expanding and Collapsing Groups

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

To expand a group and view its files, click on the disclosure triangle in the top left edge of the desired group. To close a group and view only its name, click the triangle again ([Figure 3.7](#)).

Figure 3.7 Expanding and collapsing groups



Click on a disclosure triangle to expand a group, showing all of its files.

Alt-click on a disclosure triangle to do a "deep expand." This action expands a group and all its subgroups in the project window. It does not expand groups that are at the same level as the group you Alt-clicked. Instead, use Control-click to expand these sibling groups. By doing a Control-click or Alt-click again, the expansion that you did will be collapsed.

## 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.

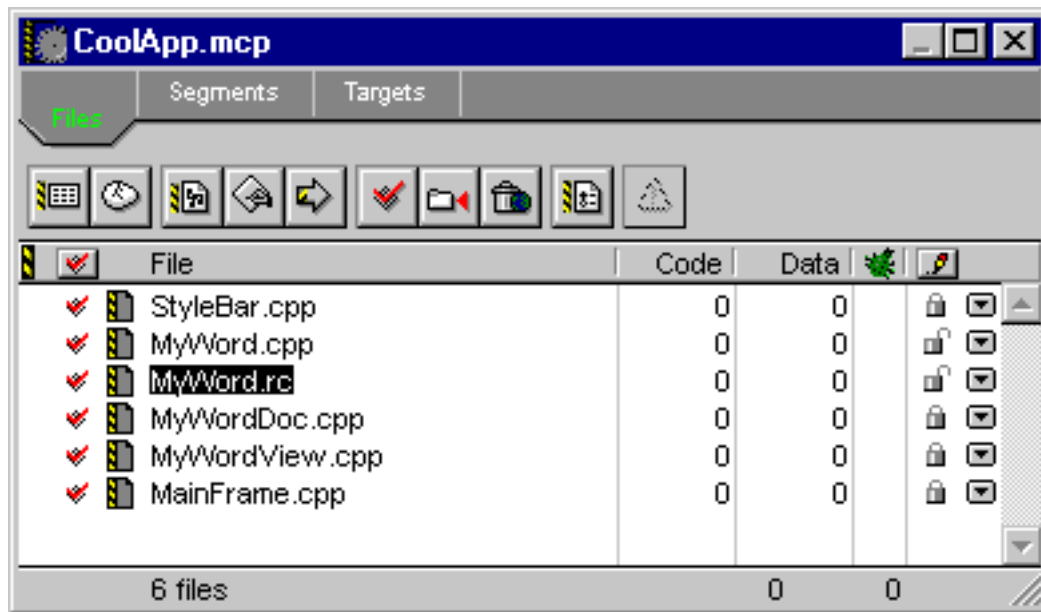
## Working with Projects

### Managing Files in a Project

---

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 on 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.

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 CodeWarrior IDE selects the file in the project window 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 files in currently-expanded groups in the Project window can be selected this way. Files in collapsed groups will not be matched with your keystrokes.

---

## **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” on page 247.](#)

For more information about the Message Window, see [“Guided Tour of the Message Window” on page 290.](#)

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 Drag and Drop](#)—add one or more files
- [Using the Add Window Command](#)—add one file

## Working with Projects

### *Managing Files in a Project*

---

#### 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 67.](#)

If no group is selected in the Files view of the project window, new items will be added to the end of the list instead of within an existing group. If no group is selected in the Segments view, new items will get added to the end of the first segment.

---

**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 73.](#)

#### Using the Add Files Command

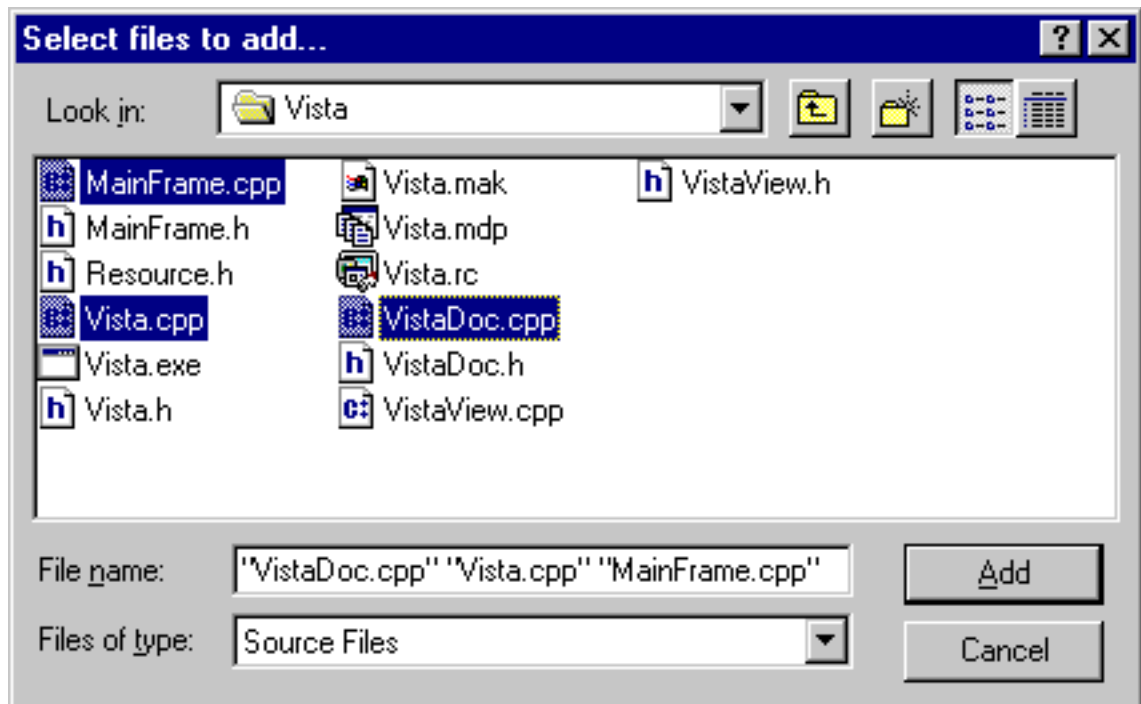
The [Add Files](#) command on the [Project Menu](#) opens a dialog box you can use to add files to your project from many locations. Use this command to add source code files, libraries, resource files, or a shared library.

This command opens a dialog box that can be used to add files to your project from a directory ([Figure 3.9](#)). In order for files to appear in this Add Files dialog box, the files must have a recognized extension (such as .c or .p) at the end of the filename. To examine and configure possible extensions for file names, refer to [“Setting a File Extension” on page 273.](#)

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

---

**Figure 3.9** Adding Files to a Project



**TIP:** To select multiple files, as shown in [Figure 3.9](#), press the Control key when clicking on a file name in the dialog box. To select a contiguous group of files, click on the first file name, then press the Shift key and click the last file that you want to end your group.

---

Note that if your project contains multiple targets, you will be prompted to choose the targets that your files should be added to.

## Working with Projects

### *Managing Files in a Project*

---

#### Using Drag and Drop

When you drag and drop files or folders onto the CodeWarrior project window, they will be added to the project.

To add files to your project using this method, first select the files or folders you want to add to the project.

You can select files in many places, including the Desktop, or the multi-file search list in the CodeWarrior IDE's Find dialog box.

To complete the add operation, drag your selection onto the project window. Note that if your project contains multiple targets, you will be prompted to choose the targets that your files should be added to.

When dragging the selected files onto the project window, the CodeWarrior IDE verifies that the files can be added to the project. When dragging a folder, the CodeWarrior IDE checks to make sure that the folder, or one of its subfolders, contains at least one source code file, library, or resource file, and that file is not already in the project.

If the selection does not contain at least one file recognized by the CodeWarrior IDE, the project window will not be highlighted, indicating that the selection cannot be added to the project.

If the project window is highlighted, then releasing the mouse adds the selected files and the contents of the selected folders to the project, appending them to the segment/group which the cursor is in. To add files to a new segment/group, release the mouse when the cursor is over the blank space after the last segment/group.

The new files will be added after the selected item in the project window, so activate that window and select the item before dragging for best results.

The CodeWarrior IDE does not allow the dragging of entire volumes (such as your hard disk) onto the project window because adding files from an entire hard disk would probably take too long.



The CodeWarrior IDE allows dragging and dropping outside of the project window. You can also drag files to another application to open them in that application.

Although the CodeWarrior IDE supports dragging and dropping files into the project window, you cannot drag files out of the project window. To learn how to remove files from the project window, refer to [“Removing Files and Groups” on page 75](#).

### **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.

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](#). If the window is untitled, the [Save As](#) dialog box appears, 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 that if your project contains multiple targets, you will be prompted to choose the targets that your files should be added to.

---

**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 recognized file name extension (to learn about configuring permissible file name extensions, refer [“File Mappings” on page 253](#)). 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 67](#).

## Working with Projects

### *Managing Files in a Project*

---

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 small triangle 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 group 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.

Dragging items past the last group adds them to the top level of the window, instead of creating a new group.

After creation, the project's groups are renumbered to include the new group. The new group will have a name like "Group 3" or some other number. For information on how to change this name, see ["Renaming Groups" on page 76](#).

## Removing Files and Groups

When removing files from the project window, you need to be aware of a subtle issue. If you remove files from the Files view, they are removed from the entire project, including all targets. When removing files from the Segments view, the files are just deleted from the current target.

To learn more about targets, refer to [“What is a Target?” on page 80](#).

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 67](#).

---

**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](#).

---

After selection, choose the [Remove Selected Items](#) 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 the CodeWarrior IDE requires at least one group in a project.

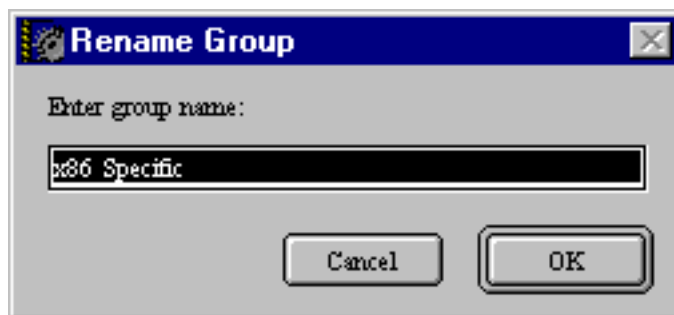
To remove a group from your project, select the group to remove and press Shift-Control-Return. CodeWarrior moves the files in this group up to the previous group and the group numbers are updated.

## 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 name text field you use to enter a new name ([Figure 3.10](#)).

**Figure 3.10** Changing a group name



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 box 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 box'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 3.11](#) to flag files that need compilation. The CodeWarrior IDE doesn't always recognize file changes and may not automatically recompile all files in certain cases, which is why the Touch Column features are useful.

There are three 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.

Another way is to select the Touch command from the [Interfaces Pop-up](#) menu. The Touch command may appear at the top of the [Interfaces Pop-up](#) and the [Group Pop-up](#).

The last way to make sure that changed files get compiled is to click on the Touch Column icon at the top of the column. This will resynchronize the state of the files in the project depending on the dates they were last modified. This is useful if the files have been modified outside of the CodeWarrior IDE, perhaps by a third-party editor.

---

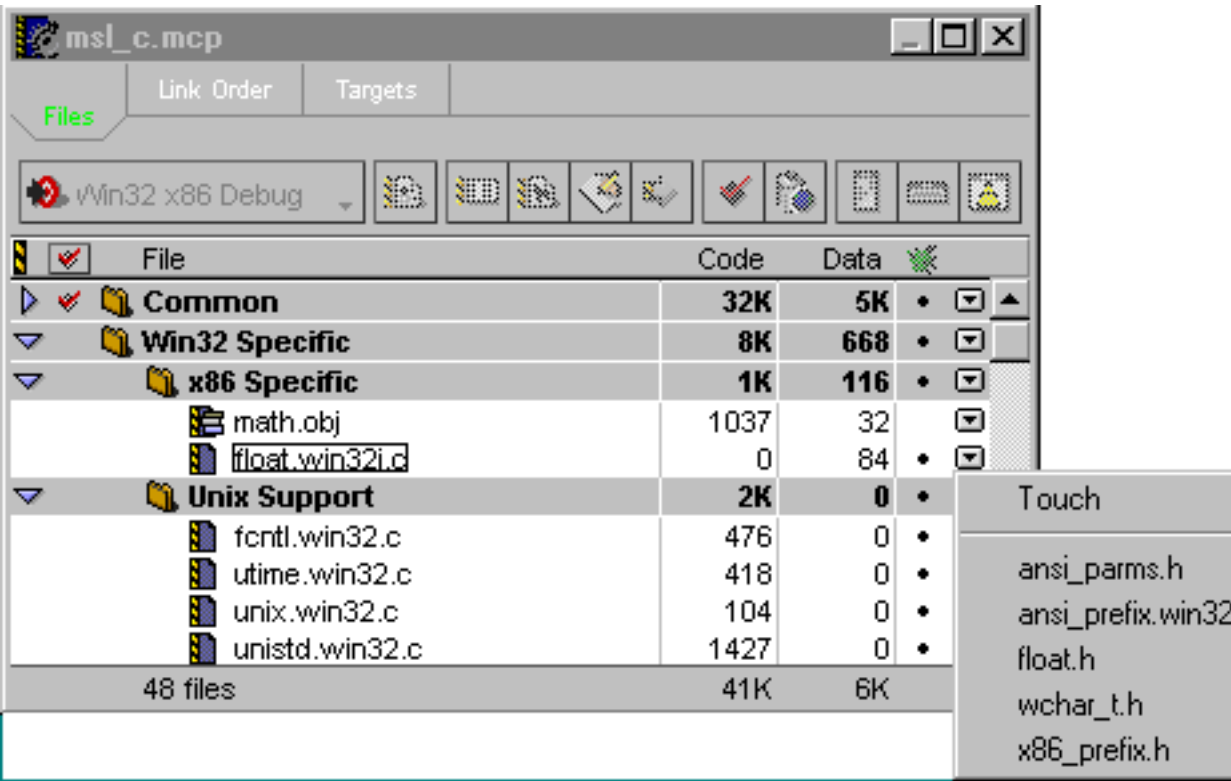
**TIP:** If the file hasn't been changed since it was last compiled, the first command in either pop-up is Touch. When you choose Touch, the CodeWarrior IDE marks your file as dirty and compiles the file the next time it makes your project. If the file has been changed since it was last compiled, the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

---

Refer to [Figure 3.11](#) to see this illustrated in a project window. You'll see a checkmark in the [Touch Column](#) next to the filename if the file

is marked for compilation, and the Touch command in the [Interfaces Pop-up](#) for the file you touched becomes Untouch.

**Figure 3.11    Marking files for compilation**



To unflag files you flagged for compilation, click again in the [Touch Column](#) left of the file name, or choose Untouch from the [Interfaces 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 274](#).

### **Synchronizing modification dates**

To update the modification dates stored in your project file, click the checkmark icon above the [Touch Column](#), next to the barber pole icon, as shown in [Figure 3.11](#).

Alternatively, choose the [Synchronize Modification Dates](#) command in the [Project Menu](#).

The CodeWarrior IDE 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, the CodeWarrior IDE marks it for recompilation. This will resynchronize the state of the files in the project depending on the dates they were last modified. This is useful if the files have been modified outside of the CodeWarrior IDE, perhaps by a third-party editor that doesn't notify the CodeWarrior IDE when it modifies a file.

## **Creating Complex Projects**

The CodeWarrior IDE provides flexible facilities for creating project files containing sophisticated build rules. This section discusses how to construct complex project files that may contain multiple different kinds of build target code, or contain other projects. This facility allows you to create powerful build hierarchies for your entire software project.

For example, you may want to create complex projects so that one project file can contain targets for both shipping and debugging versions of code. By switching between shipping and debug targets, different versions of the code can be generated during the development process. Each of these targets can have their own settings. For example, the debugging target could have optimizations disabled and debugging information enabled, and the shipping target can have code generation optimizations enabled.

Using the CodeWarrior IDE's targets and subprojects, it is even possible to do builds of both Mac OS software and Win32 software right on your computer!

## Working with Projects

### *Creating Complex Projects*

---

The topics in this section are:

- [What is a Target?](#)
- [What is a Subproject?](#)
- [Strategy for Creating Complex Projects](#)
- [Creating a New Target](#)
- [Changing a Target Name](#)
- [Changing the Target Settings](#)
- [Setting the Current Build Target](#)
- [Creating Target Dependencies](#)
- [Assigning Files to Targets](#)
- [Creating Subprojects Within Projects](#)

## What is a Target?

A target is a set of rules and settings that you configure to produce a code product from a build.

The CodeWarrior IDE has the capability to build many different code products, or targets, from one project file, as shown in [Figure 3.12](#). For example, this is useful if you want to have a build of your code for debugging, and a separate build for your shipping code.

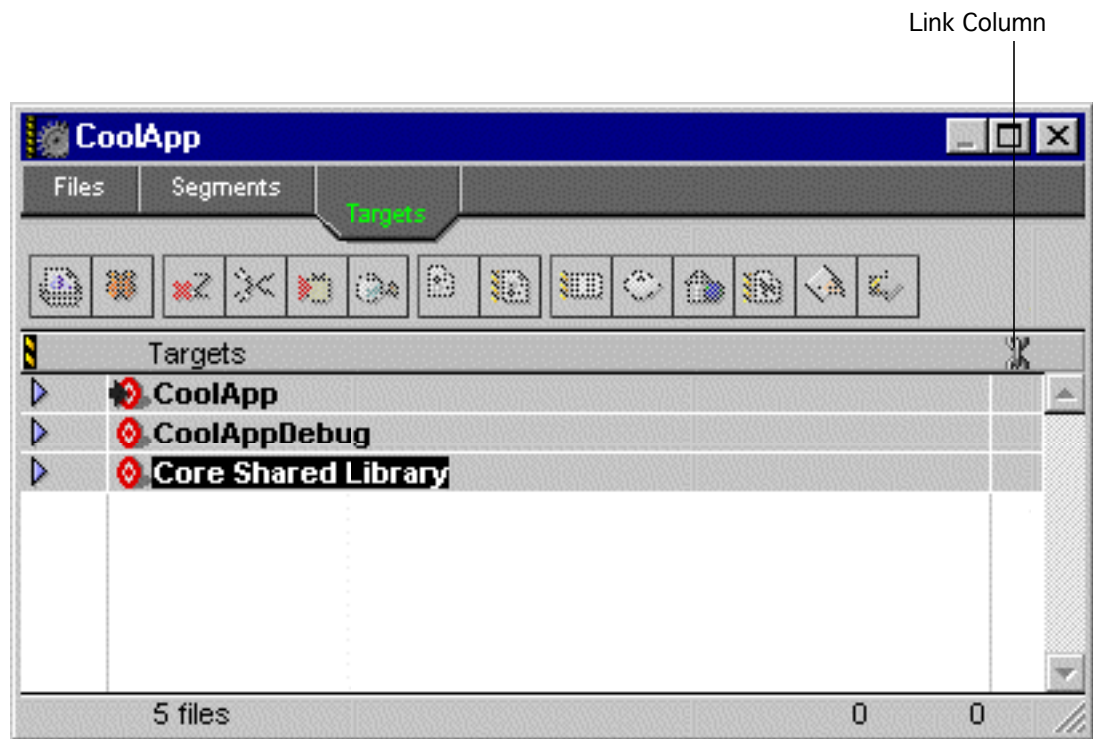
You can also define targets that are common to multiple targets, so that one target gets built before trying to build another. This could be useful for sharing resource files between other targets. In this way, you can create a target that depends on some other target, forcing the latter target to build first.

Each target in the project can have its own build settings. Each set of target settings is distinct.



To learn more about considerations for using targets, refer to [“Strategy for Creating Complex Projects” on page 83.](#)

**Figure 3.12 Multiple Targets in the Project Window**



## What is a Subproject?

A subproject is a normal, stand-alone project file that can be nested within a project, as for the project file at the bottom of the project window named `msl_cpp.mcp` shown in [Figure 3.13](#). Subprojects are useful if you have a project file that you want to keep separate from

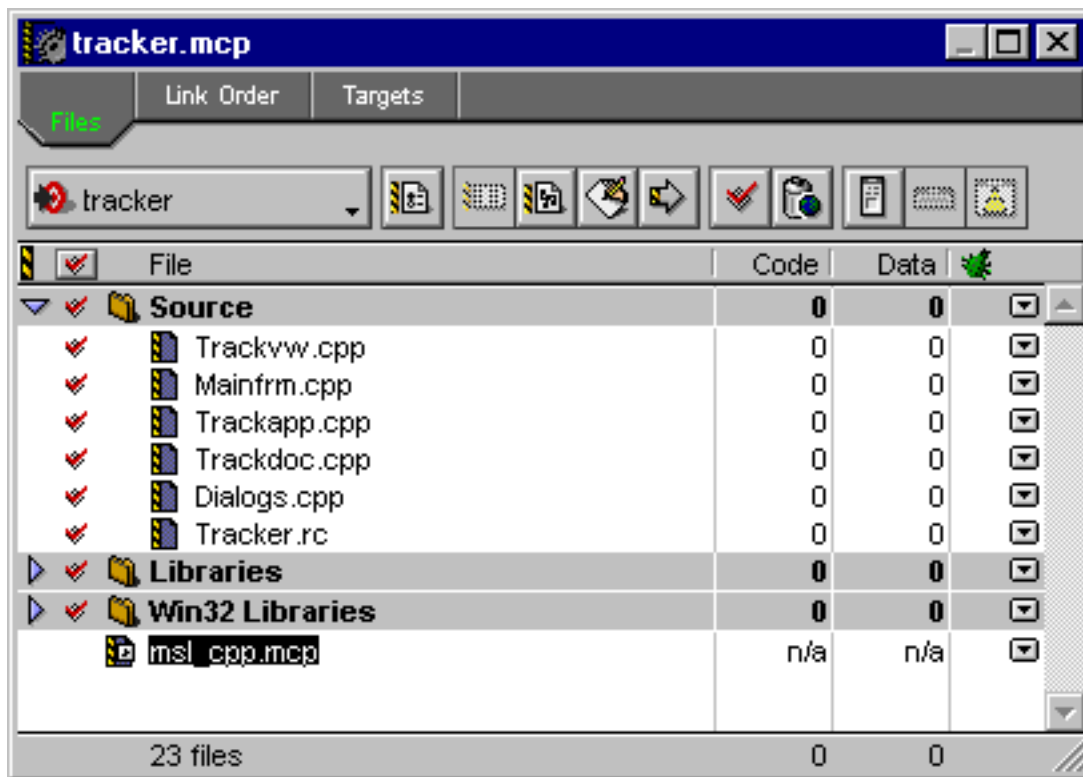
## Working with Projects

### Creating Complex Projects

---

the main project file. This allows you to factor the build process into separate project files.

**Figure 3.13 Subproject Within a Project**



One case where this factoring might be useful is for applications that utilize a code plug-in architecture. Suppose your program uses many different plug-in code modules, each sharing some common source code with other plug-ins. You can create one project file to build all the plug-ins by creating a separate target for each plug-in. In this scenario, the project file for the plug-ins is the subproject. Including the subproject in the project file of the main application allows all the plug-ins to be built, before building the main application.

A project file may be assigned to any target in a project. To learn how to do this, refer to [“Assigning Files to Targets” on page 91](#).

To learn how to add files to a project, refer to [“Adding Files” on page 69.](#)

You may select one or more targets in a subproject, to be built when the containing target in the main project is built. When the target in the main project is built, the CodeWarrior IDE first builds any selected targets in any subprojects. You can optionally link the output code of the main project against the output code of the subproject's target. A subproject's targets are not built automatically when a subproject is added to a parent project. Only the chosen targets within the subproject will be built.

Subprojects can be made target-specific. That is, if you add a subproject, you can choose which targets it belongs to. Other targets in the main project will not build the subproject unless the subproject file is added to the target you choose.

To learn more about considerations for using subprojects, refer to [“Strategy for Creating Complex Projects” on page 83.](#)

## **Strategy for Creating Complex Projects**

The choice of whether to use multiple targets or subprojects within a project file depends on what works best for you. If you want access to all the source code in one project, then multiple targets is a good choice. Subprojects are better when you prefer to keep separate stand-alone project files.

For example, if you need to build a number of plug-in shared libraries that accompany your application, create a project that builds the subprojects with a single [Make](#) command. Then, include this project file as a subproject in your main application project file. When your main application is built, the subproject's plug-ins will be built first.

There is a limit of 255 targets per project. Before you hit that limit, there's the consideration of memory and project load times. Projects with lots of targets will take up more disk space, take longer to load, and use more memory.

Once you get past ten or twenty targets, it's likely that you would benefit by moving some of them off to subprojects. Anything that is

not built often and uses a distinct set of source files is a good candidate for moving to a subproject.

## Creating a New Target

To create a new target in your project, use the [Create New Target](#) command in the [Project Menu](#). This command appears if you have the Targets category selected in your project window, as shown in [Figure 3.12 on page 81](#).

After you choose this menu command, you see the dialog box as shown in [Figure 3.14](#). In this dialog box, you can choose the name of the new target using the Name For New Target editable text field.

Then, choose whether you want your new target to be empty, or a clone of a previous target. If you choose Empty Target, you need to configure all the settings of the target as if a new project window were just opened. If you choose Clone Existing Target, the settings for the new target are the same as those of the target that you chose from the pop-up menu. You also get a copy of all the files that the original target contains.

After creating a target, you may want to associate the target with other targets, in order to create dependent build relationships. To learn about how to do this, refer to [“Creating Target Dependencies” on page 88](#).

To learn how to configure settings for a target, refer to [“Choosing Target Settings” on page 244](#).

When creating a new target that depends on using an output file from another target, you will need to click in the Link Column for the target that creates the output file. For an example of this, refer to [Figure 3.15](#). Here the target named Muscle Fat depends on both Muscle 68K, Muscle Resources and Muscle PPC being built first.

After these targets are built, Muscle Fat can be built and linked using the output files from the other two targets.

Figure 3.14 New Target Dialog Box

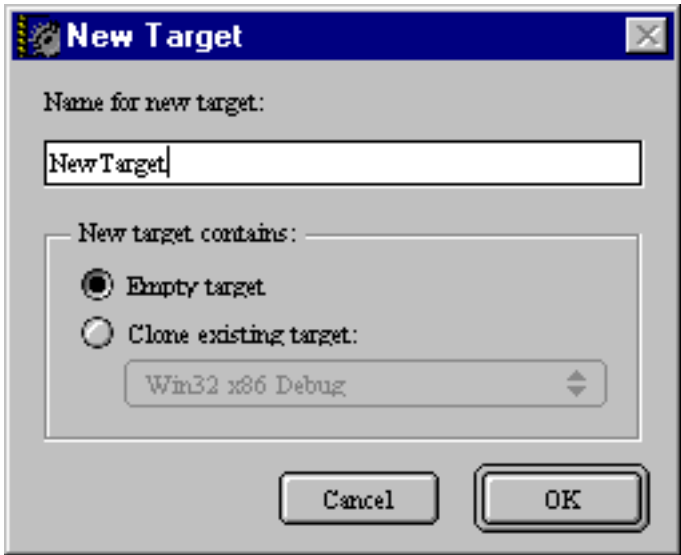


Figure 3.15 Link Column Dependencies



Link Column

## Changing a Target Name

To change the name of a target in the Targets category of the project window, as shown in [Figure 3.12 on page 81](#), just double-click on the name of the target. After that, make sure that the [Target Settings](#) panel is chosen, using the [Target Settings](#) menu command on the [Edit Menu](#). Change the name of the target using the Target Name editable text field.

You can also choose the [Target Settings](#) panel using the [Target Settings](#) menu command.

To learn more about the [Target Settings](#) panel, refer to [“Target Settings” on page 255](#).

## Changing the Target Settings

Each target in a project has its own [Preferences](#) and [Target Settings](#). To learn how to change these, refer to [“Configuring IDE Options Overview” on page 215](#).

While you can configure the [Preferences](#) and [Target Settings](#) for the target, there are some target settings that may be of immediate interest to you. Those settings are covered here in this section.

To change the settings of a target in the Targets category of the project window, as shown in [Figure 3.12](#), just double-click on the name of the target. After that, make sure that the [Target Settings](#) panel is chosen, using the pop-up menu. You can also choose the [Target Settings](#) panel using the [Preferences](#) menu command, in the Target category panel, instead of double-clicking.

Change the name of the target using the Target Name editable text field. To change the Linker or Post Linker used for the target, use the pop-up menus. Refer to the appropriate targeting manual, as detailed in [Table 1.2 on page 25](#), for the appropriate settings for your project.

To learn more about the [Target Settings](#) panel, refer to [“Target Settings” on page 255](#), or your targeting manual.

## Setting the Current Build Target

You can choose a different target within the current project to work with, using the [Set Current Target](#) command under the [Project Menu](#). This command might be useful if you want to switch between multiple targets in a project, and do a build for each one.

You can also change the current target by using the Targets category of the Project window, as shown in [Figure 3.12 on page 81](#). The current targets that will be built are denoted by the circle icon (an archery “target”) with an arrow going into it. To change to a different target for building, click once on the name of the target you want to choose as the current target.

## Creating Target Dependencies

You can configure your targets so that one target depends on another. That way, if you build a target, and the target depends on another target being built first, the second target will be built before building the first target.

[Figure 3.16](#) shows an example project window that contains two targets that do not have any dependencies. If you don’t know how to



create new targets, refer to [“Creating a New Target” on page 84](#) to learn how.

**Figure 3.16 Multiple Targets Without Dependencies**



[Figure 3.17](#) shows the finished result for the following discussion. Suppose that you wanted the Resources and Cool App Shared Libraries targets to be built before building the CoolApp target. One reason why you might want to do this is perhaps the Cool App target uses the output of the Resources and Cool App Shared Libraries targets at link time. You want these targets to be up to date before linking your Cool App target. To make Resources a target that Cool App depends on, just use drag and drop to configure the dependency. To do this, click on the Resources target and drag it on top of Cool App, then release the mouse button. Do the same thing for the other targets. [Figure 3.17](#) shows what this look like when you are finished doing this.

Click on the disclosure triangles on the left side of the screen to expand the dependency information for a given target. [Figure 3.17](#)

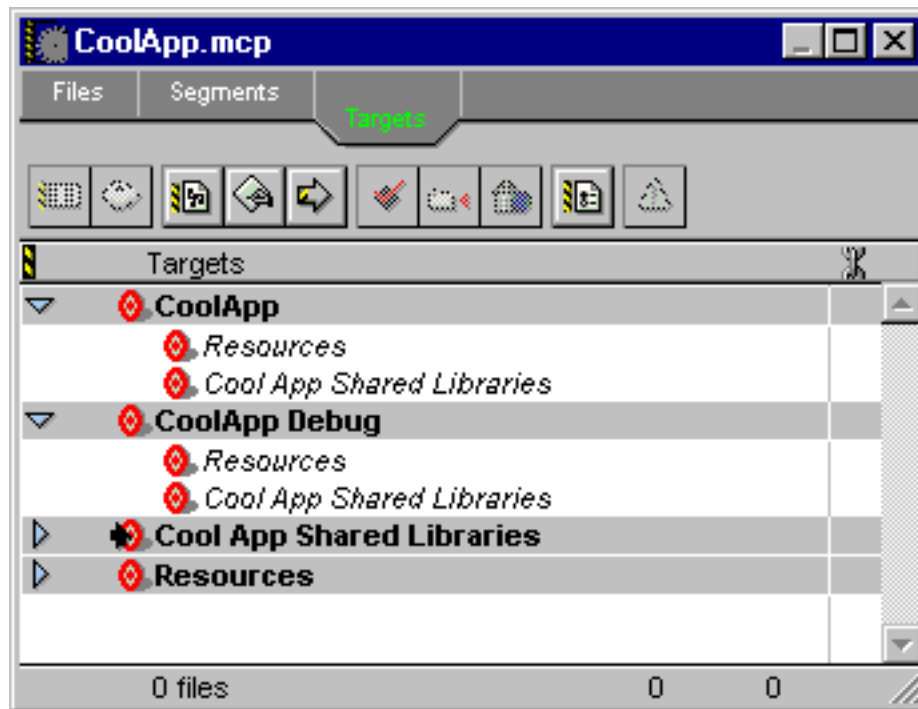
## Working with Projects

### Creating Complex Projects

---

shows what a window will look like after you have clicked some of the disclosure triangles to show the targets that are dependents.

**Figure 3.17 Multiple Targets with Dependencies**



If we then made Cool App the current target, and did a [Make](#) of it, then the Resources and Cool App Shared Libraries targets will be built first. This is because Cool App is now dependent on these targets.

If this were a real-world programming task for you, it might be a good idea to also have debug targets too.

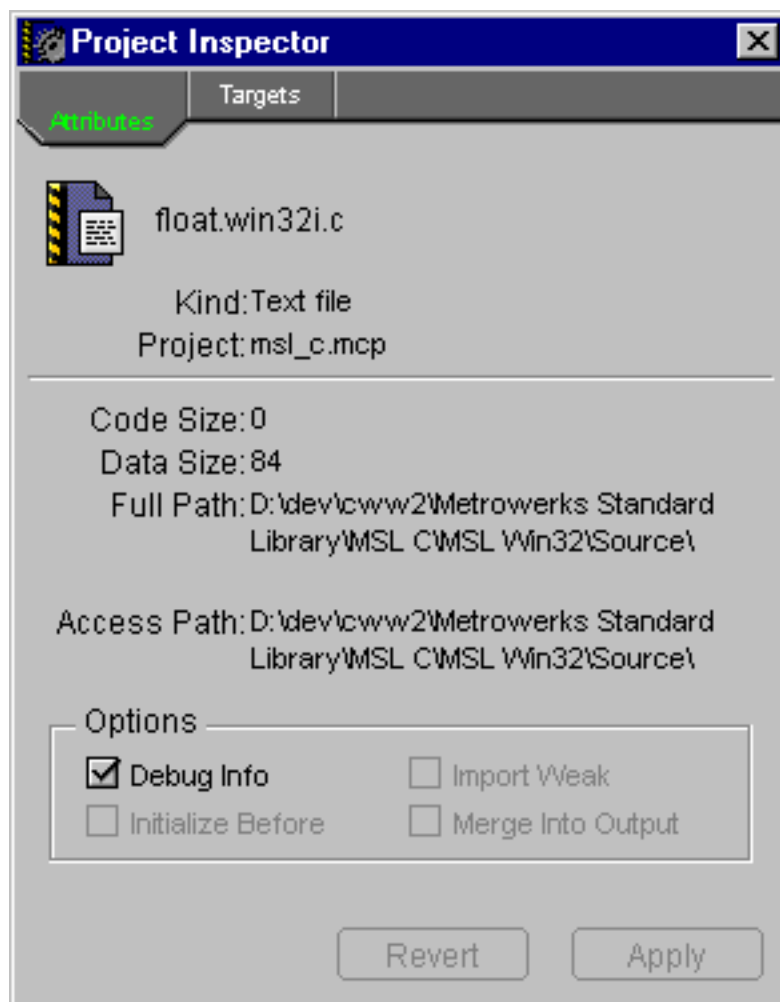
Refer to [“Setting the Current Build Target” on page 88](#) to learn how to set the current target before building if you don’t already know how to do this.

To learn more about strategies for setting up complex projects using targets and subprojects, refer to [“Strategy for Creating Complex Projects” on page 83](#).

## Assigning Files to Targets

You can select the target that a file belongs to using the [Project Inspector](#) window. First, select a file in the project window. To learn how to do this, refer to [“Selecting Files and Groups” on page 67](#). Then, choose the [Project Inspector](#) menu command from the [Window Menu](#). A window appears, as shown in [Figure 3.18](#).

**Figure 3.18** Project Inspector Window for Attributes



Click the Targets category tab to switch the view to that shown in [Figure 3.19](#). This window is showing you which targets your select-

## Working with Projects

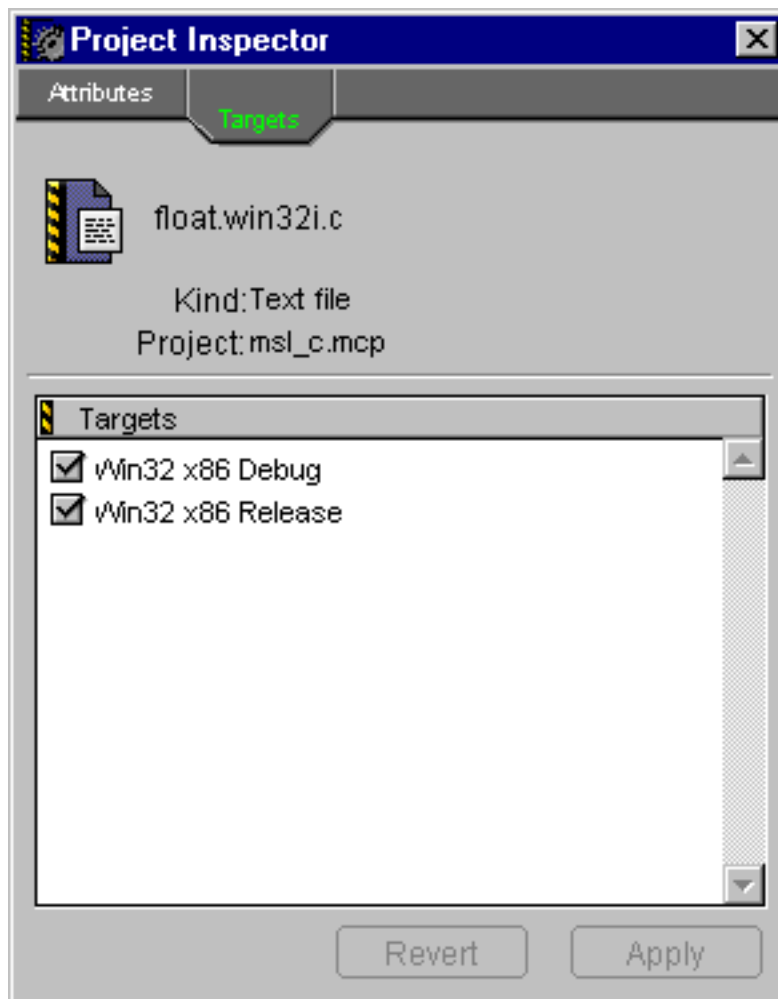
### Creating Complex Projects

---

ed file belongs to. If you want to change so that a file belongs to different targets, just click in the check boxes on the left side of the window to include or exclude the file from a given target.

You may close the window when you are finished by clicking the close box. If you make changes that you want to undo, click the Revert button. If you want to apply your changes but keep the window open, click the Apply button.

**Figure 3.19** Project Inspector Window for Targets

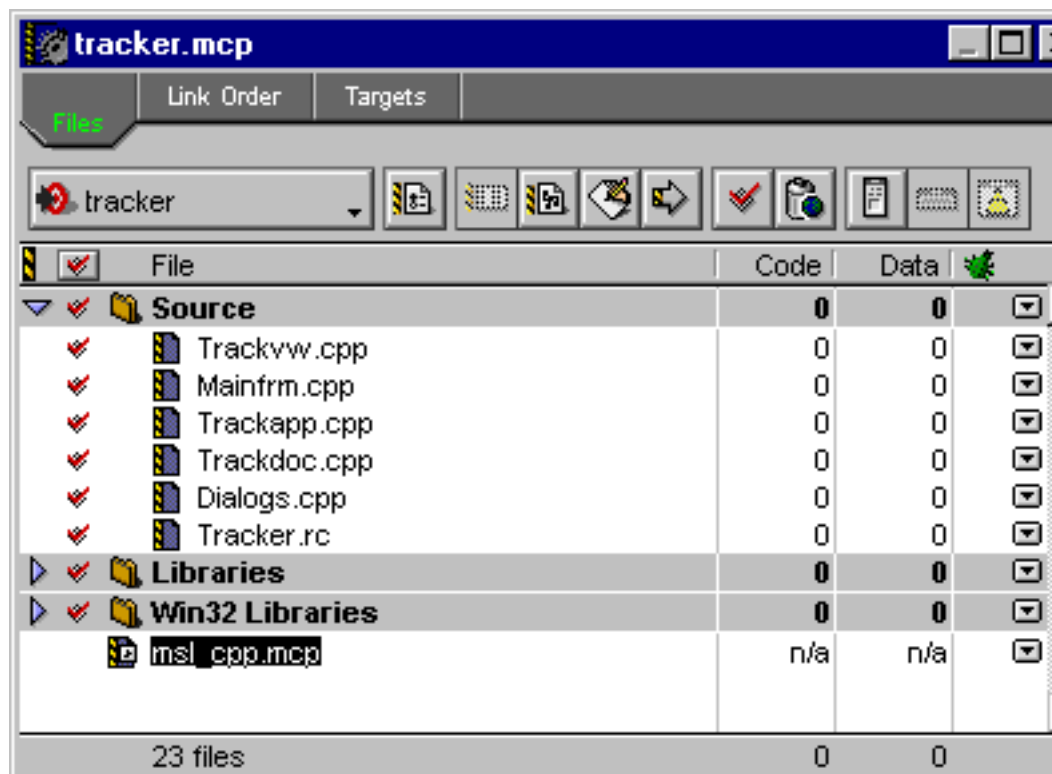


In the example of [Figure 3.19](#), the file `float.win32i.c` belongs to all targets. It will only be compiled when building for the targets that are checked.

## Creating Subprojects Within Projects

To create a subproject, just drag and drop a project file into an open project window. Note that if your main project file contains multiple targets, you will be prompted to choose the targets that the subproject should be added to. After you do this, the project window will look similar to [Figure 3.20](#), with a project file added to the list of files in your project. In this example, the file is named `tracker.mcp`. You may also add the file using the methods discussed in [“Adding Files” on page 69](#).

Figure 3.20 Subproject Within a Project



This causes the subproject file to become part of the main project file. When you do a [Make](#) on the main project, the subproject will be built first.

Once you add a subproject to a project, you can assign which targets the subproject is used in. To learn how to do this, refer to [“Assigning Files to Targets” on page 91](#).

## Examining Project Information

The CodeWarrior IDE allows you to review and configure information about your project and source code files, using the [Project Inspector](#) window, shown in [Figure 3.21](#). To show this window, choose the [Project Inspector](#) command from the [Window Menu](#).

There are 2 tabbed categories in this window, Attributes and Targets.

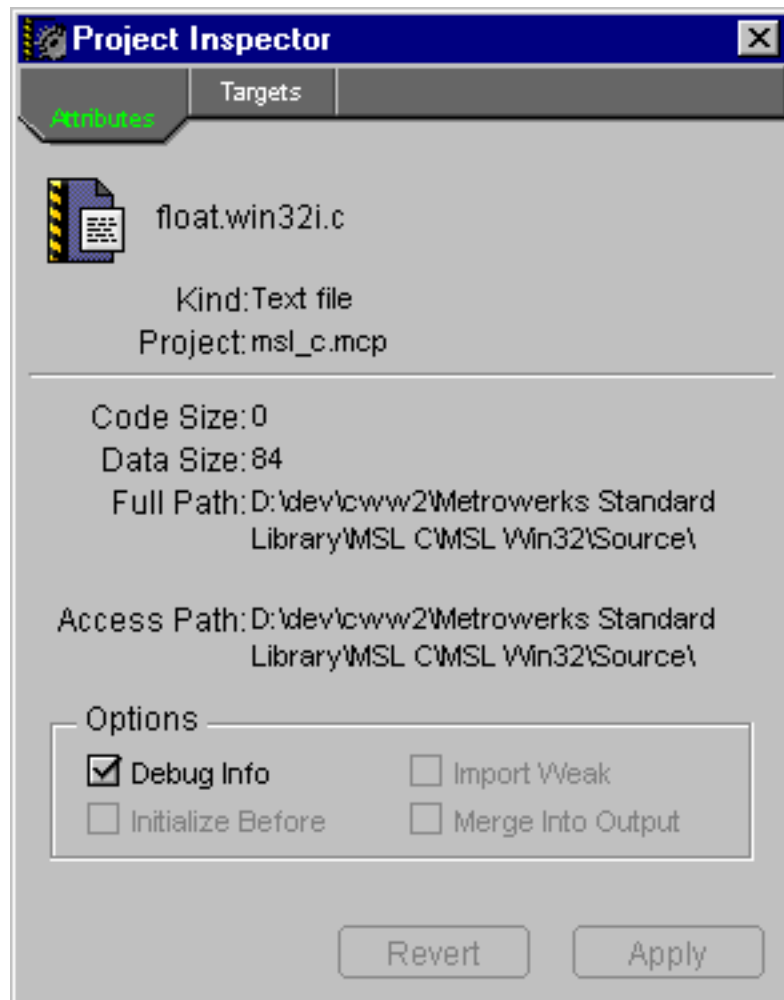
To learn how to inspect and set the Targets for which a given file will be compiled, refer to [“Assigning Files to Targets” on page 91](#), as that section of the manual discusses the Targets category in detail.

To learn more information about a given file in your project, you can review its Attributes category settings. The Attributes include the file name, its path to a location on your hard disk, the name of the project it is included in, the code and data size (if it has been compiled), and whether or not Debug Info is being generated for the file when compiled.

You may close the window when you are finished by clicking the close box. If you make changes that you want to undo, click the Re-

vert button. If you want to apply your changes but keep the window open, click the Apply button.

**Figure 3.21 Project Inspector Window**



To learn about configuring Debug Info for files, refer to [“Controlling Debugging in a Project”](#) on page 96.

## Moving a Project

Previous versions of the Windows-hosted CodeWarrior IDE used a folder called Resource.frk to store state information for the project file. Resource.frk is now obsolete and will no longer be generated or utilized by the CodeWarrior IDE.

The CodeWarrior IDE stores all required information about your project in the project file. There are other files that are stored on the hard disk along with the project file. These other files, usually stored in a folder having a similar name to your project file, are not needed by the CodeWarrior IDE to recreate your project. The files contain information about window positions, object code, debug info, browse data, and other settings which are not crucial to rebuilding the code of your project. Some of the files are needed for caching information about which files have already been compiled, so that only the files that changed are compiled each time you build.

To move your project on your hard disk, just copy the project file (ending in .mcp if it obeys the project file naming convention) to a new location on your disk. If you want all the information in the additional files to travel with the project file, you may also copy the folder containing those files. However, these files are not needed. The CodeWarrior IDE will be able to reconstruct its state when a [Bring Up To Date](#) or [Make](#) operation is performed. Generally, you would only check the main project file into a revision control system, and not the other files.

If you have set absolute [Access Paths](#), you may need to modify them when you move your project file. To learn how to do this, refer to [“Access Paths” on page 247](#).

## 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” on page 268](#) and [“Choosing Target Settings” on page 244](#) 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, as shown in [Figure 3.22 on page 99](#). When you add files to your project, CodeWarrior automatically sets this marker unless you have deselected the appropriate option in the [x86 Linker](#) or other [Target Settings](#) panel.

You can also set Debug Info generation in the [Project Inspector](#) window. To learn how to do this, refer to [“Guided Tour of the Project Window” on page 58](#). When files are added to the project, their debug state is set by the state of the [Enable Debugger](#) and [Disable Debugger](#) menu command setting in the [Project Menu](#).

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.

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. Whenever a Debug Info Marker is changed, the file will be marked for recompilation the next time your project is built.

## Working with Projects

### *Controlling Debugging in a 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](#) contains the option that enables CodeWarrior to create a debugging file.

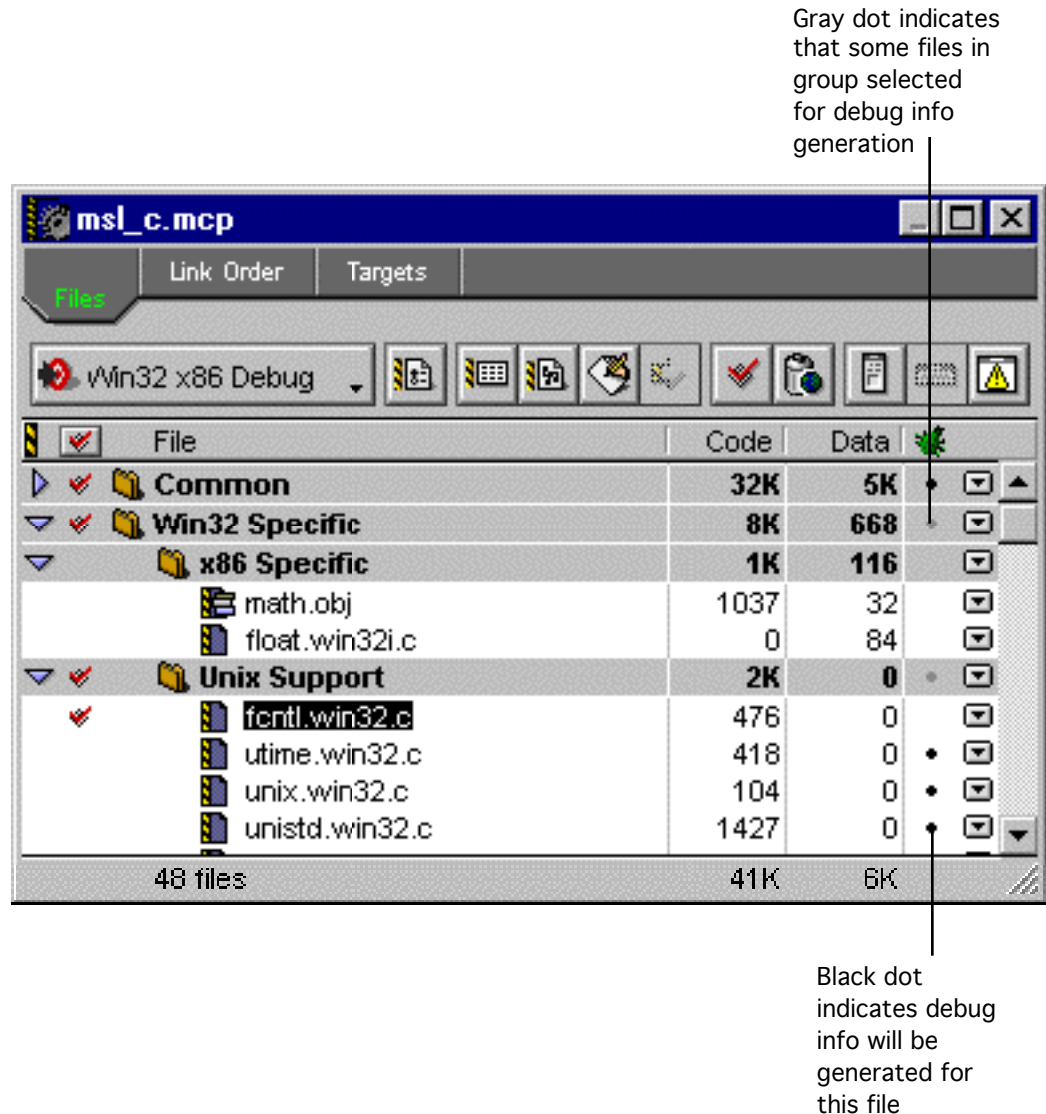
---

#### Debug Info Marker for Groups

The Debug Info marker also appears on the right end of group rows ([Figure 3.22](#)), and can be toggled on and off by clicking. The [Debug Column](#), for groups, may show one of three markers:

- **Black marker:** all files in the group generate debugging information
- **Grey marker:** only some of the files in the group generate debugging information.
- **No marker:** means no debugging info for all files in group.

**Figure 3.22**    **Debug Info Markers**



## 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:

## Working with Projects

### *Adding Preprocessor Symbols to a Project*

---

```
#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 Preprocessed Headers” on page 281](#) for more information on how to do this.

To learn more about this topic, refer to the *C Compiler Guide* manual, and the *CodeWarrior Pascal Compiler Guide* for more information.



# Working with Files

---

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

## Working with Files Overview

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

To learn about editing files, refer to [“Source Code Editor Overview” on page 117.](#)

To learn about working with files using revision control systems, refer to [“Version Control System Overview” on page 305.](#)

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, a text insertion point is placed on the first line of the window. When you begin typing, the CodeWarrior IDE 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 117.](#)

## 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](#)
- [Opening a Related File](#)

---

**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.

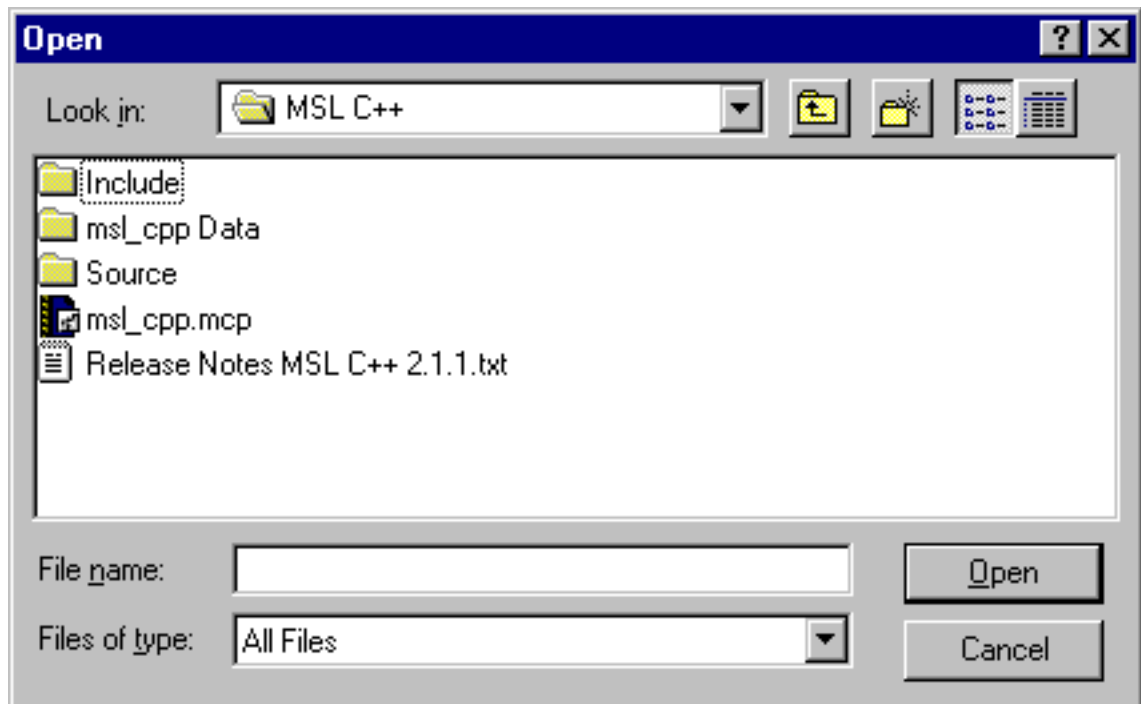
Click on the project file you would like to open, then click the Open button. The CodeWarrior IDE then opens the project. You may keep multiple project files open at a time.

For more information about working with CodeWarrior project files, see [“Projects Overview” on page 45.](#)

### 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 All 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 box



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

For more information about editing source code, see [“Source Code Editor Overview” on page 117.](#)

## 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:

- [File Column](#)—opening a file that is in the project
- [Group Pop-up Menu](#)—opening a text source file from within a collapsed group
- [Interfaces 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.

---

**TIP:** To open several files from the File Column at one time, hold down the Control key and click each file that you want to see. Then, double-click on one of the selected files.

---

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 67](#).

For more information about the File Column, refer to [“File Column” on page 62](#).

### Group Pop-up Menu

One way to open a source file is to click on the [Group Pop-up](#) for a particular group so that it pops up. A similar menu is shown in [Figure 4.2](#). From this pop-up menu you may select the file in the group that you want to open.

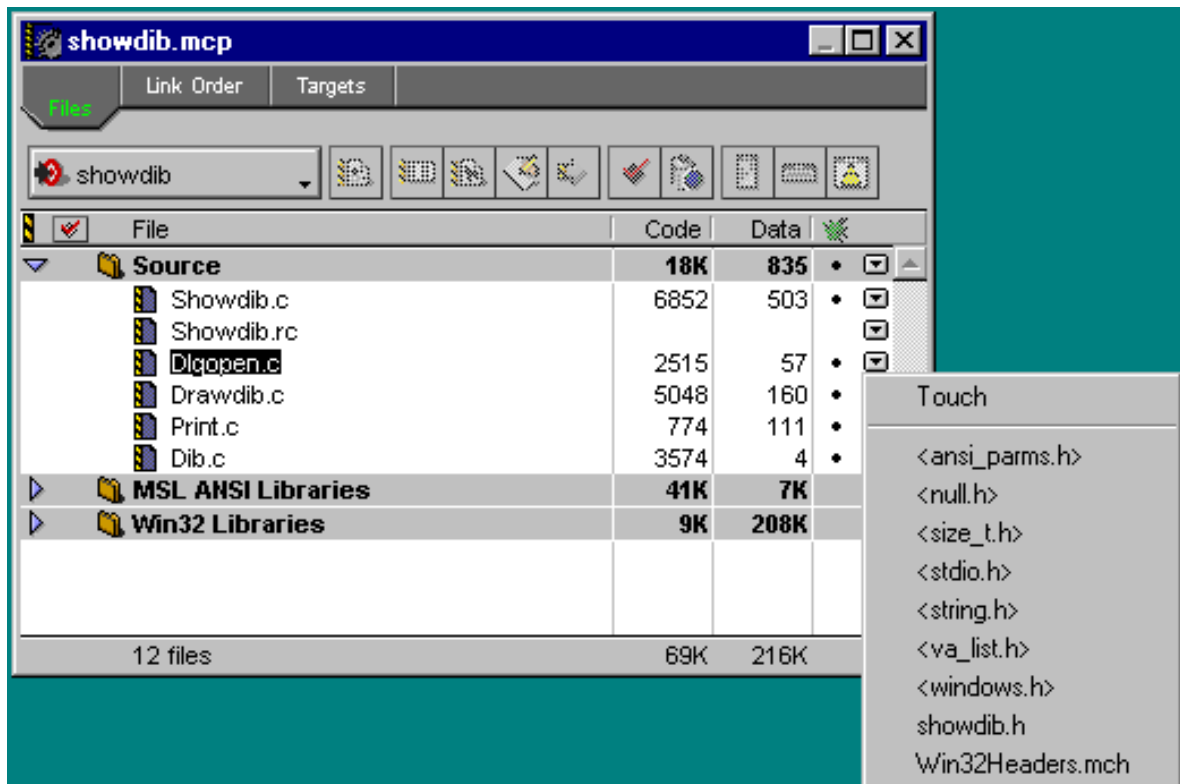


You can open a source file by choosing it from the [Group Pop-up](#) menu for the group that contains the file. This works even if the group is collapsed and the file is not visible in the project window.

### Interfaces Pop-up Menu

To open a header or interface file, click on the [Interfaces Pop-up](#) to see a list of files. Select the file you want to open from this list, as shown in [Figure 4.2](#).

**Figure 4.2** Interfaces Files Pop-up Menu in the Project window



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 or interface files, these files can not 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 on page 120](#). This pop-up menu lists all interface or header 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.

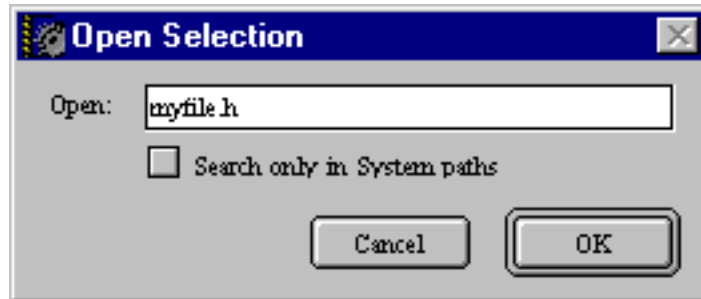
First, select text in the editor window containing the name of the interface file you would like to open. An example of a file name you might see in a C source code file is `stdio.h`. You could select `stdio.h` by double-clicking on the `stdio` portion of the text. Then, choose the [Open Selection](#)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

The CodeWarrior IDE 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](#) for the project to search for the file to open.

After you choose Open File, the CodeWarrior IDE then displays an open file dialog box, as shown in [Figure 4.3](#). Type the name of the file you wish to search for in the Open editable text field.

**Figure 4.3** Open Selection dialog box



To search only the CodeWarrior directory structure (the paths specified in the [System Include Path Pane](#) of the [Access Paths](#)), 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 paths (all paths specified in the [Access Paths](#)), turn the Search only in the System Tree option off.

If no project is open when you attempt an Open Selection operation, the CodeWarrior IDE 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” on page 247](#) for more information.

## Opening a Related File

If you are working in a source code file and wish to open the corresponding header file, or working with a header file and wish to open the corresponding source file, there is a shortcut for you. You can easily switch back and forth between the 2 files.

To learn how to do this, refer to [“Opening a Related File” on page 147](#).

## Saving a File

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

## Working with Files

### *Saving a File*

---

- [Saving One File](#)
- [Saving Files Automatically](#)
- [Renaming and Saving a File](#)
- [Backing Up Files](#)
- [Saving as a UNIX or DOS text file](#)

---

**NOTE:** When saving a file, you will often have the option of saving a file as a stationery file. A stationery file is like a template or “starter” file. For example, creating a stationery file would be useful if you had standard documentation header text that you wanted to appear at the top of every file you create. If you create a stationery file with the header text, you could start every new file by opening this stationery file.

---

### **Saving One File**

To save your changes to the current Editor file, choose the [Save](#) command from the [File Menu](#). The CodeWarrior IDE saves your file to your hard 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 exit CodeWarrior, or when the [Save A Copy As](#) command is selected. You don’t need to explicitly save projects.

---

**NOTE:** If the file is new and untitled, the CodeWarrior IDE displays the Save As dialog box, described in [“Renaming and Saving a File” on page 109](#). Choose a name and location for your new file with this dialog box.

---

### **Saving Files Automatically**

The CodeWarrior IDE 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 of this manual titled ["Editor Settings" on page 221](#).

### **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, the CodeWarrior IDE updates the project to use the new name.

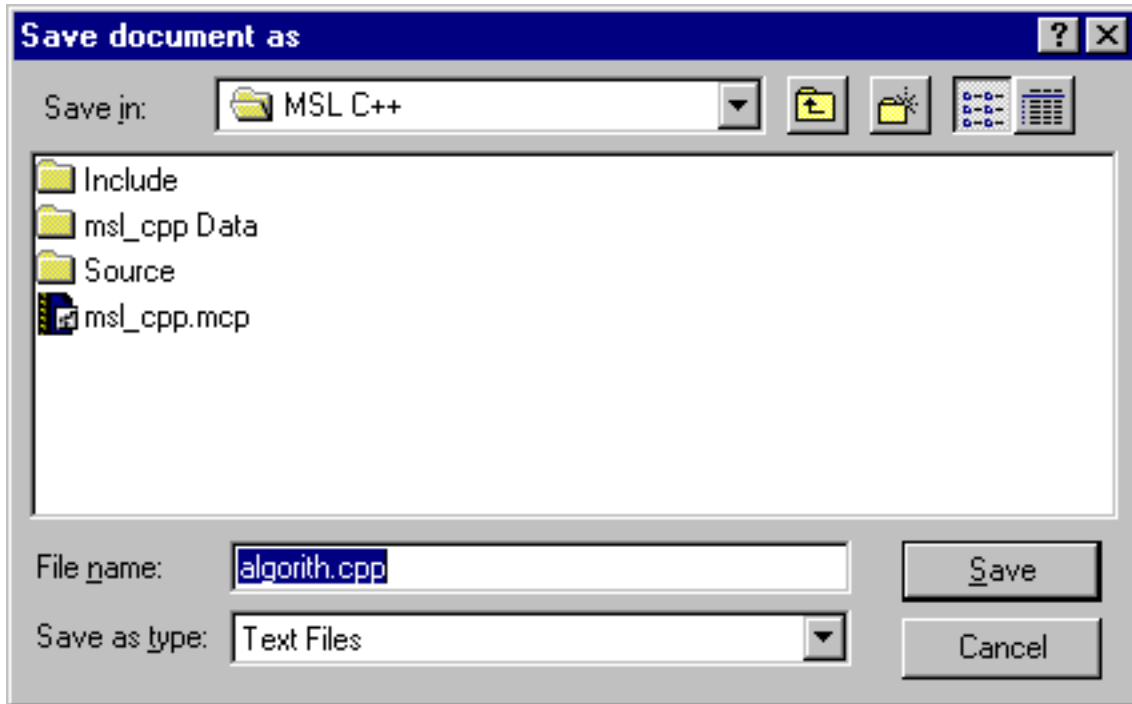
When you choose [Save As](#) from the [File Menu](#), the CodeWarrior IDE displays the dialog box shown in [Figure 4.4](#).

## Working with Files

### *Saving a File*

---

**Figure 4.4** Save As dialog box



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

The CodeWarrior IDE saves the file and changes the name of the editor window to the name you entered.

If the file is in the current project, the CodeWarrior IDE changes the file's entry in the project to match the saved name. If you don't want to change the project, but still want to save the file, you can read how to do this in ["Backing Up Files" on page 110](#).

### **Backing Up Files**

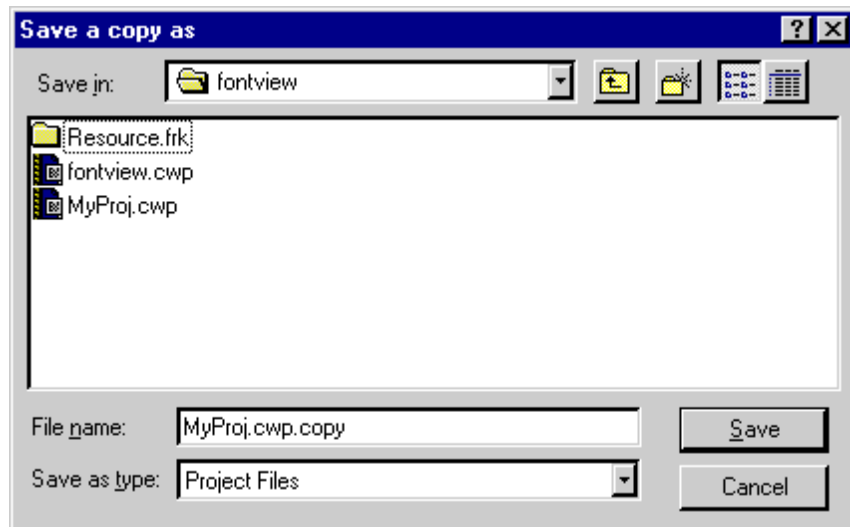
If you want to save a backup copy of a text file before you make some changes to the original, use the [Save A Copy As](#) command on the [File Menu](#). The CodeWarrior IDE creates a copy of the file under a new name that you specify, but leaves the original file unchanged and does not change the currently-open 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 4.4](#). 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 4.5](#). 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 to a new location on your hard disk, 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 96](#).

**Figure 4.5**    **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-com-

## Working with Files

### *Closing a File*

---

patible text file. When you save the file, CodeWarrior saves it in its original format.

You can use this feature to save a file in a different end-of-line format by selecting a new command from the menu.

## Closing a File

Every editor or project window in the CodeWarrior IDE 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 CodeWarrior IDE window.

The topics in this section are:

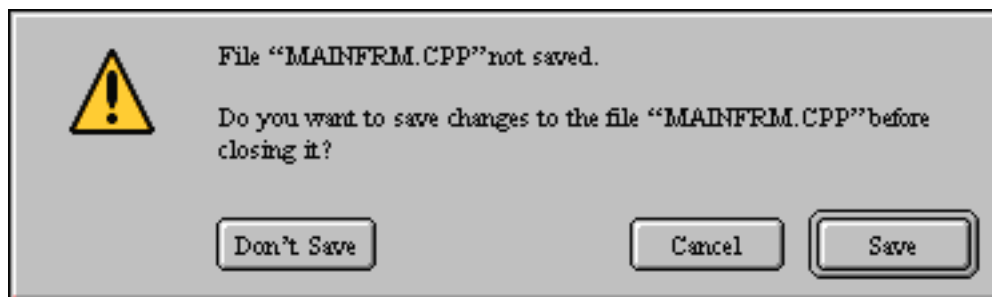
- [Closing One File](#)
- [Closing All Files](#)

### 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, the CodeWarrior IDE asks if you want to save the changes before closing the window, as shown in [Figure 4.6](#). If you choose to close the file without saving your changes, all changes are lost.

**Figure 4.6** The dialog box for unsaved changes





Another way to close a window is by clicking the close box of the active window. This is exactly the same as choosing the [Close](#) command in the [File Menu](#).

If the active window is the project window, closing the window automatically saves the project before the window closes, and you will not see the dialog shown in [Figure 4.6](#). For more on saving project files, consult [“Saving a Project” on page 56](#).

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” on page 221](#) 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 [Switch to MW Debugger](#) command on the [File Menu](#).

[Switch to MW Debugger](#) doesn't close all the CodeWarrior IDE windows, just editor windows. The Find dialog box and any Project windows 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 the CodeWarrior IDE to print open files, a project file, or the contents of a window.

The topics in this section are:

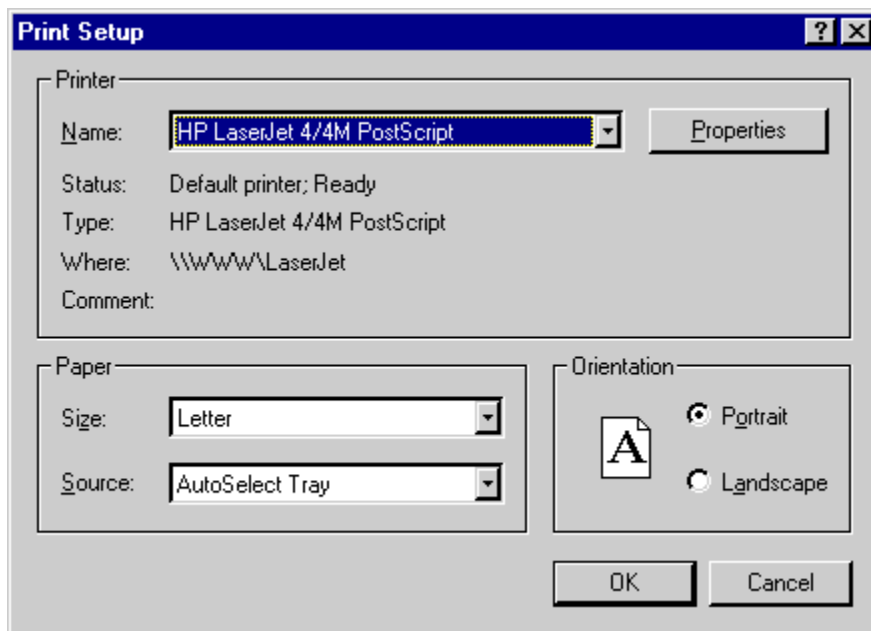
- [Setting Print Options](#)
- [Printing a Window](#)

## 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 4.7](#).

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.

**Figure 4.7** Print Setup dialog



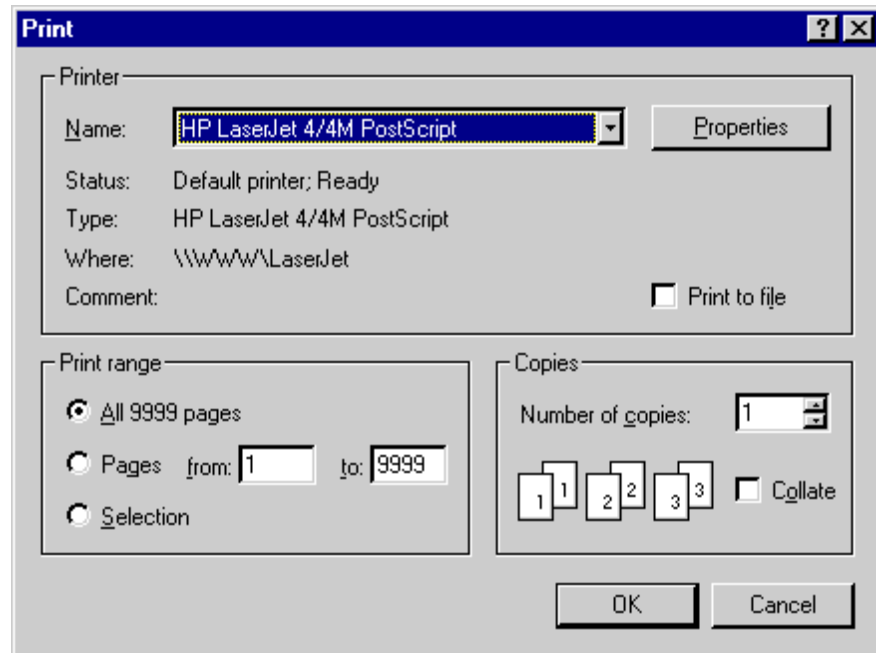
If you Click OK, CodeWarrior saves the options for the next time you print any files.

## 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 simi-

lar to that shown in [Figure 4.8](#) appears. Make any changes you require to the print settings, then click the OK button to begin printing your window.

**Figure 4.8** Print dialog



## Reverting to a Previously-Saved File

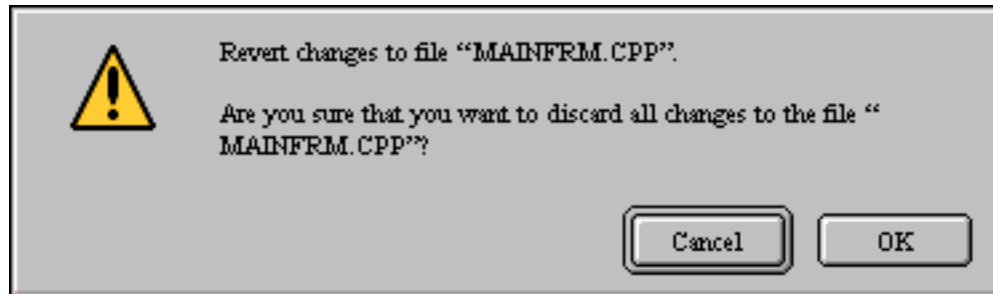
If you've opened a text file and started editing it, then realize that you don't want to use the changes you've made, use the [Revert](#) command on the [File Menu](#). When you select this command the dialog box shown in [Figure 4.9](#) appears.

## Working with Files

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

---

**Figure 4.9**    **Revert to a Previous 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 IDE 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.
- Syntax highlighting formats source code 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 customize options that affect the way the CodeWarrior Editor works. To learn more about how to do this, refer to [“Editor Settings” on page 221](#).

## Guided Tour of the Editor Window

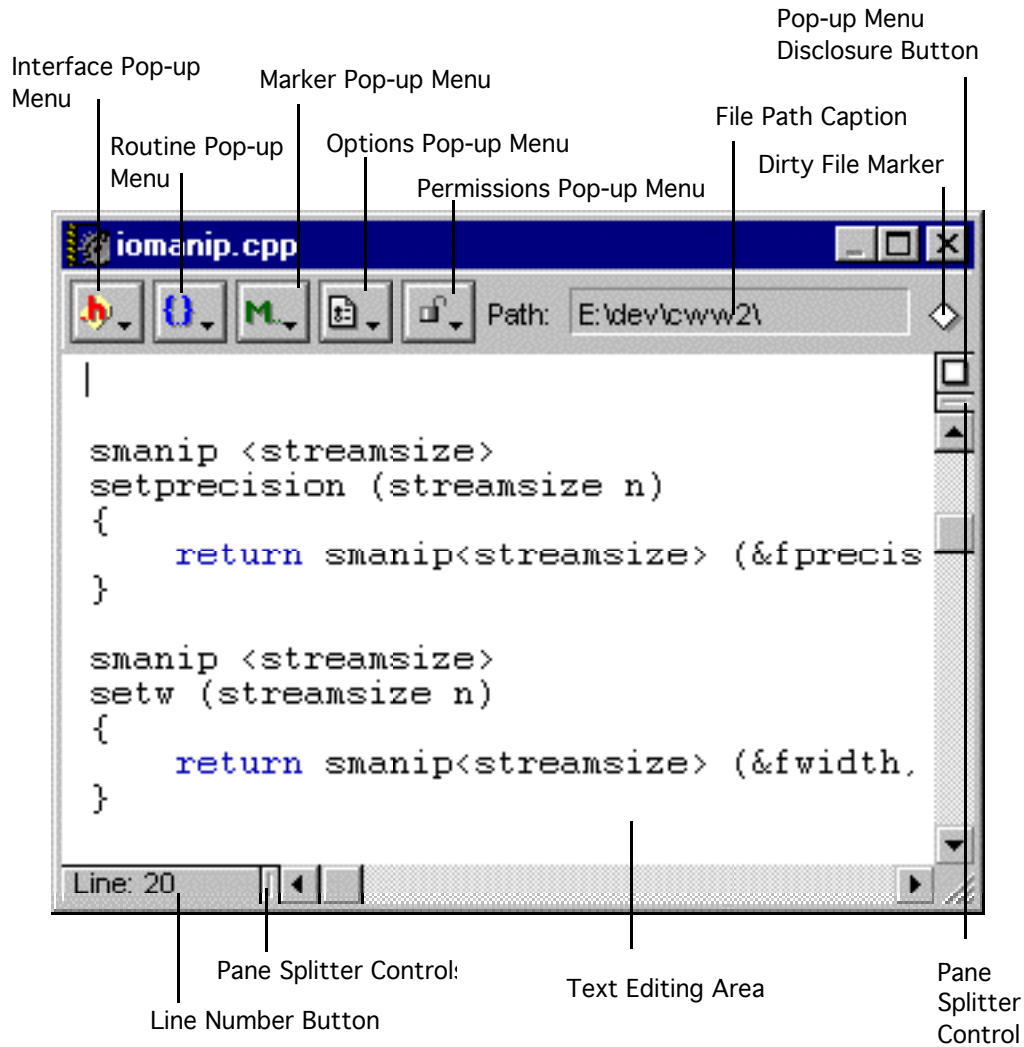
The CodeWarrior Editor window, shown in [Figure 5.1](#), contains elements you’ll find useful when viewing and editing your source files.

## 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](#)
- [Line Number Button](#)
- [Pane Splitter Controls](#)
- [Pop-Up Menu Disclosure Button](#)
- [Dirty File Marker](#)

## 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 136](#).

## Interface Pop-Up Menu



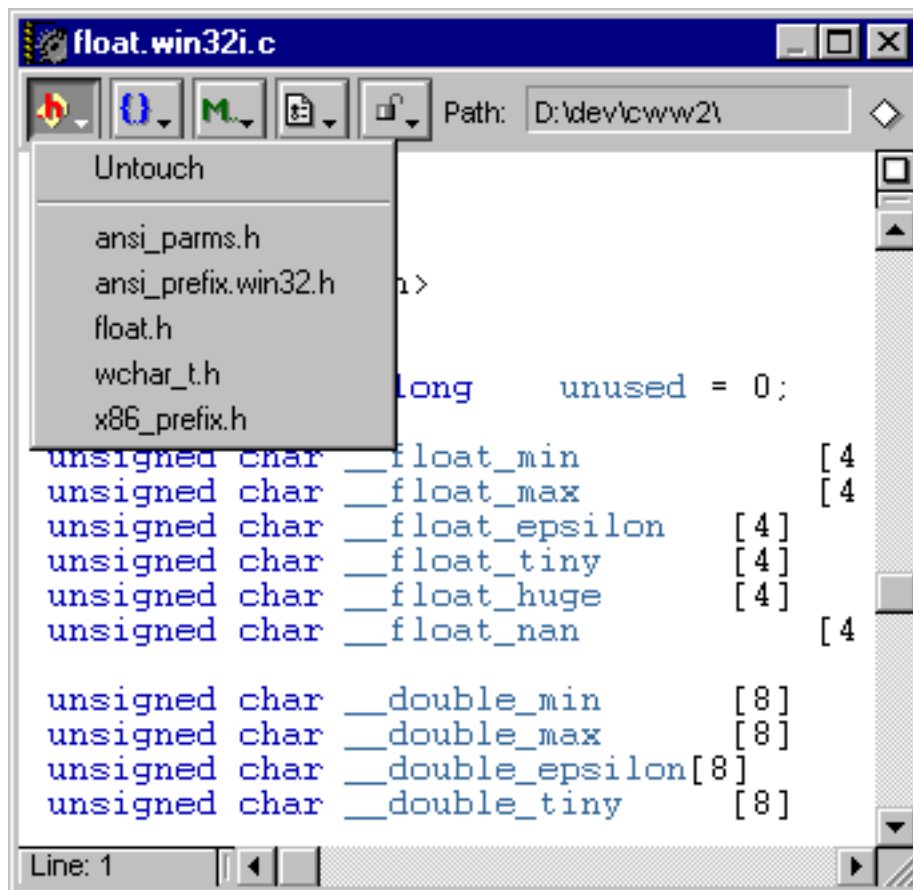
Use the Interface Pop-up Menu shown in [Figure 5.2](#) to open interface or header files referenced by the current file. You can also use the Touch and Untouch commands from this pop-up.

## 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, and libraries.

For more information on opening files, see ["Opening an Existing File" on page 102.](#)

To cause your file to be recompiled the next time the project is built, you choose the Touch command, or click on the [Dirty File Marker](#) so that it becomes checked. If you click on the Interface pop-up again you can deselect the file for compilation with the Untouch command on the menu.



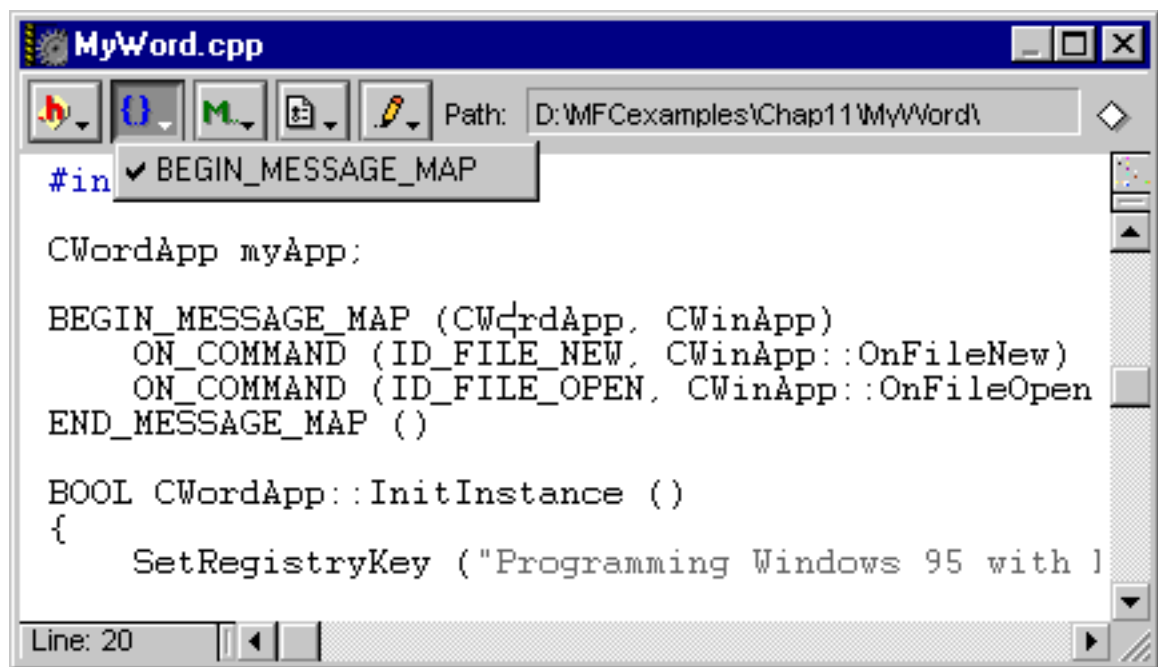
For more information on Touch and Untouch, see [“Touching and Untouching Files” on page 274](#).

## 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.

## Editing Source Code

### *Guided Tour of the Editor Window*

---

If you'd like to change the default display order of the routines to alphabetical, enable the [Sort Function Popup](#) option. See [“Editor Settings” on page 221](#) for more information about editor options.

---

**TIP:** If you're editing a Pascal file, the Routine Pop-up Menu displays procedure names in *italics*, function names are in plain face, and the main program is in **bold**.

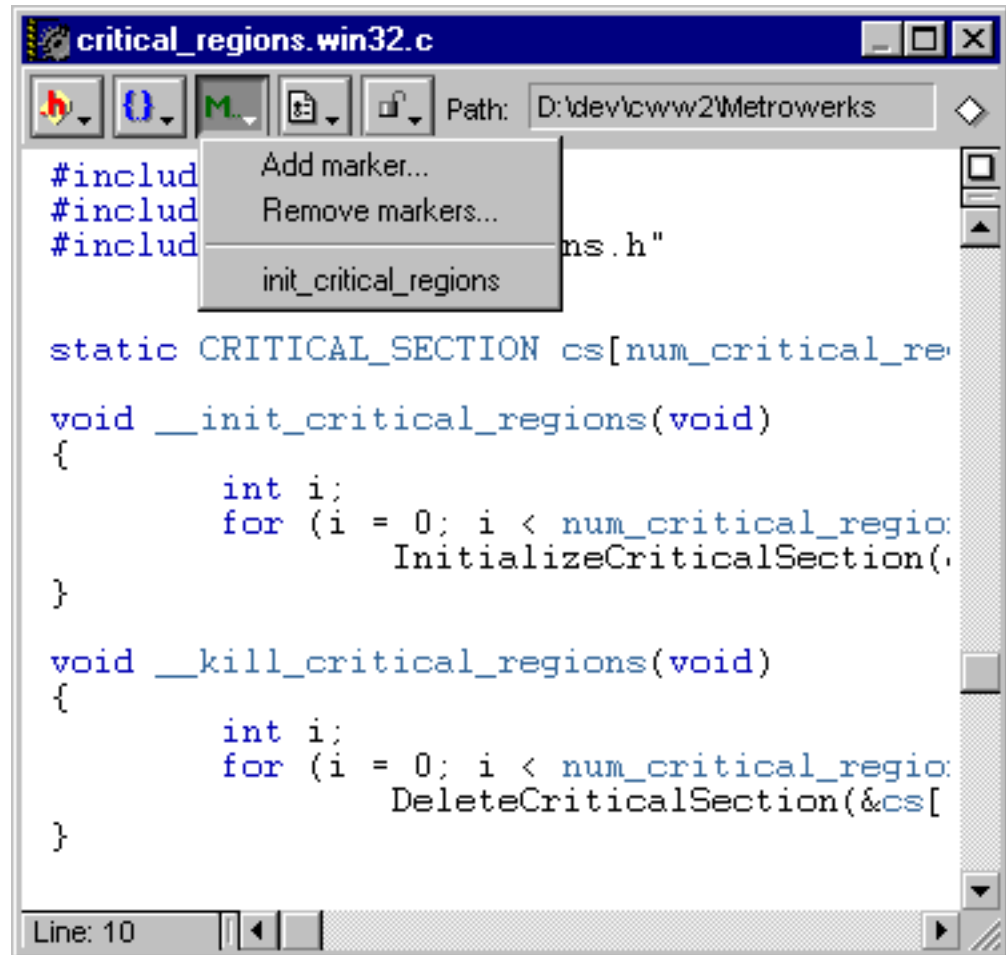
---

## 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.

**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 144](#).

## Options Pop-Up Menu



Use the Options Pop-up Menu, shown in [Figure 5.5](#), to choose syntax highlighting 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-

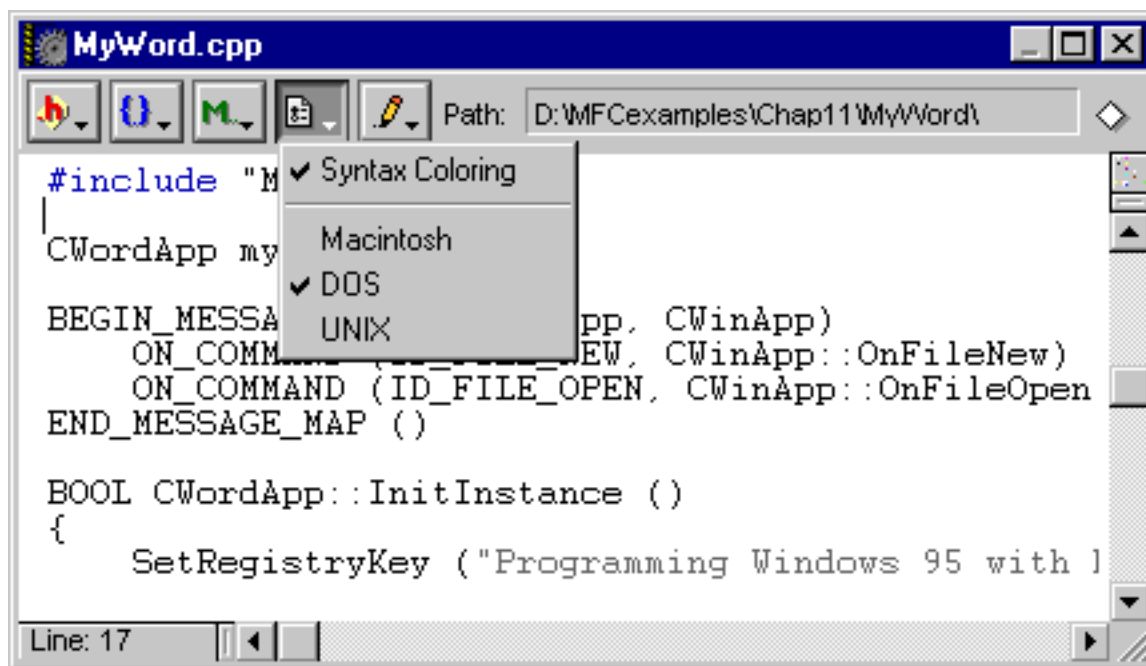
## Editing Source Code

### *Guided Tour of the Editor Window*

---

rent file type. The next time you save the file, the CodeWarrior IDE 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's end-of-line formatting (UNIX, DOS or Macintosh), and also enable or disable syntax coloring for the window.

For more information on how to use the end-of-line formatting options on this pop-up menu, see [“Saving as a UNIX or DOS text file” on page 111.](#)

For more information on the Syntax Coloring option shown in this menu, see [“Syntax Coloring” on page 226.](#)

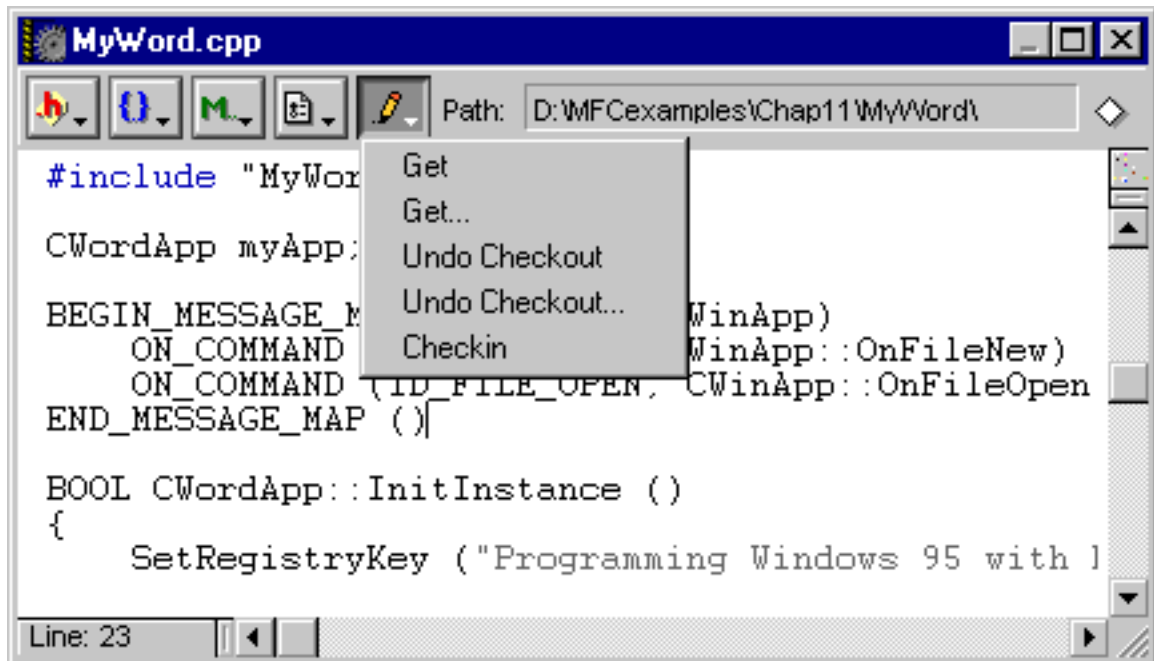
## File Path Caption

The CodeWarrior IDE 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 118.](#)

## Permissions Pop-up Menu

The Permissions Pop-up Menu, shown in [Figure 5.6](#), indicates the read/write revision control database status of the current file. If the pop-up icon box has an Unlocked icon displayed, or has the Read/Write icon displayed, you can modify the file you're working with. The icons and their meanings are described in the section called ["Using Source Code Control with Files" on page 305](#).

**Figure 5.6** Permissions Pop-up Menu



For more information about revision control system software, see ["Using Source Code Control with Files" on page 305](#).

## 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 149](#).

## 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 5.9 on page 130](#) shows an editor window with multiple panes.

For more information on this topic, see [“Splitting the Window into Panes” on page 129](#).

## 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 5.8 on page 129](#)).

For more information on using the Pop-up Menu Disclosure Button, refer to [“Seeing Window Controls” on page 127](#).

## Dirty File Marker

The Dirty File Marker tells you whether the file you are viewing is marked for compilation. Another term for Dirty File Marker might be Touch Marker. If a file is “touched,” that means it is marked for compilation. You can click on this marker to “touch” a file.

To learn more about marking files for compilation, refer to [“Touching and Untouching Files” on page 274](#).

# 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](#)
- Customizing the Toolbar

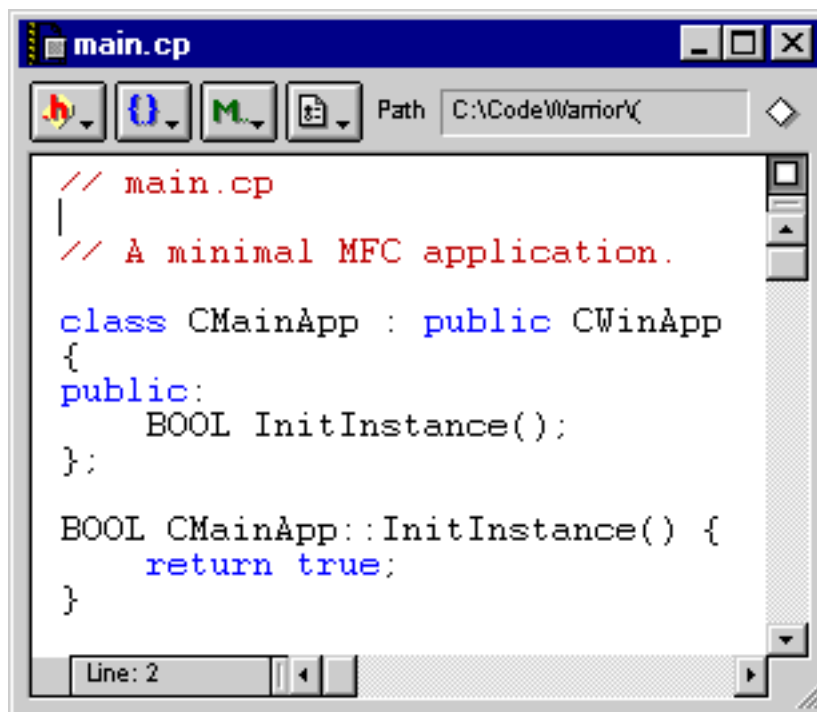
## Setting Text Size and Font

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” on page 225](#).

## 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 5.7](#).

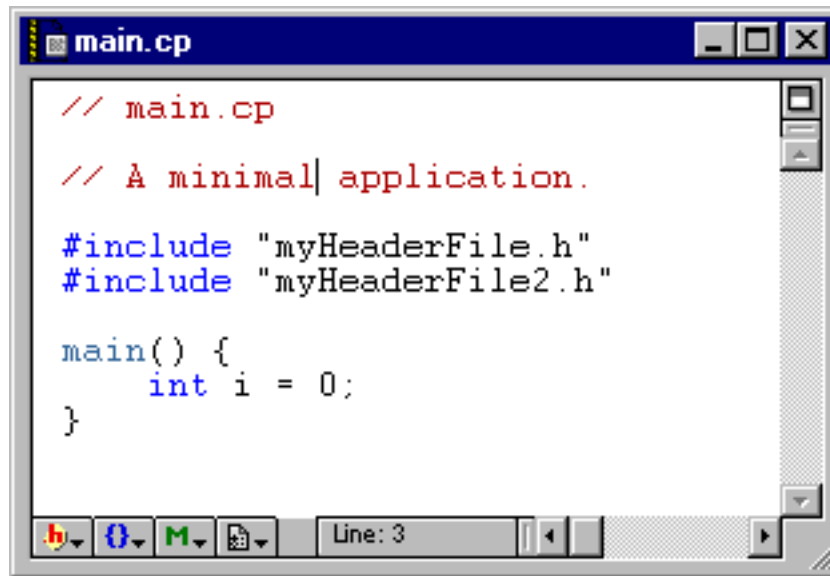
**Figure 5.7** Editor Window When Clicking the Pop-up Disclosure Button



Clicking this button changes the editor window to look like [Figure 5.8](#). Note that the [File Path Caption](#) is no longer visible.



**Figure 5.8** Pop-ups Along the Editor Window Bottom (After Clicking)



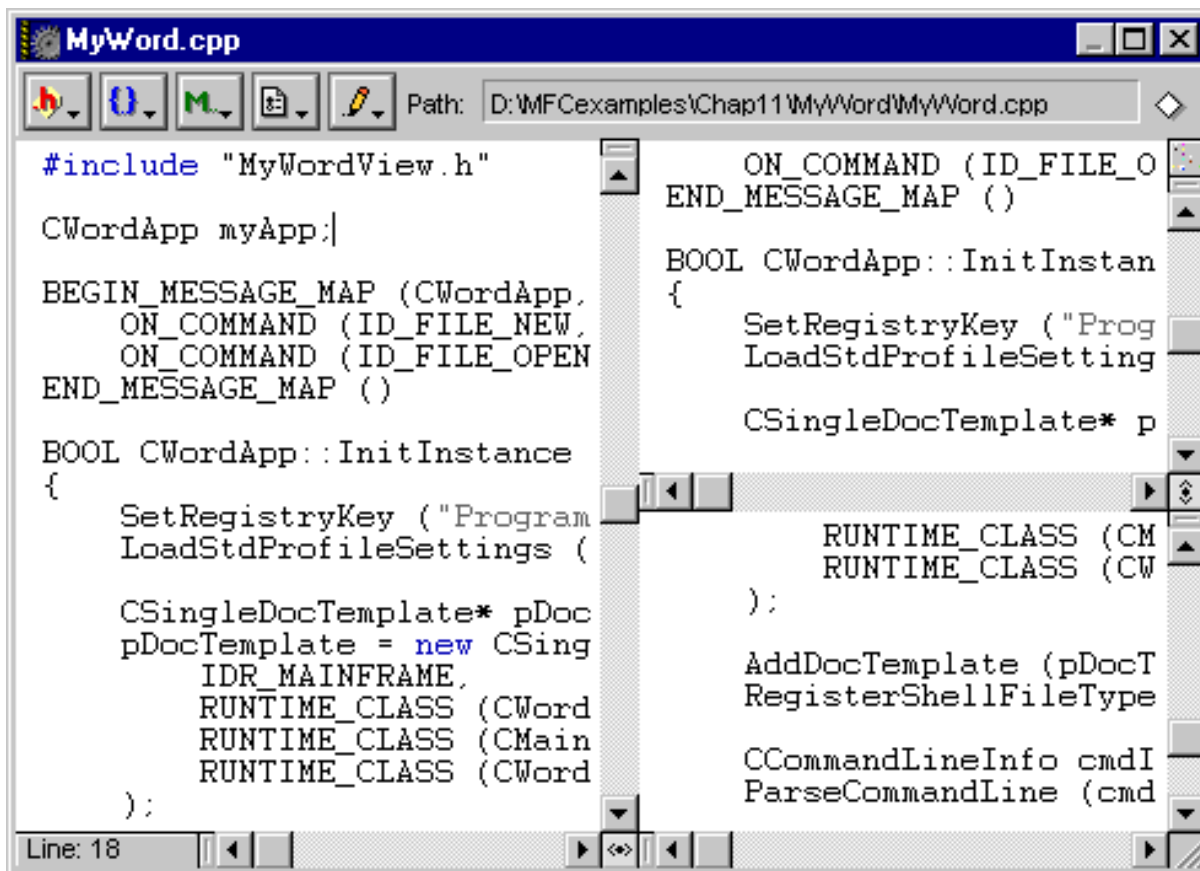
To show the pop-up menus along the top of the editor window again, click the Pop-up Menu Disclosure Button once more.

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 328](#).


## 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 5.9](#). This section describes creating, adjusting, and removing multiple panes.

**Figure 5.9** 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, and 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” on page 225](#).

## Editing Source Code

### *Basic Text Editing*

---

To learn about using this command with the CodeWarrior Browser, see [“Saving a Default Browser” on page 213.](#)

## Customizing the Toolbar

You can customize the icons that are shown on the toolbar in your Editor windows. To learn about how to do this, refer to [“Toolbar Customization” on page 269.](#)

## Basic Text Editing

The CodeWarrior IDE 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

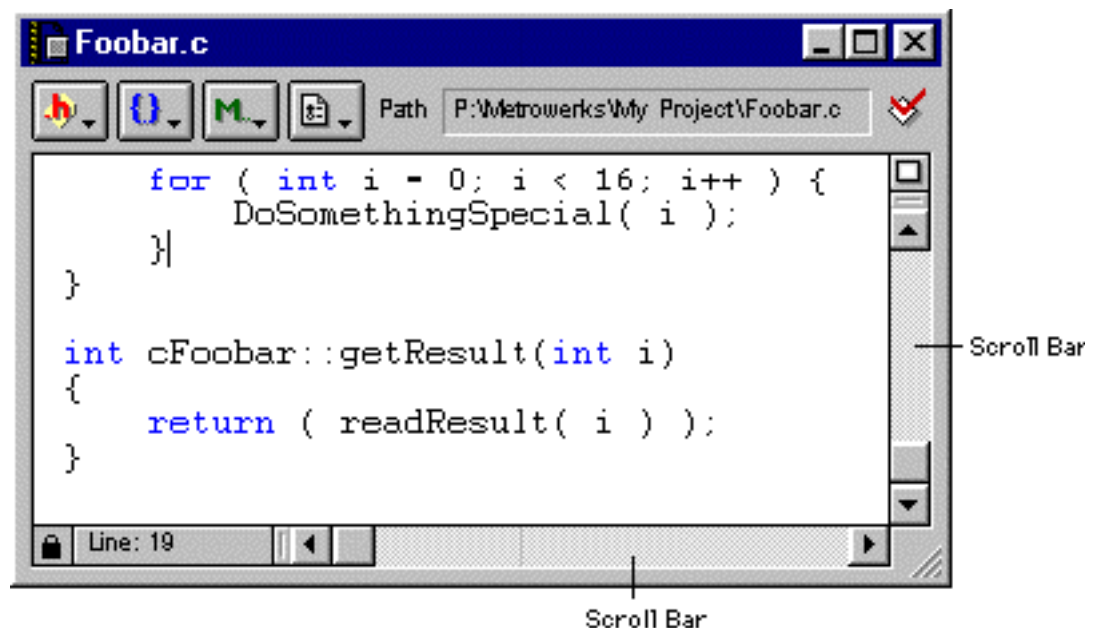
The CodeWarrior IDE gives you several ways to move the text insertion point in a file to get to where you want. Review this section again after you become more familiar with the CodeWarrior editor’s features.

### 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 5.10](#) 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 5.10** 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 222](#).

### Keyboard navigation

[Table 5.1](#) describes how to move the insertion point around in a file with function keys.

**Table 5.1** 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 5.2](#) 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 5.2** 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 132](#).

## 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 135](#).

## 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 5.1](#).

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.

The editor can select parts of text identifiers by holding down the Control key while using the left or right arrow keys, or when double-clicking. For example, double-clicking between the two "m" characters in `FindCommandStatus()` would result in the word `Command` being selected. To list and display an entire routine in the

## Editing Source Code

### *Basic Text Editing*

---

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 136](#).

You can also select blocks of code quickly using the Balance command. To learn how to do this, refer to [“Balancing Punctuation” on page 141](#).

### **Moving Text (Drag and Drop)**

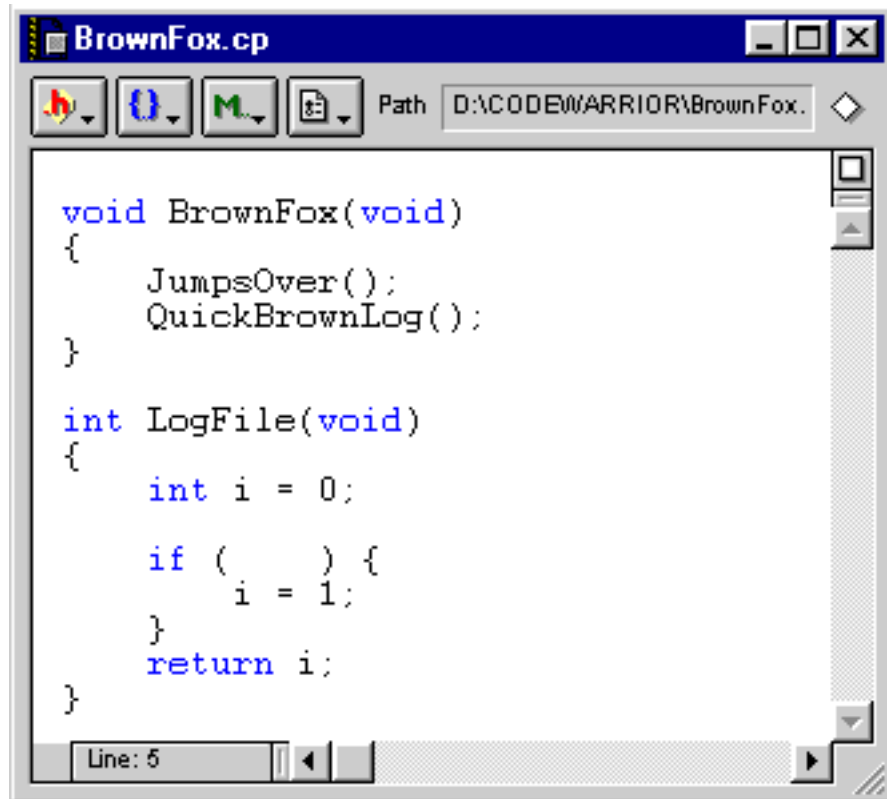
If you have some text in your editor window that you would like to move to a new location, you can use the drag and drop features of the editor to do it. In order to use drag and drop editing, this feature must be enabled in the IDE. To learn more about turning this feature on or off, refer to [“Editor Settings” on page 221](#).

The CodeWarrior editor can also accept drag and drop text items from other applications that support drag and drop. To see if one of your applications supports drag and drop, refer to the documentation that came with it.

[Figure 5.11](#) 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 5.11** Original File Before Drag and Drop



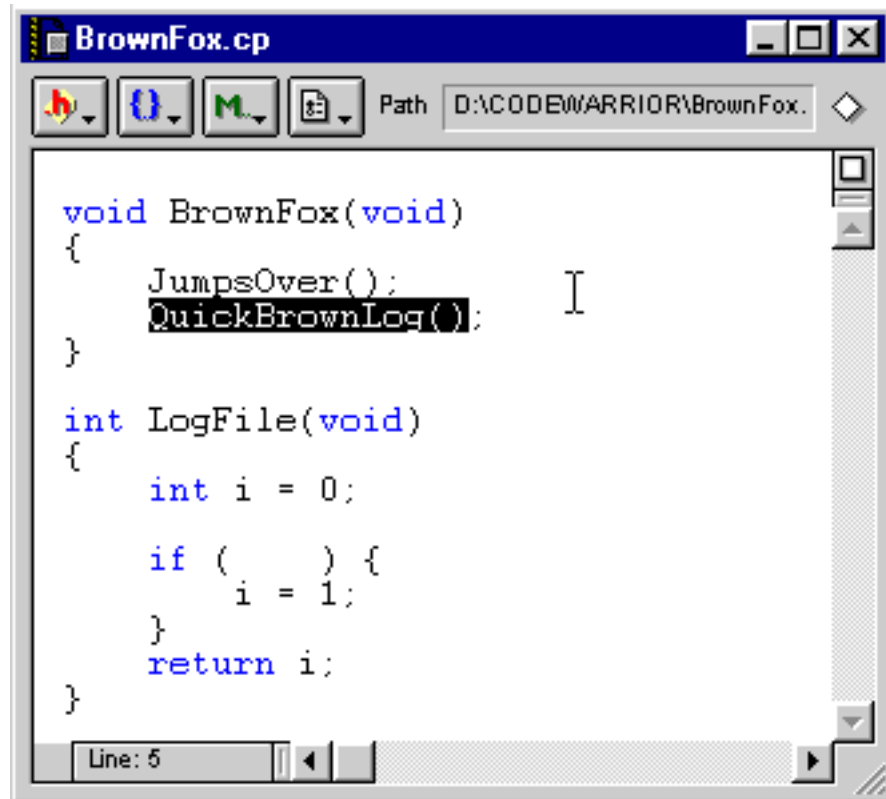
First, select the text you want to move. [Figure 5.12](#) shows a window with the text we're going to move. For information on how to select text, see [“Selecting Text” on page 135](#).

## Editing Source Code

### Basic Text Editing

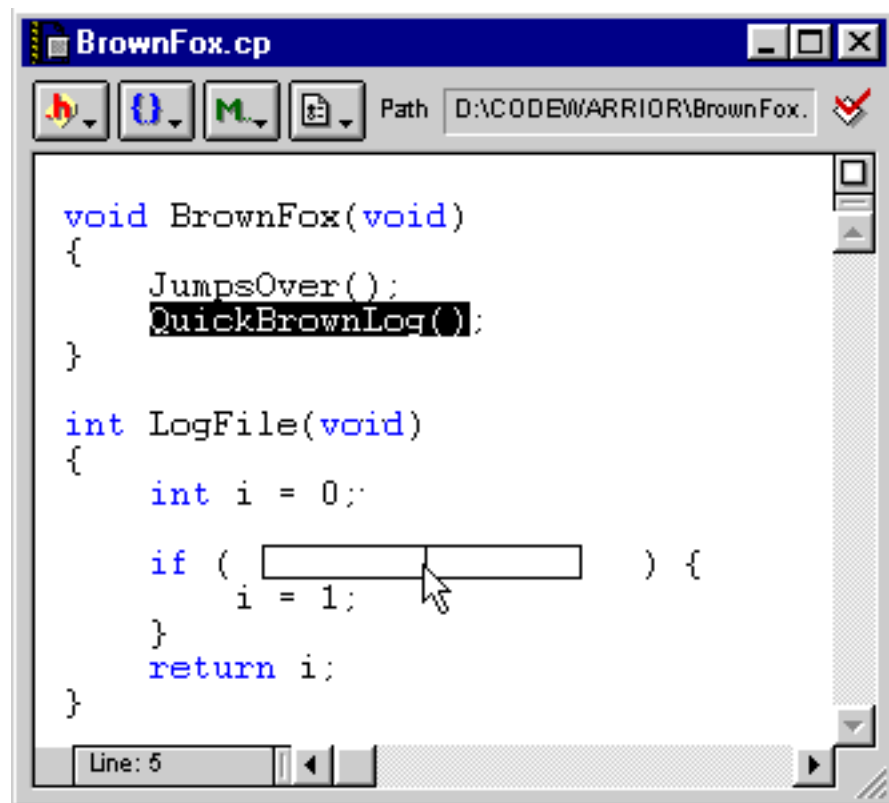
---

**Figure 5.12** 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 5.13](#) shows what the screen looks like as you drag the selected text to a new location.

**Figure 5.13** File with Drag and Drop in Progress



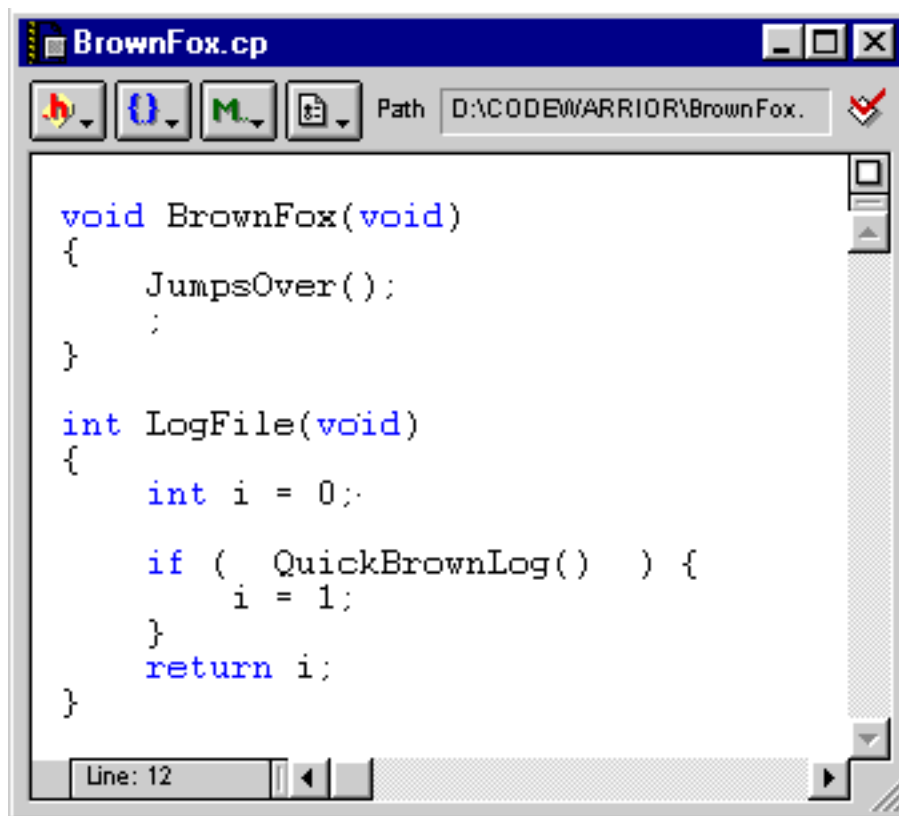
[Figure 5.14](#) shows the final result after dragging the text selection down to a new line.

## Editing Source Code

### Basic Text Editing

---

**Figure 5.14** 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 the CodeWarrior IDE, 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 312](#).

## 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.

The CodeWarrior IDE 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, 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 222](#).

## 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 135](#). 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” on page 225](#).

## 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 Undo, and Multiple 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, Multiple Undo, and Multiple Redo](#) command three times.

To learn how to enable the [Use Multiple Undo](#) option, refer to [“Use Multiple Undo” on page 224](#).

---

**WARNING!** Handle multiple [Undo](#) carefully. If the [Use Multiple Undo](#) option is turned on and you use Undo repeatedly, you may 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 115.](#)

### 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” on page 226.](#)

You can also highlight custom keywords, which are in list of words you designate. See [“Syntax Coloring” on page 226](#) 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.

You should know that you can change the key bindings that cause the text insertion point to move around in a file. To learn how to change the definitions of keys that allow you to move around in a file, refer to [“Configuring Editor Commands Bindings” on page 239.](#)

This section covers these methods:

- [Finding a Routine](#)
- [Adding, Removing, and Selecting a Marker](#)
- [Opening a Related File](#)

## Editing Source Code

### *Navigating in Text*

---

- [Going to a Particular Line](#)
- [Using Go Back and Go Forward](#)
- [Configuring Editor Commands](#)

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 183](#).

## Finding a Routine



Click the Routine icon to display the Routine pop-up menu, discussed in [“The Routine Pop-Up menu” on page 121](#), 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.

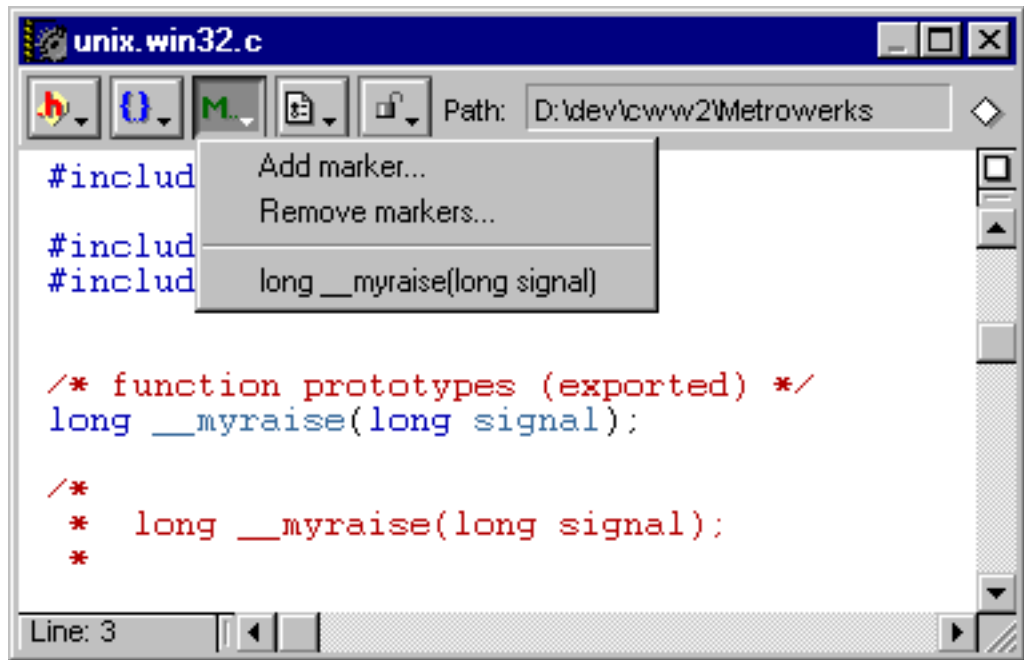
### Adding a Marker

The example text file in [Figure 5.15](#) 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.

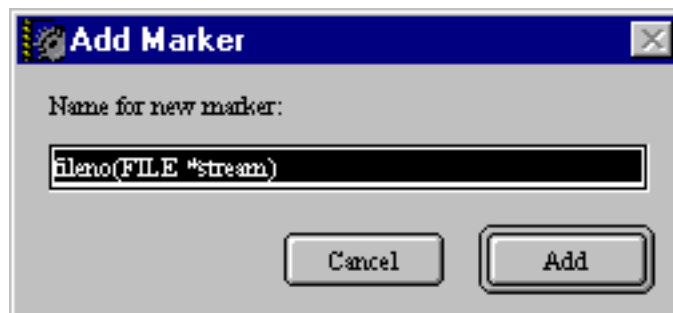


**Figure 5.15** File Marker Menu



After choosing Add marker from the pop-up, you will see the Add Marker dialog box shown in [Figure 5.16](#).

**Figure 5.16** Add Marker dialog box



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

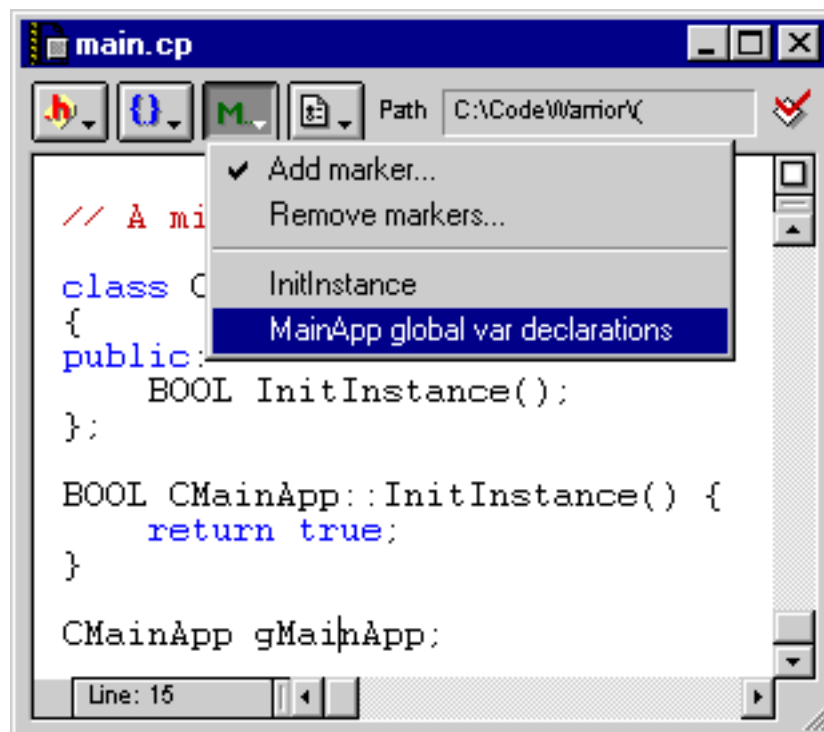
## Editing Source Code

### Navigating in Text

---

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 5.17](#).

**Figure 5.17** 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, can be used to leave markers in a file wherever the following syntax is inserted in the file. For Pascal, use `{$PRAGMA MARK}`. 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.

---

```
#pragma mark myMarker
```

---

---

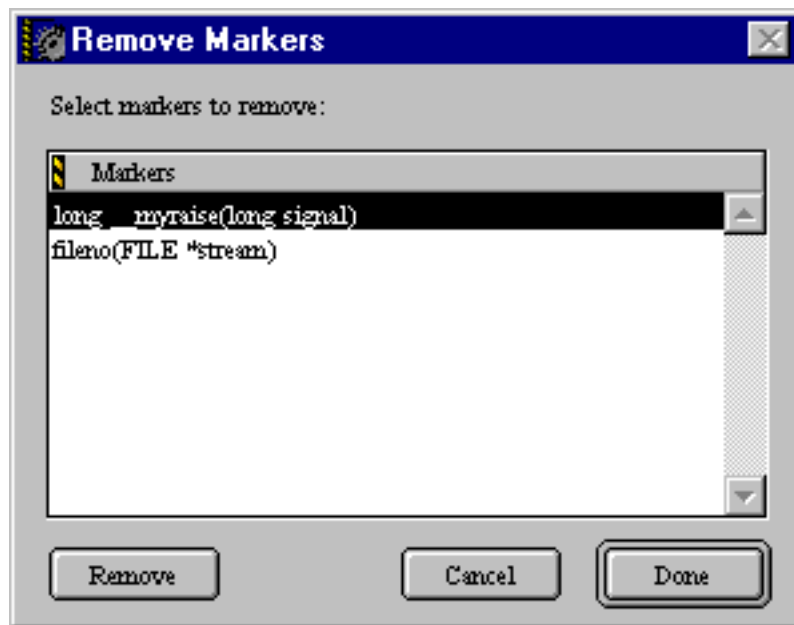
```
{ $PRAGMA MARK myMarker }
```

---

### Removing a Marker

To remove a marker, click the [Marker Pop-Up Menu](#) and choose the Remove markers command. The dialog box shown in [Figure 5.18](#) 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 5.18** Remove Markers dialog box



### 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

## Editing Source Code

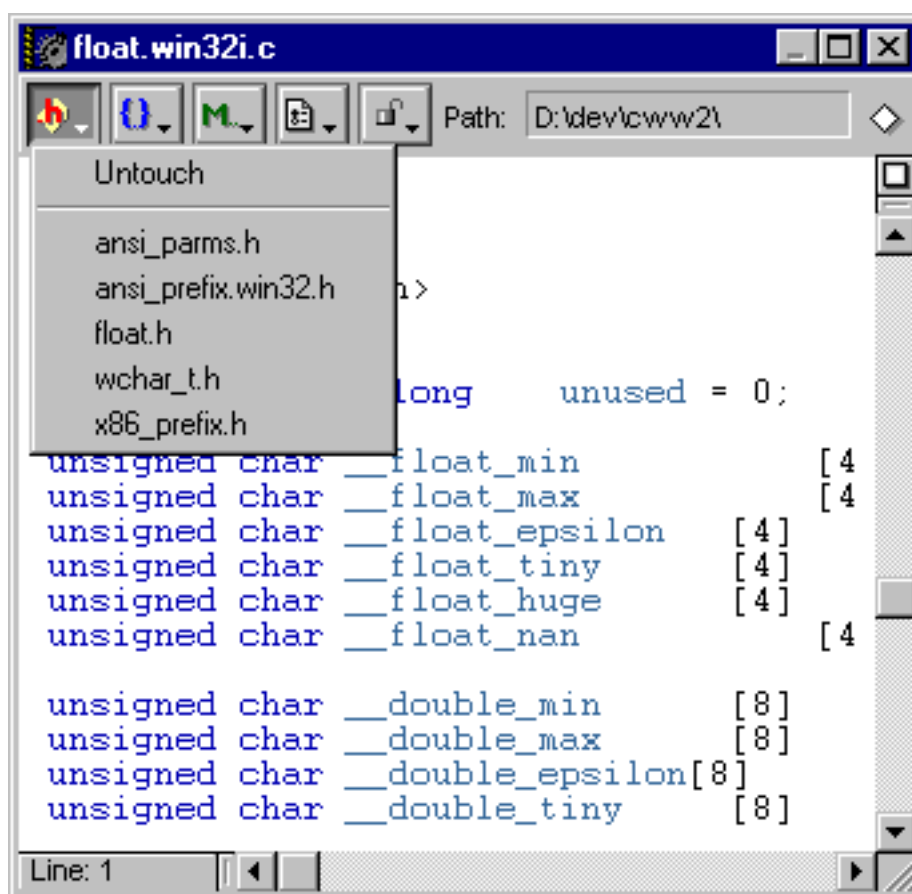
### Navigating in Text

---

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 5.19](#) 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 5.19** The Interface pop-up menu



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

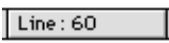
There is another method for opening an interface or header file that your source code file uses. To open the related file, you type a special key combination after selecting the file name in the active win-

dow. To learn more about this method for opening files, refer to [“Opening an Existing File” on page 102](#).

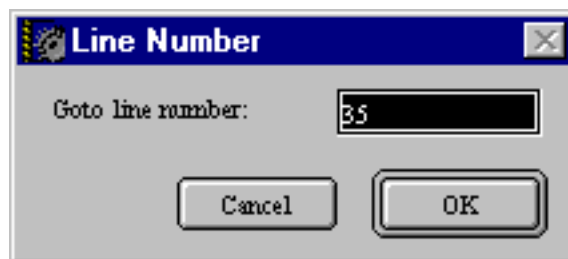
To learn about the Touch and Untouch commands, refer to [“Touching and Untouching Files” on page 274](#).

## 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 box shown in [Figure 5.20](#). Then enter the number of the line you want to go to and select OK.

**Figure 5.20** Go to Line Number dialog box



## 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 208](#) 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 183](#).

### Configuring Editor Commands

The CodeWarrior IDE will allow you to customize key bindings for the editor to suit your working style. Refer to [“Configuring Editor Commands Bindings” on page 239](#) to learn about the commands you can customize to your liking.



# Searching and Replacing Text

---

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

## Searching and Replacing Text Overview

The CodeWarrior IDE provides comprehensive search and replace features with the Find dialog box. 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.

The CodeWarrior IDE also provides facilities for searching text without using the Find dialog box, so that you get quicker searches.

The topics in this chapter are:

- [Guided Tour of the Find Dialog Box](#)
- [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 Dialog Box

The [Find](#) window, shown in [Figure 6.1](#) and [Figure 6.4 on page 159](#), is a versatile feature of the CodeWarrior IDE development environment. To show the Find dialog box, choose the [Find](#) command under the [Search Menu](#). With this window you do text searching through a single file or multiple files in your project. You can search

## Searching and Replacing Text

### *Guided Tour of the Find Dialog Box*

---

and replace text strings and text substrings (using pattern matching), using groups of different files that you specify.

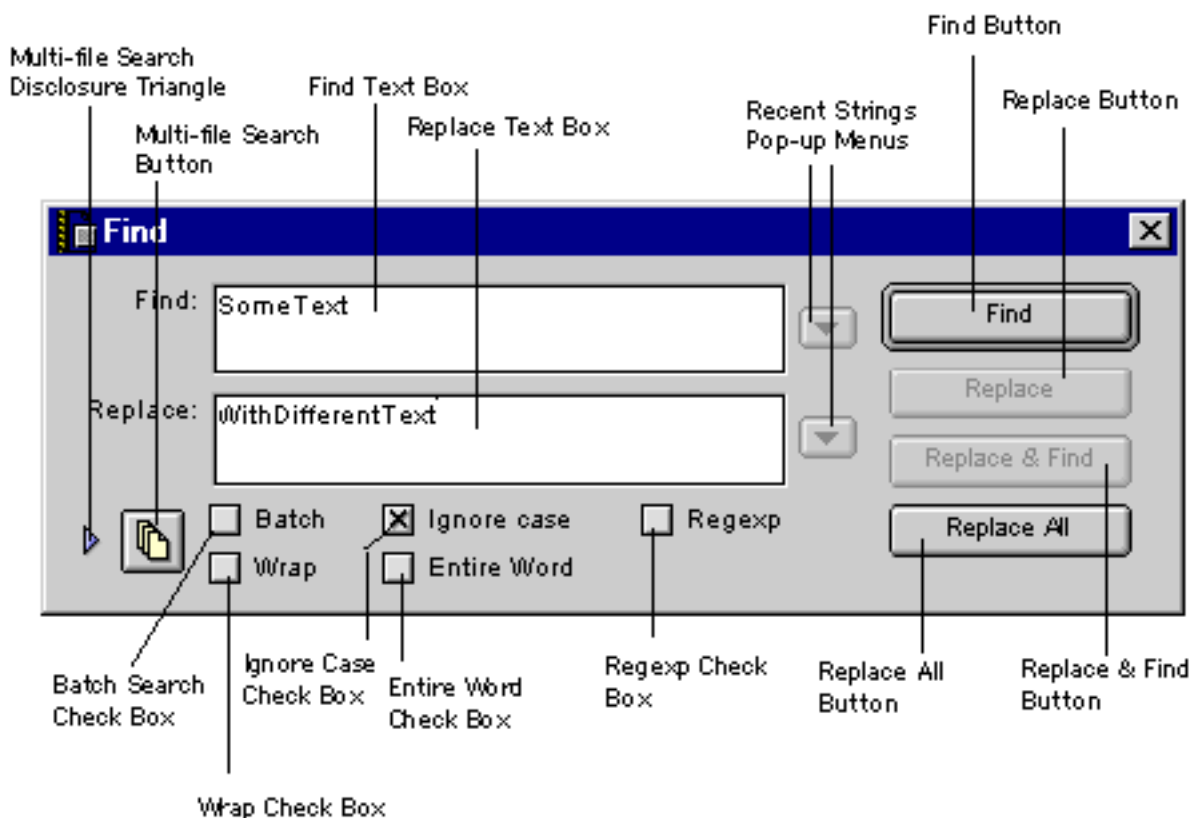
There are two different sections in the Find dialog box.

- [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 Checkbox](#)
- [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 dialog box, shown in [Figure 6.1](#). You enter text in this field that you want to search for.

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 312](#).

Also, the discussion [“Enter ‘Find’ String” on page 317](#) tells how to enter text into the Find Text Box without using the Find dialog box.

### Replace Text Box

The Replace Text Box is one of the editable text fields in the Find dialog box, shown in [Figure 6.1](#). The text you enter in this field will be used to replace the text you’re searching for.

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 312](#).

## Searching and Replacing Text

### *Guided Tour of the Find Dialog Box*

---

Also, the discussion [“Enter ‘Replace’ String” on page 317](#) tells how to enter text into the Replace Text Box without using the Find dialog box.

#### 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 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 dialog box, shown in [Figure 6.1 on page 152](#). It allows you to begin a text search operation once you set the other Find dialog box controls, and have completed certain required fields in the Find dialog box.

To learn more about finding text, see [“Searching for Selected Text” on page 163](#).

### Replace Button

The Replace Button is one of the buttons in the Find dialog box, shown in [Figure 6.1 on page 152](#).

When you enter text in the [Replace Text Box](#) and the [Find Button](#) is clicked, the CodeWarrior IDE 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 168](#).

### Replace & Find Button

The Replace & Find Button is shown in [Figure 6.1 on page 152](#). 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 164](#) and [“Searching and Replacing Text in Multiple Files” on page 171](#).

### Replace All Button

The Replace All Button is shown in [Figure 6.1 on page 152](#). 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 168](#).

### Batch Checkbox

The Batch Check Box is shown in [Figure 6.1 on page 152](#). Selecting this check box causes the results of the Find command to print into the Message window.

## Searching and Replacing Text

### *Guided Tour of the Find Dialog Box*

---

If you'd like to learn more about the role of the Batch Check Box in searching, see [“Using Batch Searches” on page 169.](#)

#### **Wrap Check Box**

The Wrap Check Box is shown in [Figure 6.1 on page 152](#). 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 166.](#)

#### **Ignore Case Check Box**

The Ignore Case Check Box is shown in [Figure 6.1 on page 152](#). This check box causes the CodeWarrior IDE to disregard the 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 167.](#)

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

The Entire Word Check Box is shown in [Figure 6.1 on page 152](#). This check box causes the CodeWarrior IDE 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 167.](#)

#### **Regexp Check Box**

The Regexp Check Box is shown in [Figure 6.1 on page 152](#). This check box causes the CodeWarrior IDE 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 178.](#)

### Multi-File Search Disclosure Triangle

The Multi-File Search Disclosure Triangle is shown in [Figure 6.1 on page 152](#). 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 159](#), or [Figure 6.3 on page 158](#)

To learn more about multi-file searching using the [Find](#) window, see [“Searching and Replacing Text in Multiple Files” on page 171](#).

### 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 154](#), 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 the dialog box of [Figure 6.3](#), the items in the [Multi-File Search Section](#) of the [Find](#) window are dimmed.

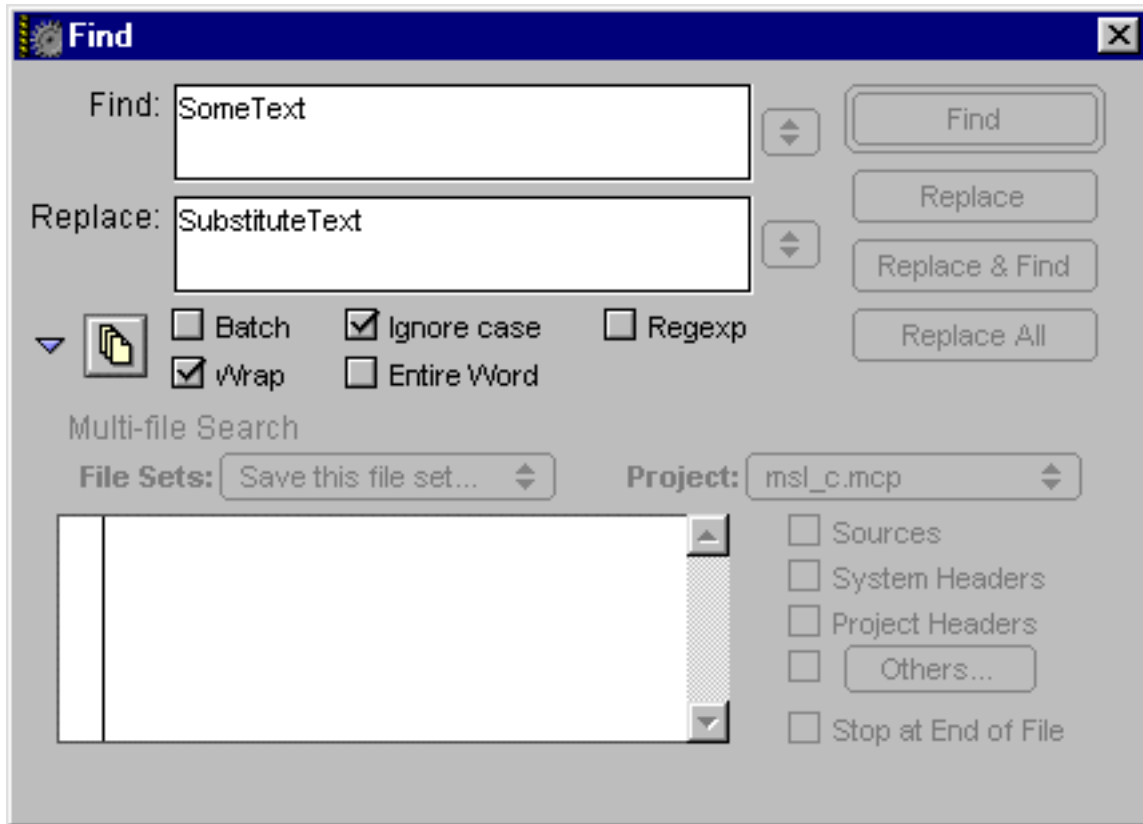
To learn more about Multi-File Search Button, see [“Activating Multi-File Search” on page 171](#).

## Searching and Replacing Text

### *Guided Tour of the Find Dialog Box*

---

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

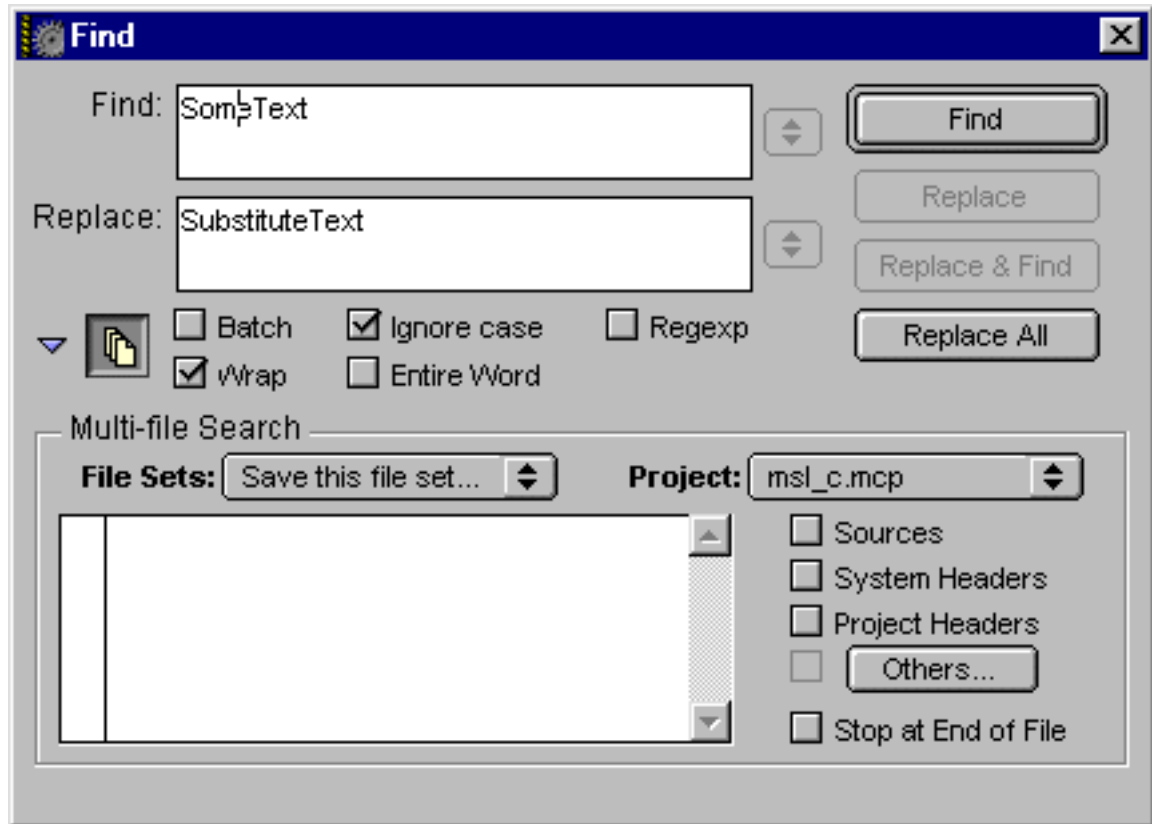


## Multi-File Search Section

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](#)
- [Project Pop-up Menu](#)
- [Stop at End of File Check Box](#)
- [Sources Check Box](#)
- [System Headers Check Box](#)
- [Project Headers Check Box](#)
- [Others Button](#)

**Figure 6.4** The Find Dialog Box 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.

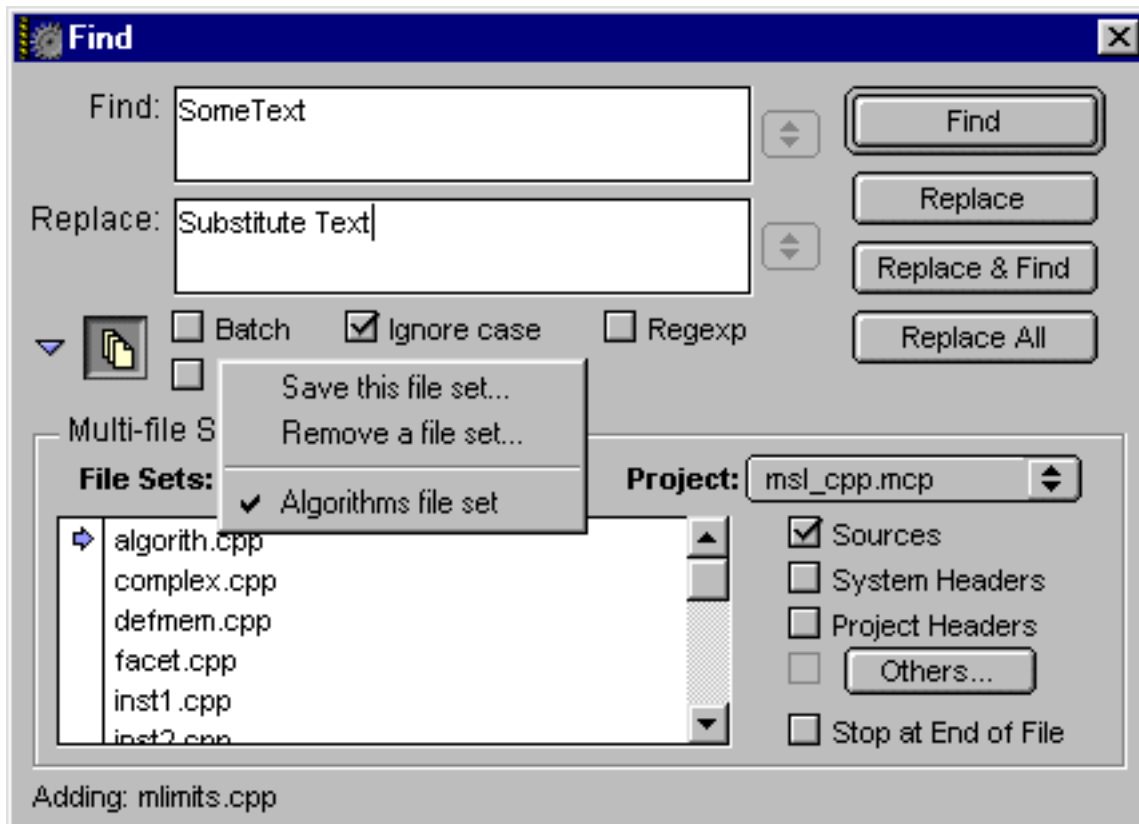
You can build up sets of files that you care about, such as collections of header or interfaces files, that will be available whenever you want to search through the files for text.

For more information about Multi-file sets of files, see [“Choosing Files to be Searched”](#) on page 172.

## Searching and Replacing Text

### *Guided Tour of the Find Dialog Box*

**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.

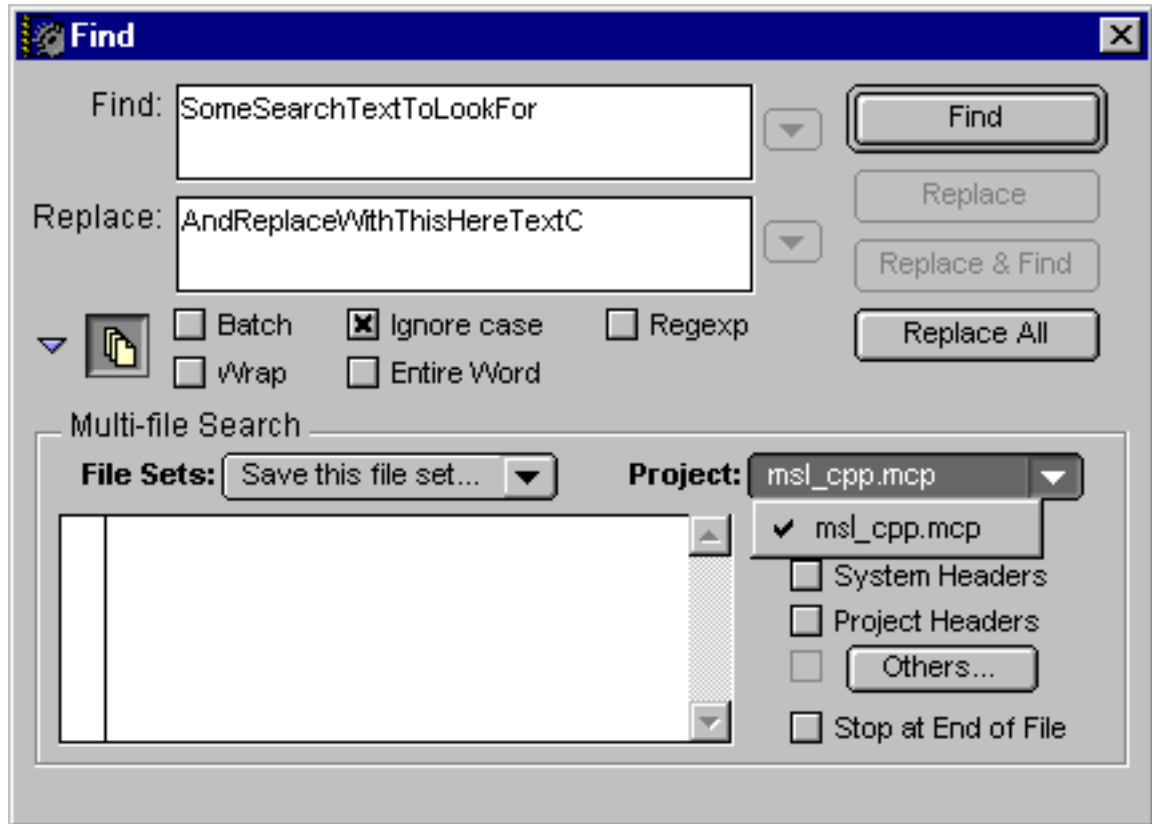
For more information about adding files and removing files in file sets, see [“Choosing Files to be Searched” on page 172](#).

### Project Pop-up Menu

The Project Pop-up Menu, shown in [Figure 6.6](#), allows you to choose the project file which you want to perform your search. Since the CodeWarrior IDE can have multiple projects open at a time, this menu provides a way to perform the same search in different projects.



**Figure 6.6 Project Pop-up Menu**



### **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 CodeWarrior IDE 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 CodeWarrior IDE reaches the end of a file, it 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 166](#).

## Searching and Replacing Text

### *Guided Tour of the Find Dialog Box*

---

#### **Sources Check Box**

The Sources Check Box is shown in [Figure 6.4 on page 159](#). 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 172](#).

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

The System Headers Check Box is shown in [Figure 6.4 on page 159](#). This check box adds all the system header or interfaces 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 173](#).

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

The Project Headers Check Box is shown in [Figure 6.4 on page 159](#). This check box adds all the header or interfaces 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 173](#).

#### **Others Button**

The Others Button is shown in [Figure 6.4 on page 159](#). 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 174](#).

## Searching for Selected Text

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

When you search for text using this method, the CodeWarrior IDE uses the option settings that you last chose in the [Find](#) window. To change these option settings, you must use the Find dialog box.

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 135](#).

### 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](#).

The CodeWarrior IDE 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](#).

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

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

## Searching and Replacing Text

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

---

#### 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 dialog box.

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.

The CodeWarrior IDE 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 dialog box 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 102](#).

If you haven't yet created a file, see [“Creating a New File” on page 101](#).

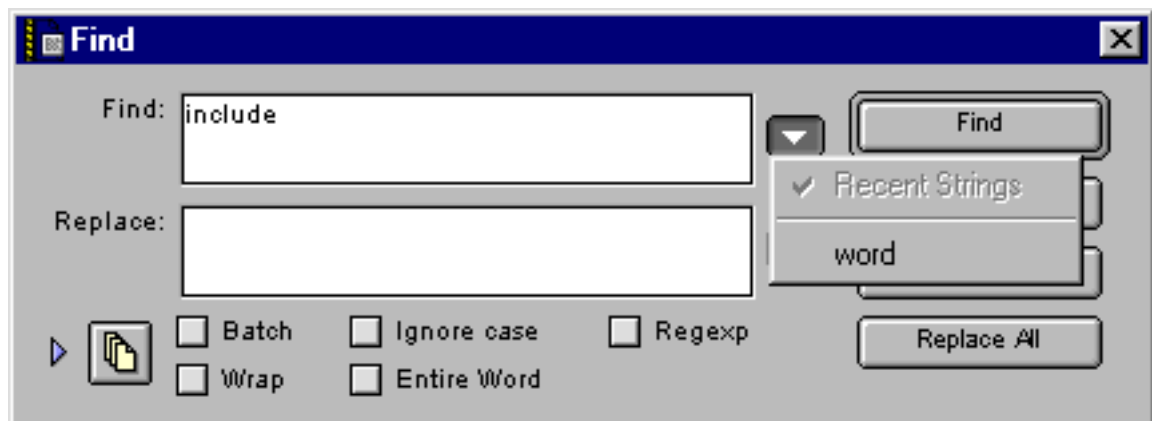
The topics in this section are:

- [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 box, or choose a string from the [Recent Strings Pop-Up Menu](#), as shown in [Figure 6.7](#).

**Figure 6.7 Find Text Box and Recent Strings Pop-up Menu**



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

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 166](#).

## Searching and Replacing Text

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

---

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 167.](#)

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 171.](#)

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 168.](#)

## Controlling Search Range

The [Wrap Check Box](#) option in the Find dialog box 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](#), the CodeWarrior IDE 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, the CodeWarrior IDE 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 152](#). This check box causes the CodeWarrior IDE 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 CodeWarrior IDE will also find occurrences like “foobar” or “FOOBAR”, and other possible combinations of upper and lower-case text characters.

### Entire Word Check Box

The Entire Word Check Box is shown in [Figure 6.1 on page 152](#). This check box causes the CodeWarrior IDE 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”, the CodeWarrior IDE finds only “Word”. If this option is off, it matches text like “Words”, “Word-Count”, and “BigWordCount”.

## Searching and Replacing Text

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

---

## 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 dialog box. You can read more about how to find text by referring to [“Finding Search Text” on page 165](#).

Next, enter the replacement text string in the [Replace Text Box](#) field of the Find dialog box.

To replace all the occurrences of the [Find Text Box](#) string, click the [Replace All Button](#) in the Find dialog box, 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

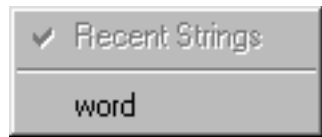
To selectively replace text, first enter some text to find, then choose the [Find](#) operation on the [Search Menu](#), or click the [Find Button](#) in the Find dialog box. You can read more about how to find text by referring to [“Finding Search Text” on page 165](#).

Next, enter the replacement text string in the [Replace Text Box](#) field of the Find dialog box.

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 6.8](#)) contains the last five strings you have used.



**Figure 6.8** Recent Strings pop-up menu



Now choose whether to replace the string you found. For convenience, there are three buttons in the Find dialog box 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 dialog box 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 dialog box.

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 dialog box.

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 dialog box. 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.

## Using Batch Searches

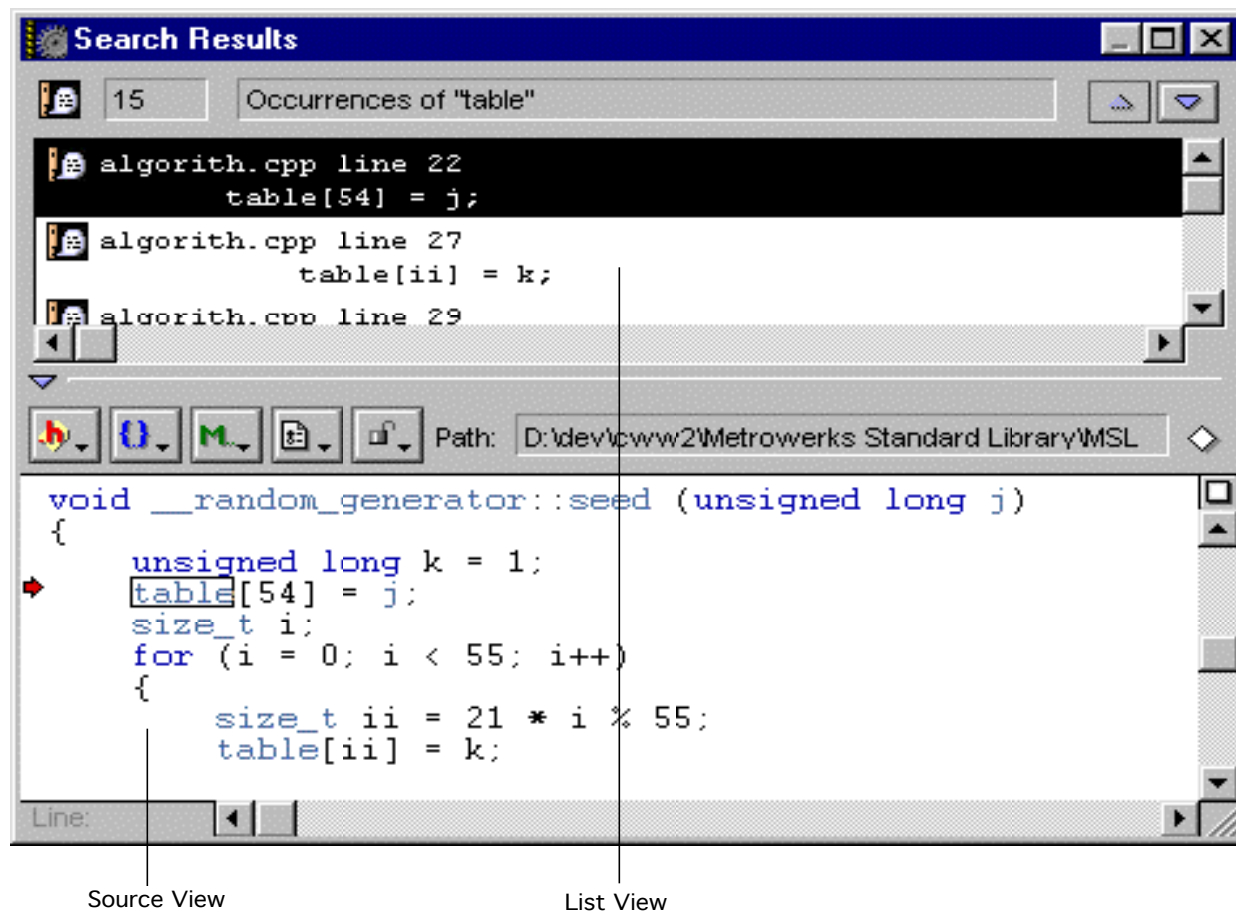
The CodeWarrior IDE gives you a way to collect all matching descriptions of your text search in one window for easy reference.

If the [Batch Checkbox](#) option is checked in the Find dialog box, and the Find button is clicked, the CodeWarrior IDE searches for all occurrences of the [Find Text Box](#) string and lists them in the Search Results message window, as shown in [Figure 6.9](#).

## Searching and Replacing Text

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

**Figure 6.9** Batch search results



The Search Results window shown in [Figure 6.9](#) 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 290](#).

## Searching and Replacing Text in Multiple Files

The CodeWarrior IDE 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 208](#).

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 the CodeWarrior IDE to search through multiple files, you need to activate multi-file searching in the Find dialog box.



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.

Click the [Multi-File Search Disclosure Triangle](#) to the left of the [Multi-File Search Button](#), shown in [Figure 6.10 on page 172](#), so that the triangle points down.

The CodeWarrior IDE displays the Find dialog box with the [Multi-File Search Section](#) enabled, as shown in [Figure 6.10](#).

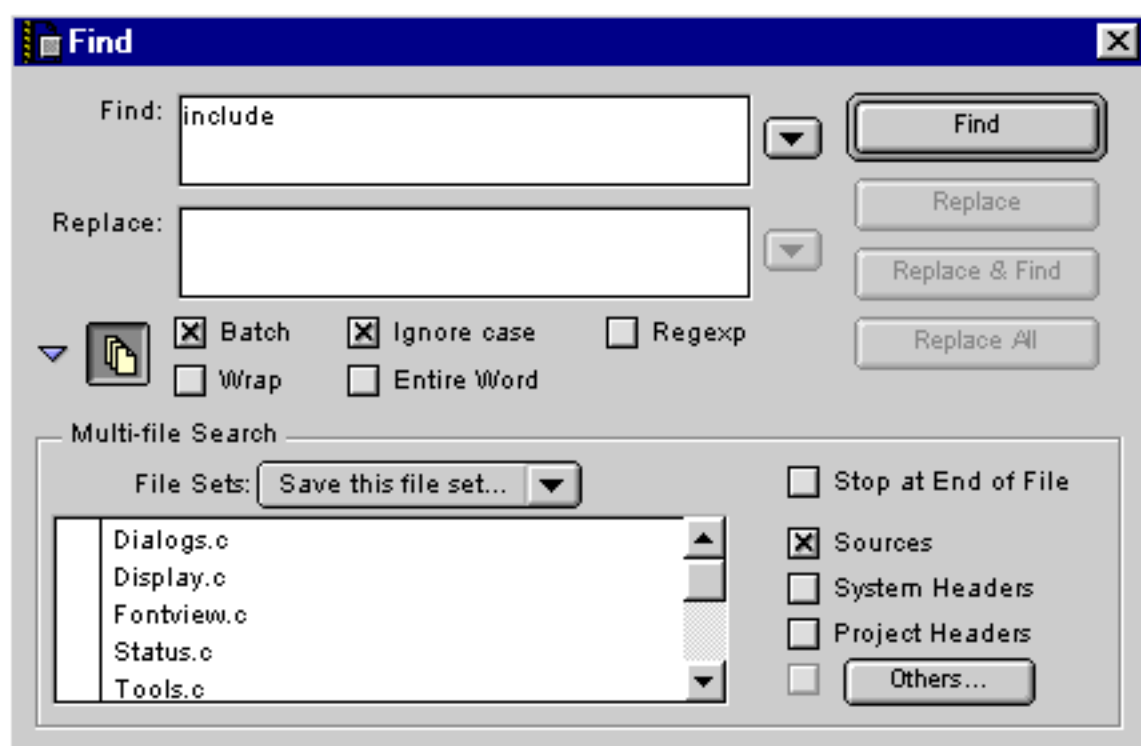
## Searching and Replacing Text

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

---

To learn about how to configure the [Multi-File Search Section](#) of the Find dialog box, refer to [“Choosing Files to be Searched” on page 172](#), [“Saving a File Set” on page 176](#), [“Removing a File Set” on page 177](#), and [“Controlling Search Range” on page 177](#).

**Figure 6.10** The Find dialog box 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](#), the CodeWarrior IDE removes all associated files from the file list.

To include only some of the files, turn on the [Sources Check Box](#) and delete the files you don't want by selecting them and pressing Delete after clicking on the file name.

If turning on this option doesn't add any files, update your project's internal list of header and interfaces files with the Make command. To learn how to do this, refer to ["Making a Project" on page 276](#).

### **Adding project header files**

To add all the project header or interfaces files from the current project, turn on the [Project Headers Check Box](#). When you turn off the [Project Headers Check Box](#), the CodeWarrior IDE removes all associated files from the file list.

To include only some of the files, turn on the [Project Headers Check Box](#) and delete the files you don't want by selecting them and pressing Delete.

If turning on this option doesn't add any files, update your project's internal list of header or interfaces files with the Make command. To learn how to do this, refer to ["Making a Project" on page 276](#).

### **Adding system header files**

To add all the system header or interfaces files from the current project, turn on the [System Headers Check Box](#). When you turn off the [System Headers Check Box](#), the CodeWarrior IDE removes all associated files from the file list.

To include only some of the files, turn on the [System Headers Check Box](#) and delete the files you don't want by selecting them and pressing Delete.

If turning on this option doesn't add any files, update your project's internal list of header or interfaces files with the Make command. To learn how to do this, refer to ["Making a Project" on page 276](#).

## Searching and Replacing Text

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

---

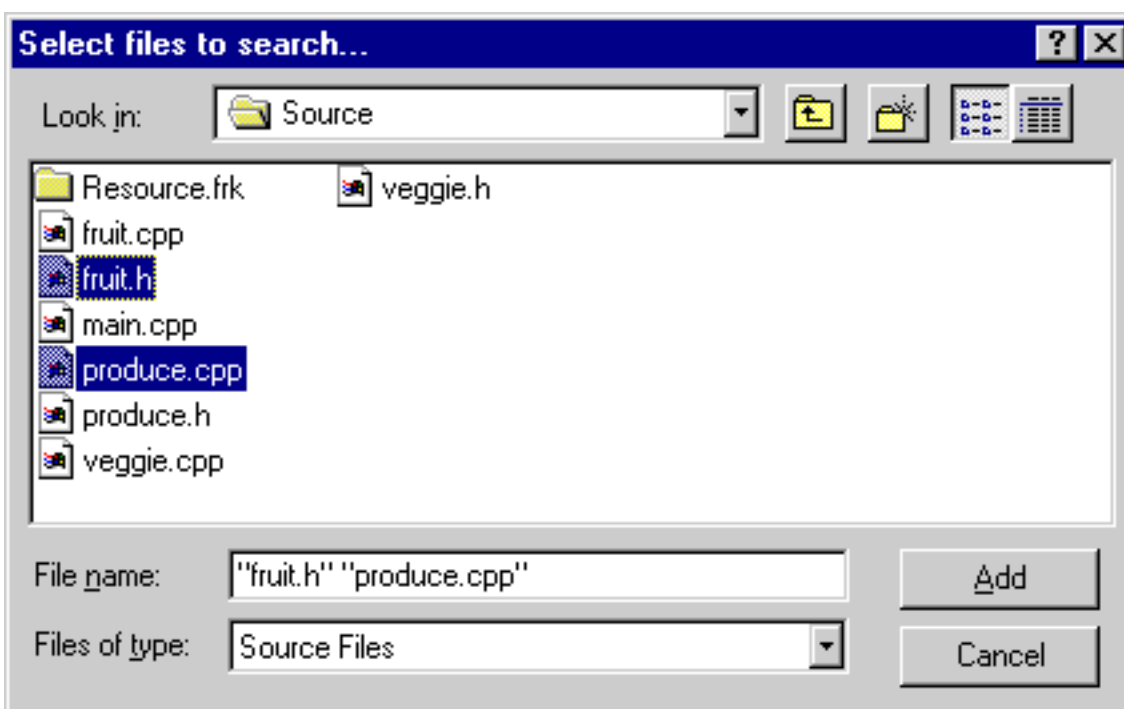
#### Adding and removing arbitrary files

For your multi-file searches, you can add and remove files using the Add File dialog box shown in [Figure 6.11](#). 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 dialog box. Then, choose any files from the dialog's File List.

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 6.11** Adding files to a file set with the Add Dialog Box



The Select files to search dialog box shows the files in the current directory you may choose to add to the file set.

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

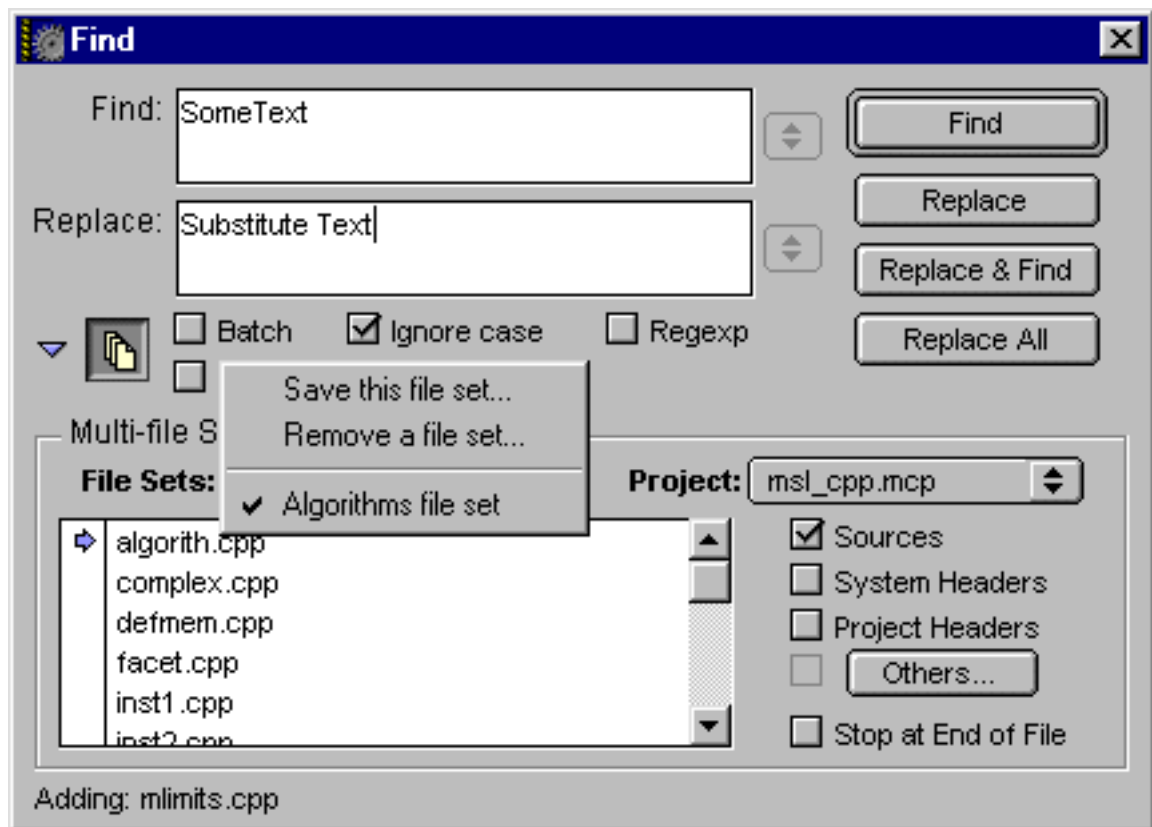
When you're finished selecting files, click the Add button. If you change your mind, click Cancel and the file set is unchanged.

To add more files later, just click the [Others Button](#) in the Find dialog box and the same dialog box 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 6.12](#).

**Figure 6.12** File Sets Pop-up menu



## Searching and Replacing Text

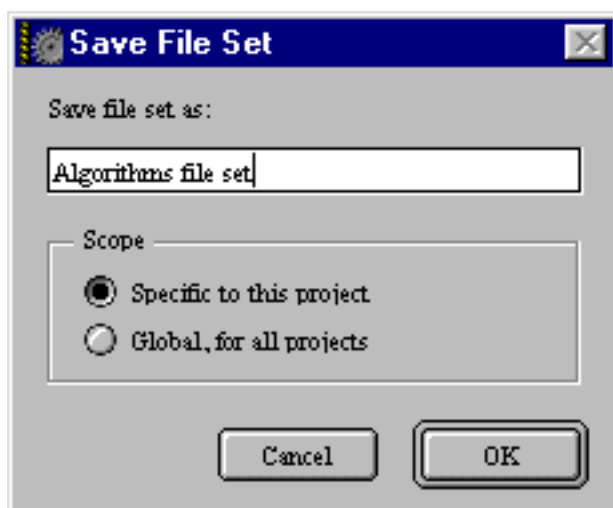
### *Searching and Replacing Text in Multiple Files*

---

## 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. The CodeWarrior IDE displays the Save File Set dialog box shown in [Figure 6.13](#).

**Figure 6.13** The Save File Set dialog box



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 box.

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

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

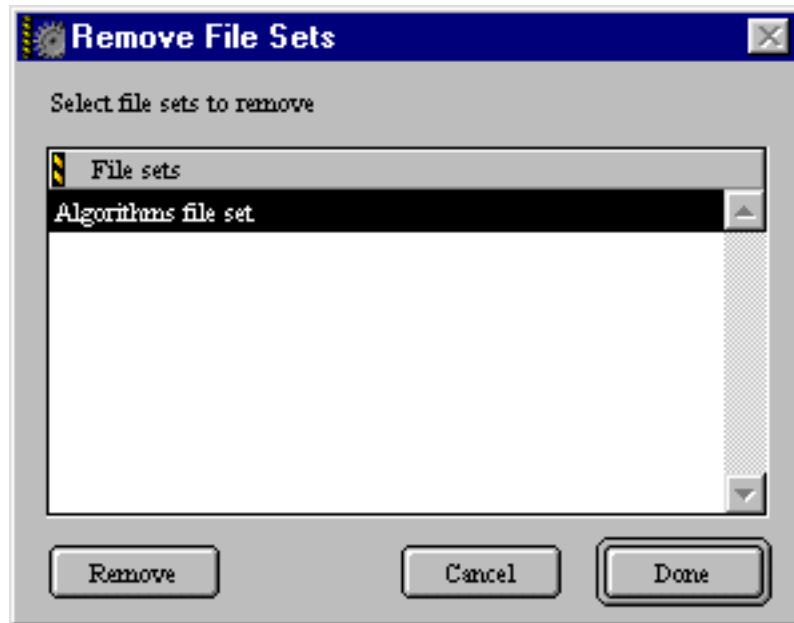
After making your selection and naming the file set, click the Save button. If you change your mind and don't want to save the file set, click Cancel.



## Removing a File Set

To remove a previously-saved file set, choose Remove a file set from the [File Sets Pop-Up Menu](#) in the Find window. Select the set you want to delete in the dialog box that appears, shown in [Figure 6.14](#), and click the Remove button. CodeWarrior removes the file set so that future searches will not have this file set available. If you change your mind about removing the file set, click Cancel instead.

**Figure 6.14** Remove a file set dialog box



## Controlling Search Range

The CodeWarrior IDE 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, just save that set and restore it later.

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

## Searching and Replacing Text

*Using Regular Expressions (grep)*

---

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](#) 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.

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 163](#), or [“Searching and Replacing Text in Multiple Files” on page 171](#).

## 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 CodeWarrior IDE, the CodeWarrior IDE analyzes whether the text matches the regular expression you have entered.

This section discusses regular expressions the CodeWarrior IDE 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,

<b>This expression...</b>	<b>matches this...</b>	<b>but not this...</b>
a	a	b
\. \*	. *	dog
100	100	ABCDEFGF

### Matching any character

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

<b>This expression...</b>	<b>matches this...</b>	<b>but not this...</b>
.art	dart cart tart	art hurt dark

### Repeating expressions

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:

## Searching and Replacing Text

Using Regular Expressions (*grep*)

---

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

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, sup-

## Searching and Replacing Text

*Using Regular Expressions (grep)*

---

pose 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...
<code>(ab)\1</code>	abab	abc
<code>(ab. )\1</code>	abcabc ablabl	abcabl abab

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`.



# Browsing Source Code

---

This chapter describes Code Warrior's class browser, a tool you use to examine your project source code from various perspectives.

## Browser Overview

This chapter gives you a full description of the CodeWarrior IDE browser. The browser lets you decide what code is important to look at, and lets you get to that code quickly and easily.

The CodeWarrior IDE browser creates a database of all the symbols in your code, and provides you with a user interface to access the data quickly and easily, regardless of language.

Historically, programmers have used browsers primarily with object-oriented code, but the CodeWarrior IDE browser works with procedural and object-oriented code. It works with most compilers, including C, C++, Pascal, and Java.

To help you understand the browser, we're going to look at it from three perspectives: high-level architecture, user interface, and functionality. The topics in this chapter include:

- [Understanding the Browser Strategy](#)—what the browser is and what it does from a high-level perspective
- [Guided Tour of the Browser](#)—what you see when you work with the browser
- [Using the Browser](#)—how to use the browser effectively
- [Customizing the Browser](#)—how to customize the toolbar

The rest of this section shows you how to activate the browser so that you can start using it.

## Activating the Browser

To learn more about how to activate the browser, refer to [“Activate Browser” on page 252](#). When the browser is activated, the compiler generates the browser database information.

For more information on browser settings and options, see [“Setting Browser Options” on page 207](#).

## Understanding the Browser Strategy

When the browser is activated, the CodeWarrior IDE compilers generate a database of information about your code. This database includes data not only about your code, but about the relationships between various parts of your code, such as inheritance hierarchies.

The browser is a user interface that allows you to sort and sift through this information in ways that suit your needs.

Like any good database access program, the browser does not dictate how you should look at your information. It gives you a variety of tools to suit your working style.

There are three principal ways of looking at the information available to you in the browser:

- [Catalog View](#)—a comprehensive view of all data
- [Browser View](#)—a class-based view
- [Hierarchy View](#)—an inheritance-based view

These sections take a brief look at each option. [“Guided Tour of the Browser” on page 188](#) discusses the user interface in detail.

The browser also implements instant access to information. By clicking and holding the mouse on any symbol for which there is information in the database, you get instant access to related source code. [“Context Pop-Up Menu” on page 205](#) discusses this feature.



In addition, the browser gives you one more approach to deciding how you should view data. You decide how broad your view is. You may want to look at data in all your classes. Or, you may wish to focus on one class.

Within the browser and hierarchy views, you can look at multiple classes or single classes. [Table 7.1](#) summarizes the various major choices you have when using the browser

**Table 7.1    Browser viewing options**

viewing style	wide focus	narrow focus
comprehensive	catalog	not applicable
class-based	multi-class browser	single-class browser
inheritance-based	multi-class hierarchy	single-class hierarchy

The browser-related menu commands in the [Window Menu](#)—[Show Catalog Window](#), [Show Hierarchy Window](#), and [New Class Browser](#)—display wide-focus views. Once you have the wide view, you can focus on a particular class.

No matter what viewing style or focus you happen to be in at any given moment, the browser has simple and intuitive mechanisms for switching to another kind of view.

## Catalog View

The catalog view lets you see all of your data sorted by category into alphabetical lists. [Figure 7.1](#) shows a catalog view, with the various categories shown in the pop-up menu.

You select the particular category you want to examine. You can focus on classes, constants, enumerations, routines, global variables, macros, routine templates, and type definitions.

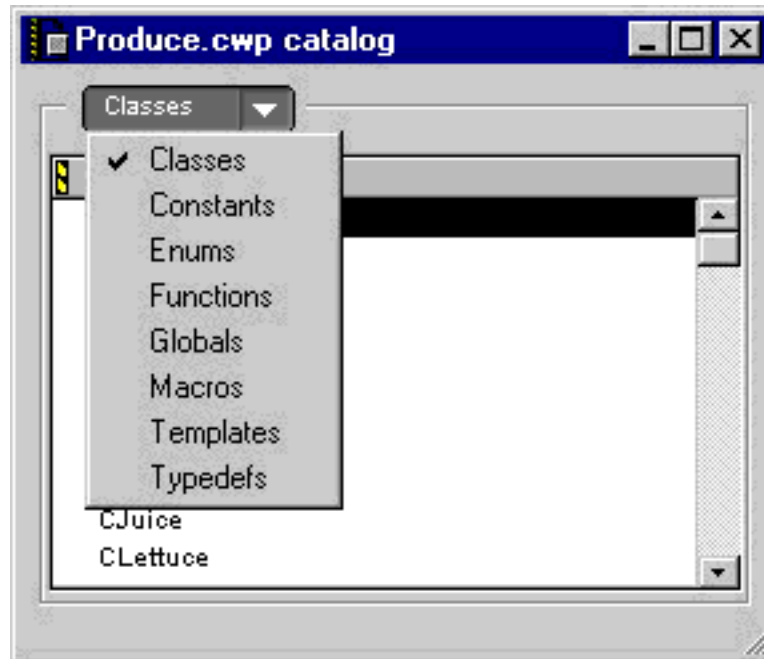
See [“Catalog Window” on page 189](#) for details on the catalog window interface.

## Browsing Source Code

*Understanding the Browser Strategy*

---

**Figure 7.1** A catalog view



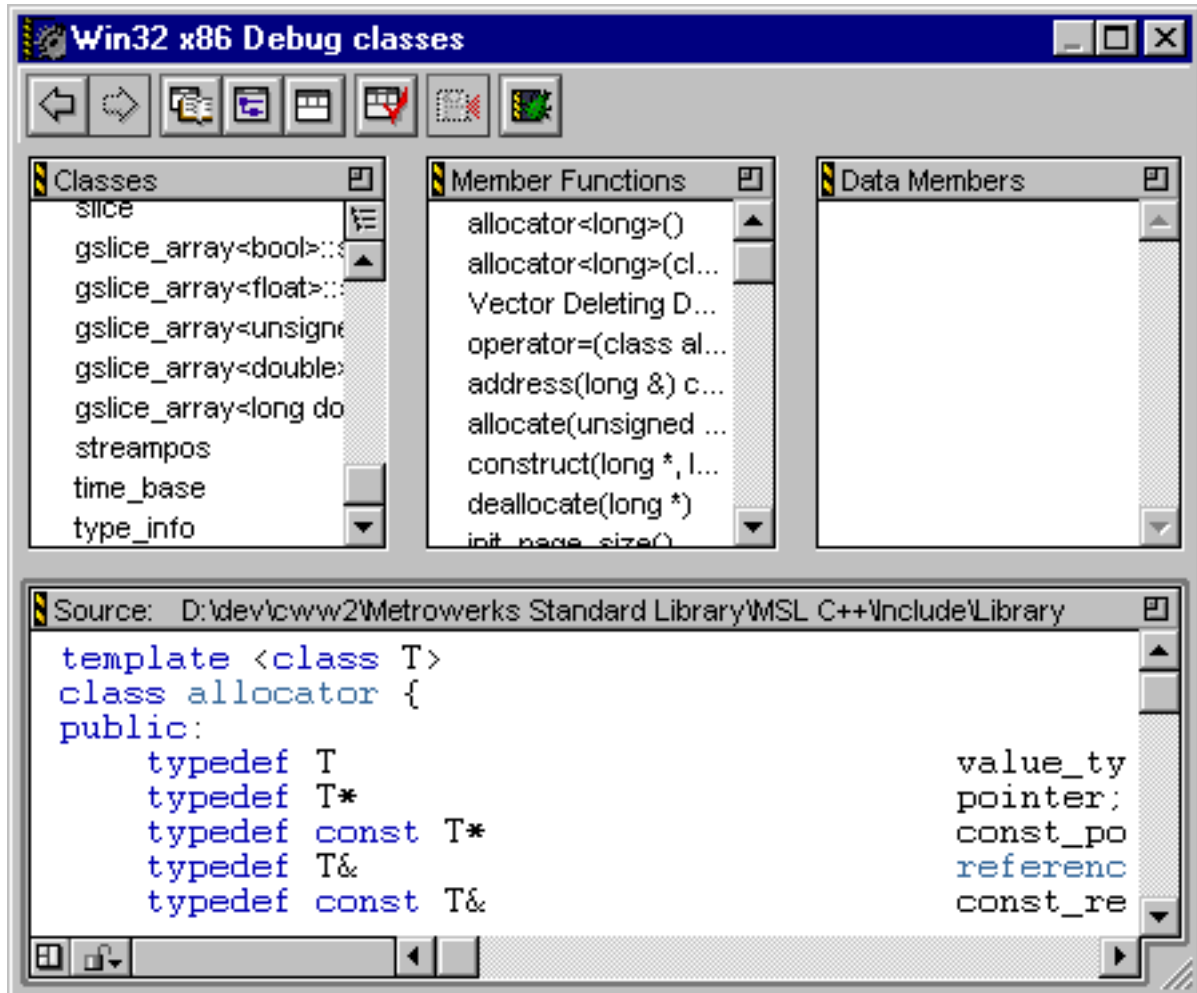
## Browser View

The browser view is like a traditional class browser. You use this view to look at your data from a class-oriented perspective. [Figure 7.2](#) shows what a multi-class browser view looks like.

In the browser view, you have a list of classes. For the selected class in that list, you see all of its member routines and its data members. When you select an item, the source code related to that item appears in the Source code pane.

See [“Multi-Class Browser Window” on page 191](#) and [“Single-Class Browser Window” on page 197](#) for details on the browser view interface.

**Figure 7.2** A browser view



## Hierarchy View

The hierarchy view is a graphical view of your class hierarchy. You use this view to understand or follow class relationships. [Figure 7.3](#) illustrates a multi-class hierarchy view for some classes.

The hierarchy view gives you a real feel for the way your classes are connected with each other. You can expand and collapse a hierarchy at will.

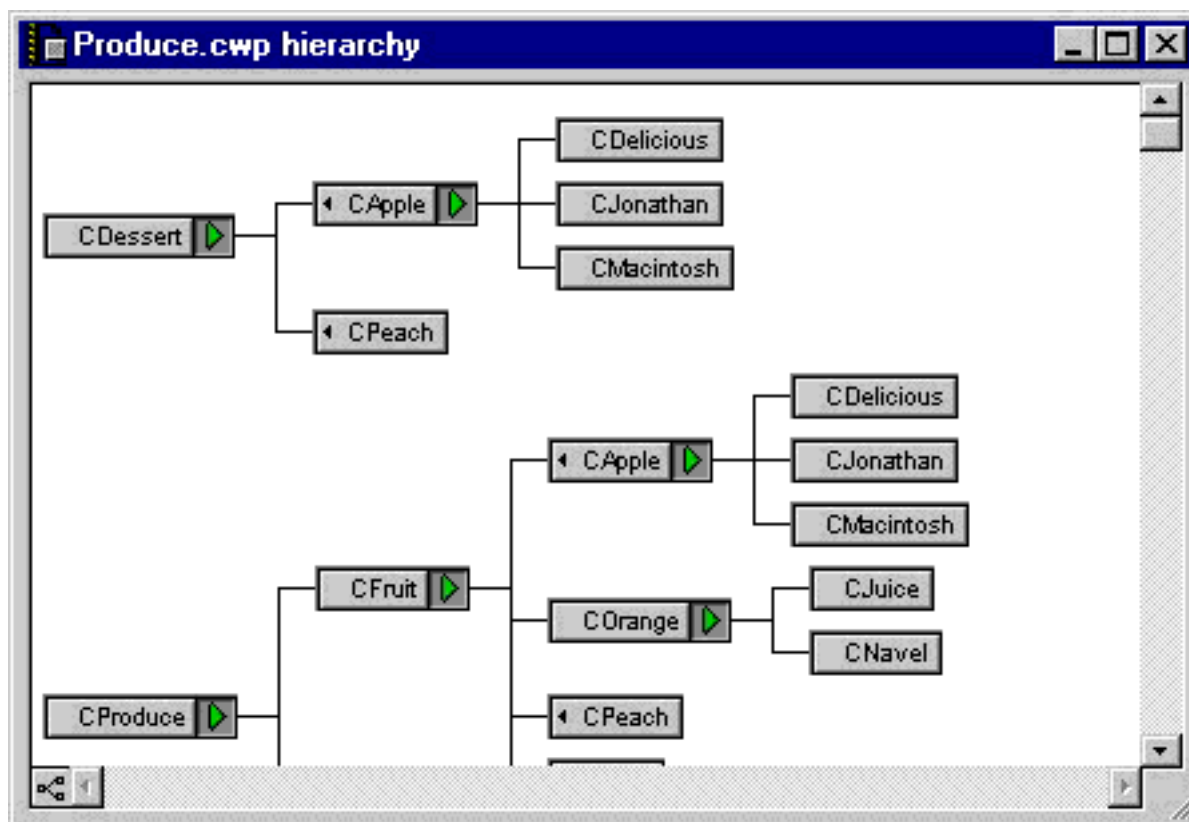
## Browsing Source Code

### *Guided Tour of the Browser*

---

See [“Multi-Class Hierarchy Window” on page 199](#) and [“Single-Class Hierarchy Window” on page 202](#) for details on the hierarchy view interface.

**Figure 7.3** A hierarchy view



## Guided Tour of the Browser

At first glance, the browser interface is quite complicated. There are multiple windows filled with controls and information. However, there are really only three kinds of views: catalog, browser, and hierarchy.

In addition, there is a fundamental and vital feature of the browser interface that makes navigating code simple. If you click and hold the mouse button down on any symbol for which there is data in the

browser database, a pop-up menu appears. The menu lists a variety of destinations related to the symbol.

For object-oriented code (member functions only) , if you click and hold on a routine name, you have the opportunity to see the declaration or definition of any routine with that name. You can also open up a symbol browser that lists every implementation of the routine. Depending upon the nature of the symbol (class name, routine name, enumeration, and so forth), the pop-up menu lists different destinations appropriate for the item. This gives you instant access to the source code related to any symbol.

This section examines each window used by the browser, its controls, and the [Context Pop-Up Menu](#). The sections are:

- [Catalog Window](#)
- [Multi-Class Browser Window](#)
- [Single-Class Browser Window](#)
- [Multi-Class Hierarchy Window](#)
- [Single-Class Hierarchy Window](#)
- [Symbol Window](#)
- [Context Pop-Up Menu](#)

## Catalog Window

The Catalog window displays browser data sorted by category into alphabetical lists. Choose [Show Catalog Window](#) from the [Window Menu](#) to display the Catalog window. [Figure 7.4](#) shows the catalog window.

The items in this window are:

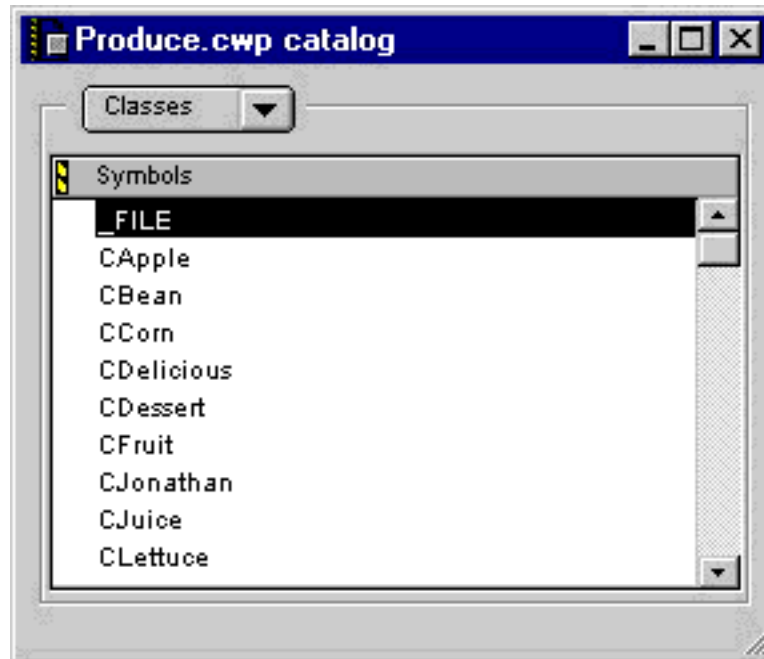
- [Category pop-up menu](#)
- [Symbols pane](#)

## Browsing Source Code

### *Guided Tour of the Browser*

---

**Figure 7.4** A catalog window



#### Category pop-up menu

The Category pop-up menu controls the current type of information on display in the [Symbols pane](#). The currently-selected item is marked with a bullet.

**Figure 7.5** The Category Pop-Up menu



## **Symbols pane**

The Symbols pane displays every item in the browser database that is a member of the currently-selected category. The items are listed alphabetically.

---

**NOTE:** Routines are listed alphabetically by *routine* name, but the class name appears first. As a result, it may appear that the routines are not listed alphabetically.

---

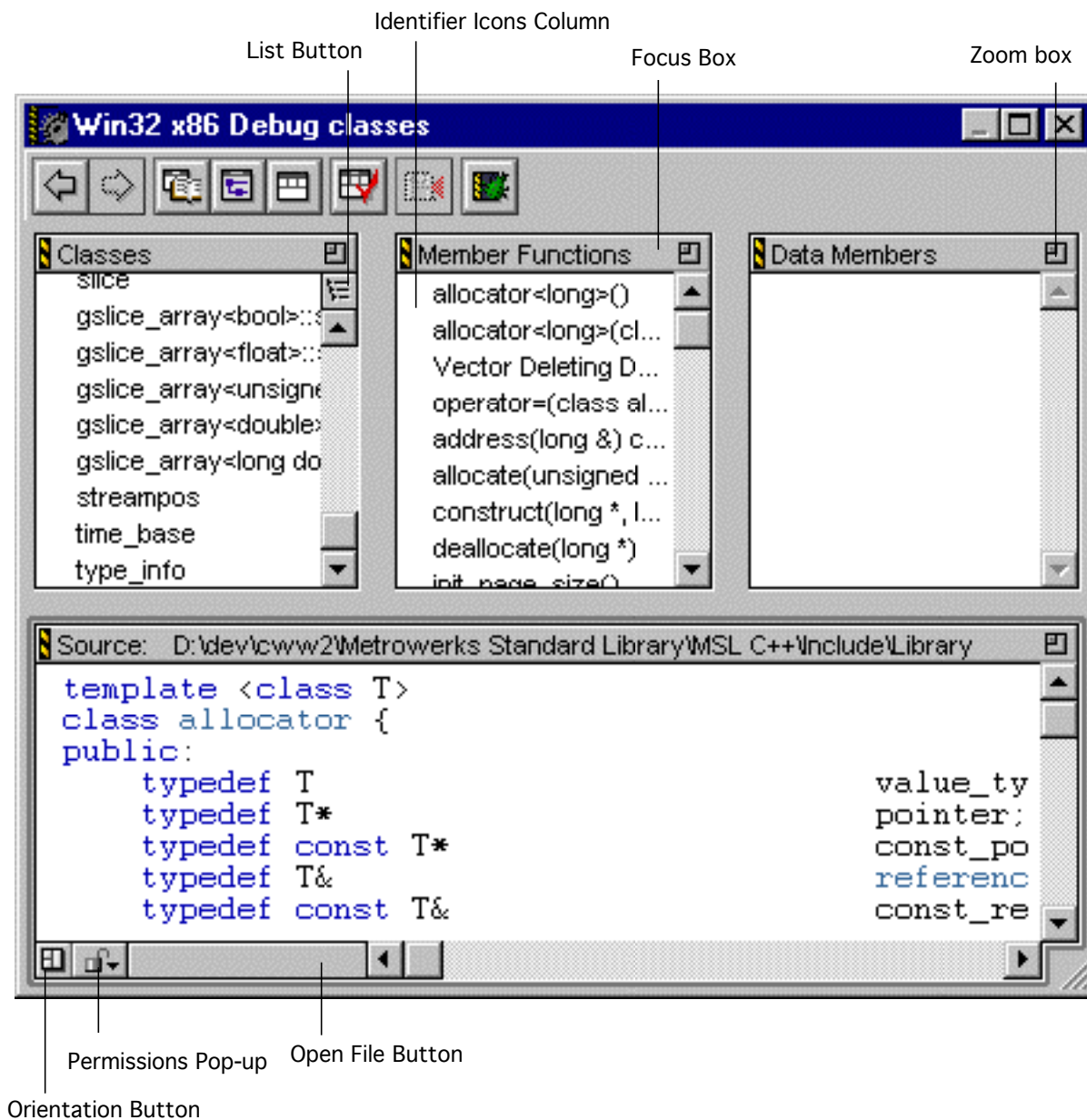
## **Multi-Class Browser Window**

The Multi-Class Browser window gives you a class-based view of every class in the browser database. The window has several panes displaying lists of information. Choose [New Class Browser](#) from the [Window Menu](#) to display the Multi-Class Browser window.

## Browsing Source Code

### Guided Tour of the Browser

**Figure 7.6 The multi-class browser window**



The [Classes pane](#), [Member Functions pane](#), and [Data Members pane](#) are lists of their respective data. The currently active pane has a grey focus box around the pane indicating it is active. You can change the active pane by clicking in the desired pane. You can also use the Tab key to rotate through the panes (except for the Source pane).



---

**TIP:** Using the Tab key can be mildly hazardous to your source code. If the Source pane is active and you press the Tab key, you enter a tab into your source code. Once you are in the Source pane, you can't press the Tab key to get out of it. You must use the mouse.

---

You can click an item in any list to select it, or navigate through the items in the active list by typing or using the arrow keys. You can type in a name, and as you type the selection changes.

The items in this window include:

- [Orientation button](#)
- [List button](#)
- [Open File button](#)
- [Classes pane](#)
- [Member Functions pane](#)
- [Data Members pane](#)
- [Source pane](#)
- [Resize bar](#)
- [Identifier icon](#)

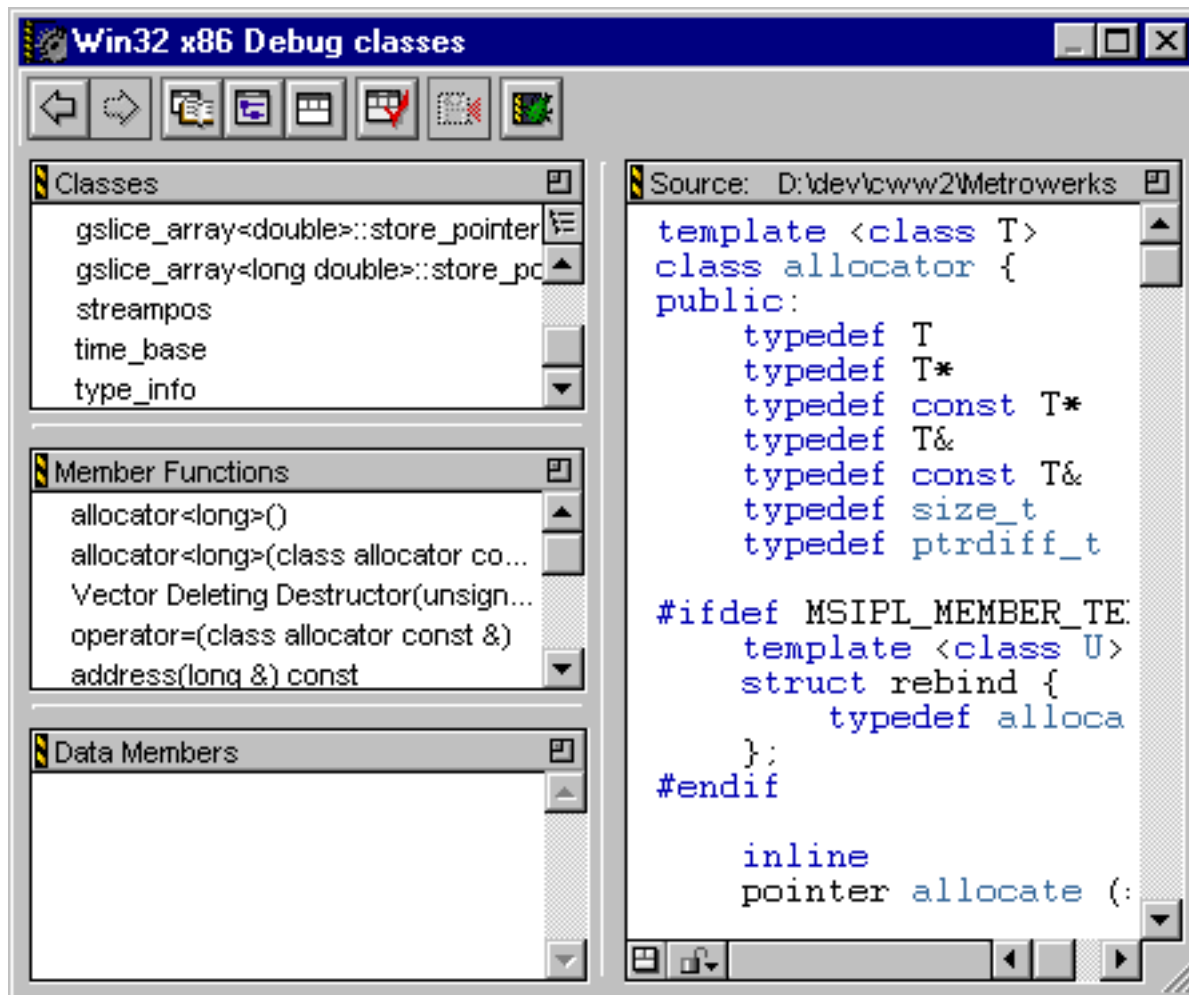
### Orientation button

The Orientation button at the bottom left of the [Source pane](#) controls the distribution of panes in the window. The window may be oriented horizontally, as in [Figure 7.6](#), or vertically as in [Figure 7.7](#).

## Browsing Source Code

### Guided Tour of the Browser

Figure 7.7 The multi-class browser in vertical orientation



#### List button

The List button at the top right of the [Classes pane](#) controls the display of classes. You can toggle between an alphabetical list or a hierarchical list.

#### Open File button

The File button displays the name of the file that contains the code on view in the [Source pane](#). Click this button to open the source file. It uses the CodeWarrior Editor to open the file.

## **Classes pane**

The Classes pane lists all the classes in the browser database.

You can view the list alphabetically or by class hierarchy. You toggle the view by clicking the [List button](#) at the top right of the pane.

In hierarchy view, disclosure triangles appear next to class names that have subclasses. Click the disclosure triangle to show or hide subclasses.

---

**TIP:** Alt-click a disclosure triangle to open all subclasses at all levels. This is a “deep” disclosure. Control-click to display subclasses in a class and all of its siblings. This is a “wide” disclosure. Control-Alt-click to open both wide and deep.

---

When you select a class in the Classes pane, the [Multi-Class Hierarchy Window](#) selection changes too. It will scroll to the newly selected class if necessary.

## **Member Functions pane**

The Member Functions pane lists all member functions defined in the currently selected class. This list does not include inherited functions. In a [Single-Class Browser Window](#), you may display inherited member functions.

Constructors and destructors are at the top of the list. After that, all other entries are alphabetical.

## **Data Members pane**

The Data Members pane lists all data members defined in the currently selected class. This list does not include inherited data members.

The entries in this list are alphabetical. In a [Single-Class Browser Window](#), you may display inherited data members. If inherited members are being displayed, data members are listed by super-class, but alphabetically within each class.

## Browsing Source Code

### *Guided Tour of the Browser*

---

#### Source pane

The Source pane displays the source code for the currently selected item. If the item is a class, this pane shows the class declaration. If the item is a routine, this pane shows the routine definition. If the selected item is a data member, it shows the data member declaration from the interface file.

The text in the source pane is fully editable.

The path to the file that contains the code on display is shown at the top of the pane.

There are two buttons at the bottom of the pane. The [Orientation button](#) modifies the arrangement of panes in the window. The [Open File button](#) opens the source file containing the code on display.




#### Resize bar

A Resize bar appears between each pair of panes. To resize a pane, click and drag the resize bar next to that pane.

#### Identifier icon

A routine or data member may have an identifier icon beside its name. The following table describes the icons.

**Table 7.2**    **Browser identifier icons**

Icon	Meaning	The member is...
	static	a static member
	virtual	a virtual function that you can override, or an override of an inherited function
	pure virtual	a member function that you must override in a subclass if you want to create instances of that subclass

## Single-Class Browser Window

The Single-Class Browser Window gives you a detailed view of one class in the browser database. The window is similar to the [Multi-Class Browser Window](#). [Figure 7.8](#) shows the window.

You can open a single-class browser using several techniques, including:

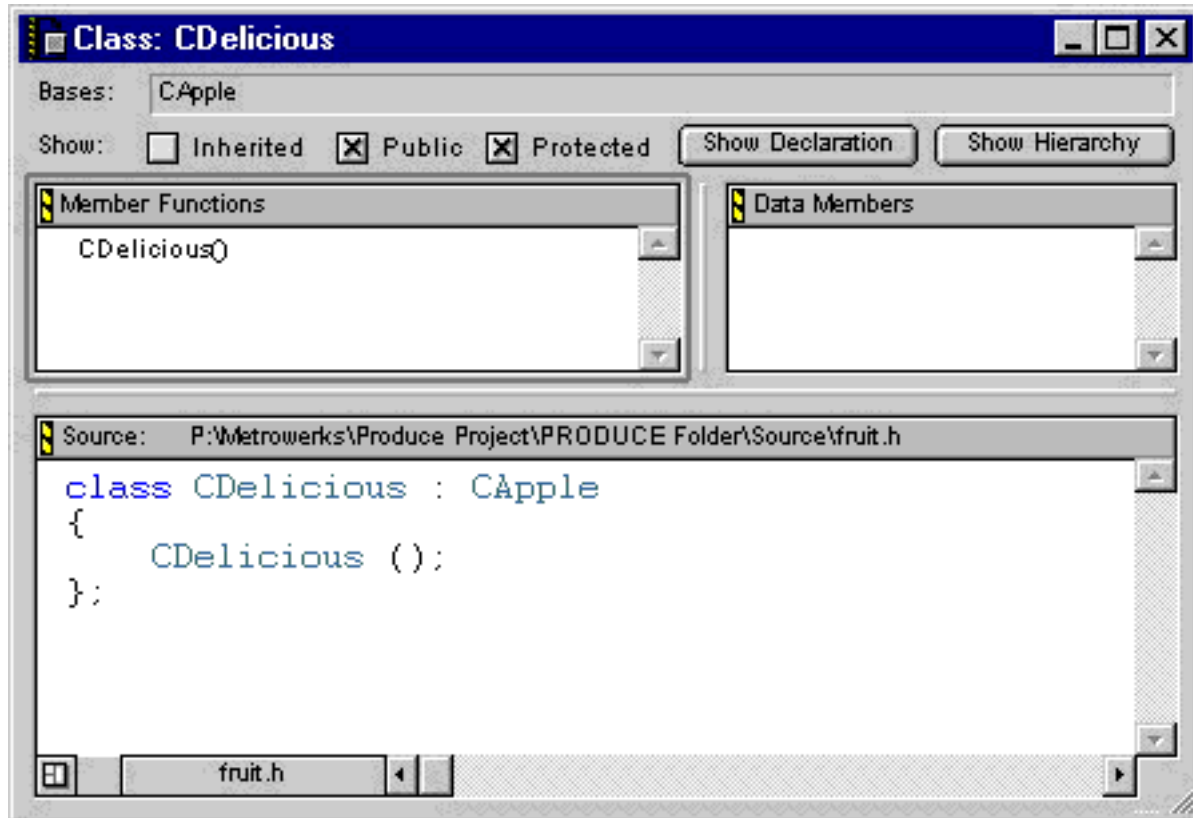
- Use the [Context Pop-Up Menu](#) in the [Catalog Window](#)
- Double-click a class name in a [Multi-Class Hierarchy Window](#)
- Double-click a class name in a [Single-Class Hierarchy Window](#)
- Use the [Context Pop-Up Menu](#) in the [Multi-Class Browser Window](#)

## Browsing Source Code

### *Guided Tour of the Browser*

---

**Figure 7.8** The single-class browser window



The appearance and behavior of this window is similar to the [Multi-Class Browser Window](#). For a discussion of the window in general, see “[Multi-Class Browser Window](#)” on page 191. That topic includes discussion of these items also found in the Single-Class Browser Window:

- [Orientation button](#)
- [Open File button](#)
- [Member Functions pane](#)
- [Data Members pane](#)
- [Source pane](#)
- [Resize bar](#)
- [Identifier icon](#)

This window also has some unique items. They are:

- [Bases text field](#)
- [Show checkboxes](#)
- [Show Declaration button](#)
- [Show Hierarchy button](#)

### **Bases text field**

The Bases text field lists all the immediate base classes for the class on display in the browser window. This list does not include more distant ancestors of the class, only those from which it is a direct and immediate descendant.

This field is informational only and cannot be edited.

### **Show checkboxes**

The check boxes in the Show section of the Single-Class Browser Window control what kinds of data are displayed in the window. You may turn any individual item on or off. The possibilities are: inherited, public, protected, and private.

The choices you make apply to both member functions and data members.

### **Show Declaration button**

If you click the Show Declaration button, the class declaration appears in the [Source pane](#).

### **Show Hierarchy button**

If you click the Show Hierarchy button, you open a [Single-Class Hierarchy Window](#) for the class on display in the Single-Class Browser Window.

## **Multi-Class Hierarchy Window**

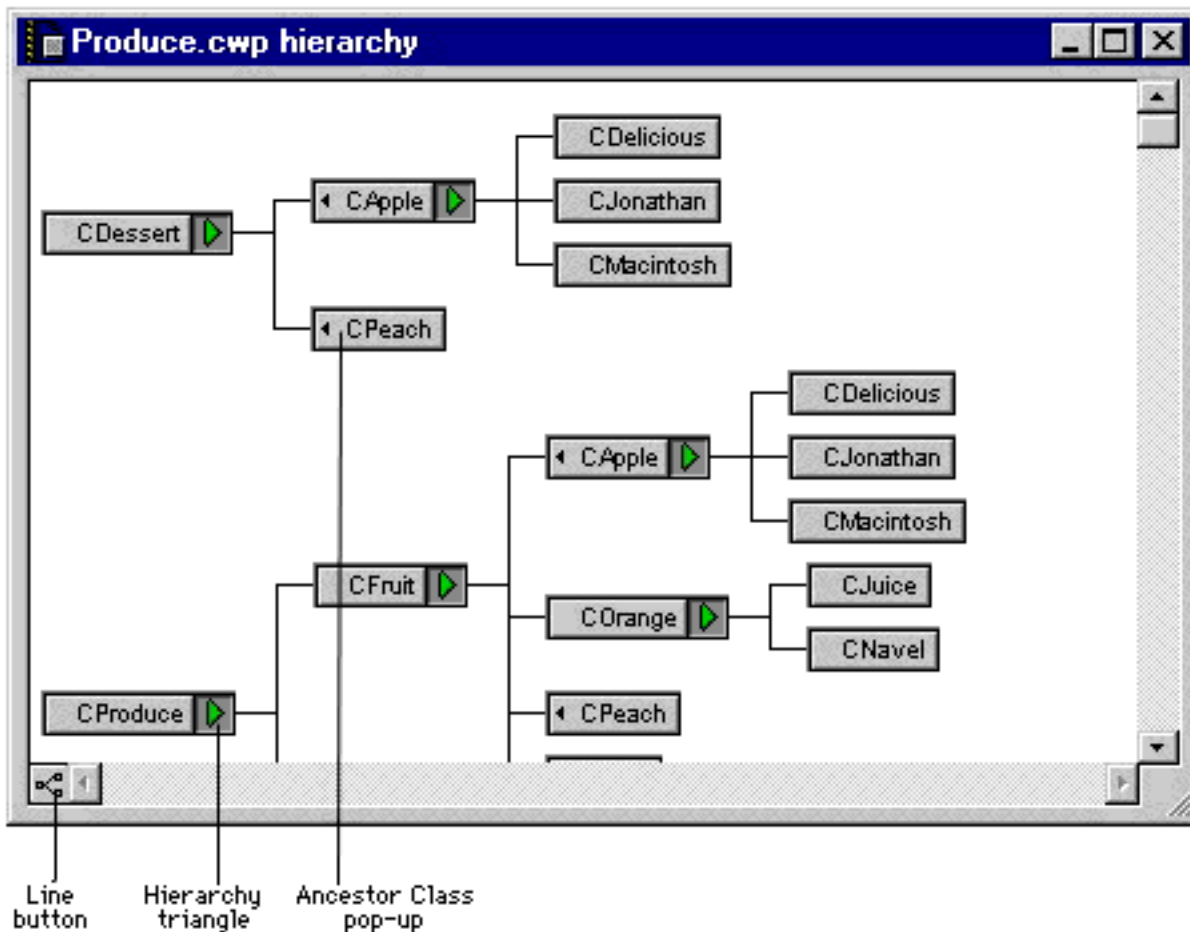
The Multi-Class Hierarchy window displays a complete graphical map of the classes in the browser database. Each class name appears

## Browsing Source Code

### *Guided Tour of the Browser*

in a box, and is connected to other related classes by lines. Choose [Show Hierarchy Window](#) from the [Window Menu](#) to display the Multi-Class Hierarchy window. [Figure 7.9](#) shows the window.

**Figure 7.9 The Multi-Class Hierarchy Window**



Use the arrow keys to change the selected class “geographically.” The up and down keys work on siblings. The left and right keys work on ancestors and descendants.

You can type ahead to change the selection. Use the Tab key to change the selected class alphabetically.



---

**TIP:** If you select a class in the [Classes pane](#) in the [Multi-Class Browser Window](#), the selection in the [Multi-Class Hierarchy Window](#) changes too.

---

If you double-click a class entry, or select the entry and press the Enter key, you open a [Single-Class Browser Window](#) for that class.

In addition to the entry for each class, this window has three items:

- [Line button](#)
- [Hierarchy expansion triangle](#)
- [Ancestor Class pop-up menu](#)

### **Line button**

The Line button controls the appearance of the lines that connect related classes. You can toggle between diagonal lines and straight lines. The choice is entirely aesthetic.

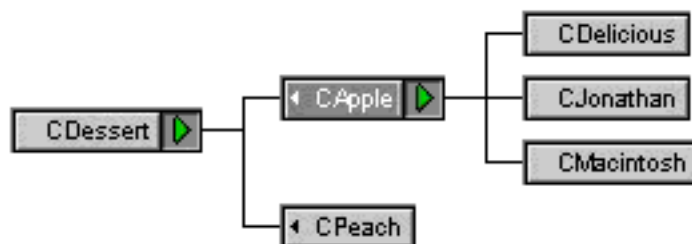
### **Hierarchy expansion triangle**

The Hierarchy expansion triangle controls the display of subclasses.

If you click this triangle, the next level of subclasses appears or disappears. To be more precise, the expanded state restores to what it was the last time this class was open.

For example, if you were to click the CDessert expansion triangle in the state shown in [Figure 7.10](#), all the exposed classes would disappear. If you click it again to open the hierarchy, it will return to the state shown, with the CApple subclasses visible as well as the immediate descendants of CDessert.

**Figure 7.10** An expansion triangle example



---

**TIP:** Alt-click a disclosure triangle to open all subclasses at all levels. This is a “deep” disclosure. Control-click to display subclasses for a class and all of its siblings. This is a “wide” disclosure. Control-Alt-click to open both wide and deep. You can use Control-Alt-click to expand or collapse an entire map if you click the expansion triangle for a base class that has no ancestors.

---

### Ancestor Class pop-up menu

Click on the Ancestor Class triangle to display the pop-up menu. The menu lists immediate ancestors. When you choose an item in the pop-up menu, you jump to that class in the map. If the item is not currently visible, the computer beeps.

This control appears only for classes that have multiple base classes.

### Single-Class Hierarchy Window

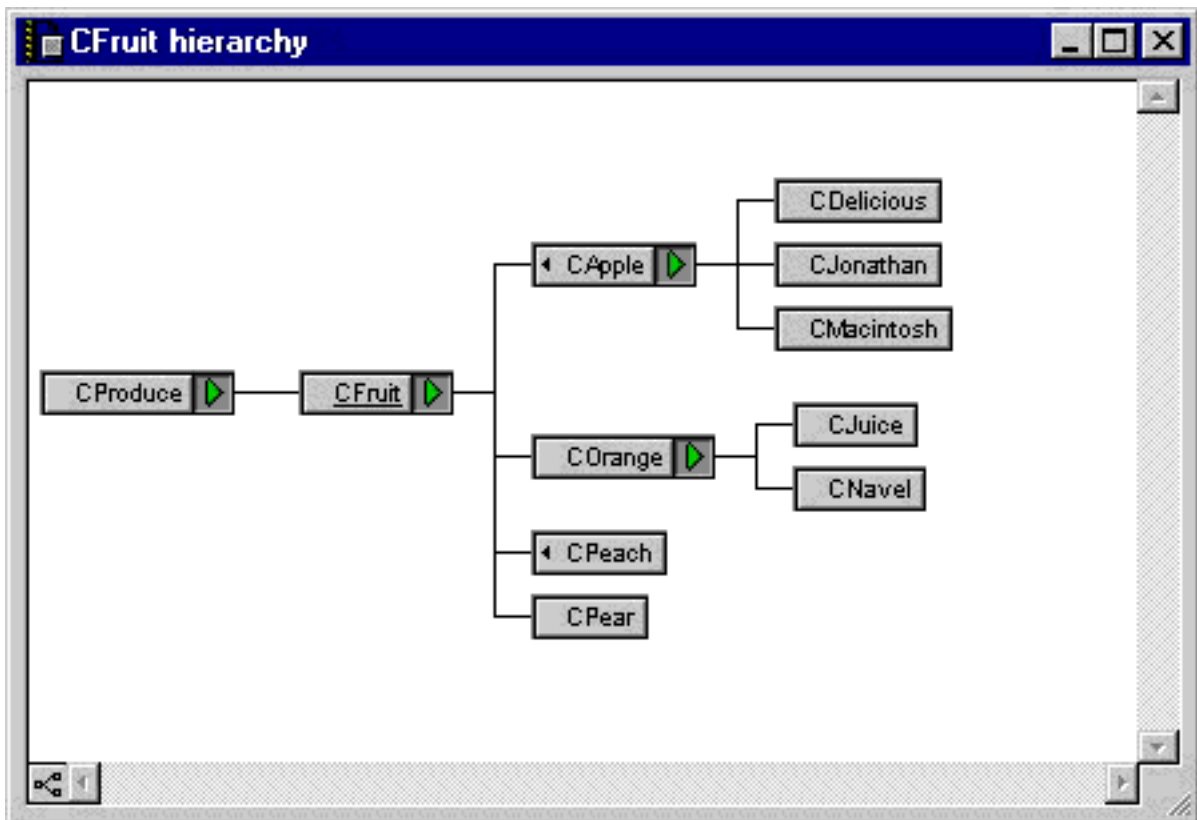
The Single-Class Hierarchy Window displays a complete graphical map for a single class in the browser database. The map displays *all* immediate ancestors of the class, and all of its descendants. (The [Multi-Class Hierarchy Window](#) only shows one base class.)

[Figure 7.11](#) shows the window, displaying multiple base classes and subclasses. The underlined class name is the focus of the window.

You can open a single class hierarchy view using several techniques, including:

- Use the [Context Pop-Up Menu](#) in the [Catalog Window](#)
- Use the [Context Pop-Up Menu](#) in the [Multi-Class Hierarchy Window](#)
- Use the [Context Pop-Up Menu](#) in the [Multi-Class Browser Window](#)
- Use the [Show Hierarchy button](#) in a [Single-Class Browser Window](#)

**Figure 7.11** The Single-Class Hierarchy window



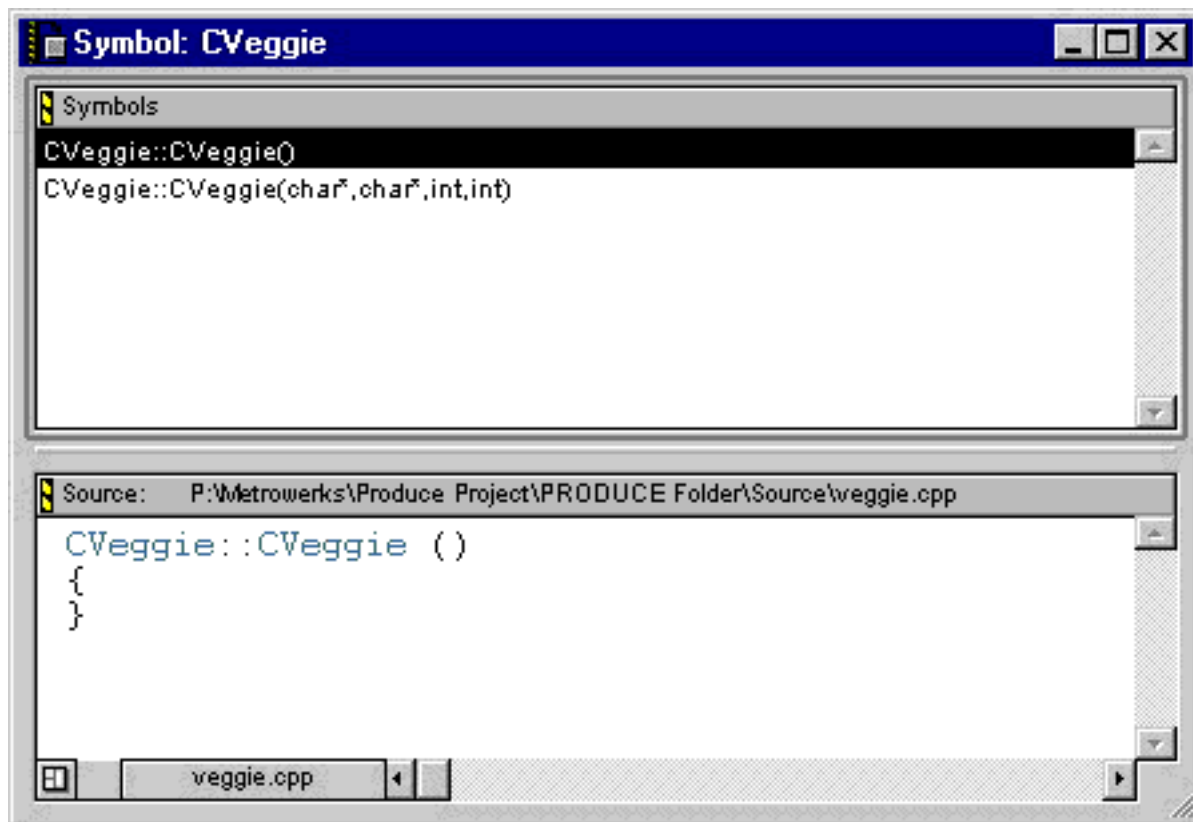
This window is identical to the [Multi-Class Browser Window](#), except that it displays a limited map. For information about how this window behaves, see [“Multi-Class Browser Window” on page 191](#).

## Symbol Window

The Symbol window lists all implementations of any symbol that has multiple definitions. Most commonly, these are multiple versions of overridden functions in object-oriented code. However, the Symbol window works for *any* symbol that is multiply defined in the database.

By selecting an implementation in this list, you see its definition in the Source pane. [Figure 7.12](#) shows this window.

**Figure 7.12** The browser Symbol window



You open a Symbol window by clicking and holding on a symbol name in any browser or editor window. When you do, a [Context Pop-Up Menu](#) appears. If the routine has multiple implementations,

one item in the pop-up menu will be “Find all implementations of.” When you choose that item, the Symbol window appears.

---

**TIP:** In a [Source pane](#) or editor window, Alt-click or Control-click a function or other symbol name to find all implementations and open the Symbol window without using the pop-up menu.

---

Most of the items in this window are identical to the [Multi-Class Browser Window](#). For a discussion of the window in general, see [“Multi-Class Browser Window” on page 191](#). That topic includes discussion of these items, also found in the Symbol window:

- [Orientation button](#)
- [Open File button](#)
- [Source pane](#)
- [Resize bar](#)

This window also has one unique item, the Symbols pane.

### **Routine Symbols pane**

The Symbols pane lists all versions of a routine in the database.

When you select an item in the list, that item’s definition appears in the [Source pane](#).

## **Context Pop-Up Menu**

When the browser is active, click and hold on any symbol for which there is data in the browser database. When you do, a pop-up menu appears with a variety of items.

The nature of the items in the pop-up menu depends upon the nature of the symbol you are investigating. Some items may allow you to open browser windows. In every case, one or more items in the menu direct you to a location in code.

In effect, every symbol in your code—routine name, class name, data member name, constant, enumeration, template, macro, type

## Browsing Source Code

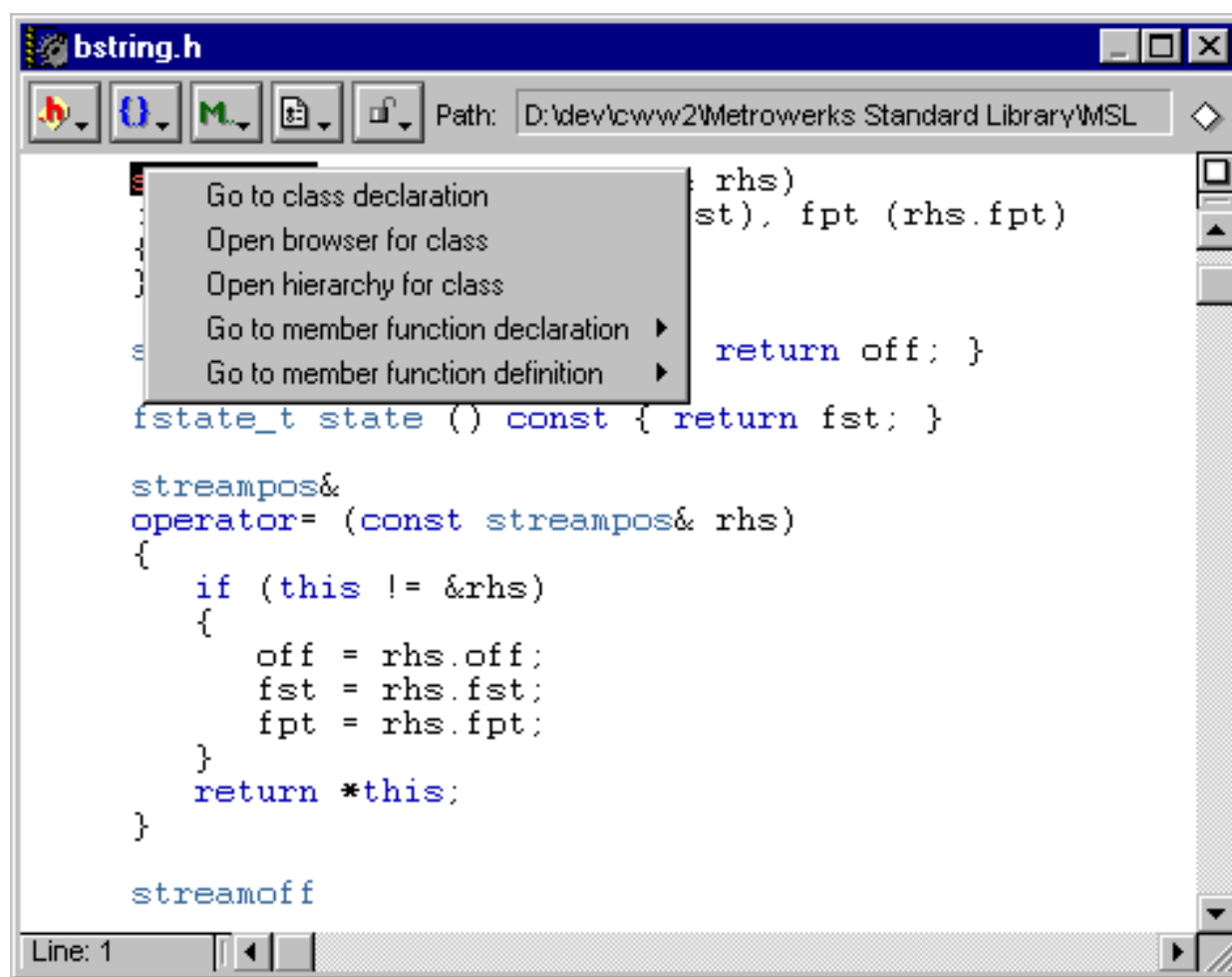
### *Guided Tour of the Browser*

---

definition, and so forth—becomes a hypertext link to a location or multiple locations in your source code.

For example, if you click and hold on a class name, you open the class declaration, open a [Single-Class Browser Window](#), or a [Single-Class Hierarchy Window](#). If you click and hold on a routine name, you get different choices, as shown in [Figure 7.13](#).

**Figure 7.13** The context pop-up menu for a routine



Other menus for other kinds of symbols have items of a similar nature.

---

**TIP:** The context pop-up feature works not only in browser windows, but in the CodeWarrior source code Editor too! This is a great reason for always having the browser enabled, even if you don't use the browser windows.

---

Of particular note in the [Context Pop-Up Menu](#) for routine names is the "Find all implementations of" item. When you choose this item, you open a [Symbol Window](#) in the browser.

## Using the Browser

The browser provides multiple paths through the data related to your code. There is no way that this manual can define all possible browser paths through arbitrary source code. What we can do is give you a feel for how to work with the browser, and outline some techniques you use to accomplish common tasks.

Topics in this section include:

- [Setting Browser Options](#)
- [Navigating Code in the Browser](#)
- [Opening a Source File](#)
- [Seeing a Declaration](#)
- [Seeing a Routine Definition](#)
- [Editing Code in the Browser](#)
- [Analyzing Inheritance](#)
- [Finding Functions That Are Overrides](#)
- [Seeing MFC Classes](#)
- [Saving a Default Browser](#)

### Setting Browser Options

Browser-related menu items and browser-specific options become available when you activate the browser. See "[Activating the Browser](#)" on page 184 for information on how to turn the browser on.

## Browsing Source Code

### *Using the Browser*

---

When the browser is on, browser-related menu items are enabled. These are the [Show Catalog Window](#), [Show Hierarchy Window](#), and [New Class Browser](#) items in the [Window Menu](#).

---

**TIP:** A quick way to tell whether the browser is enabled is to look in the [Window Menu](#) at the browser-related menu items. If they are enabled, the browser is activated.

---

In addition, there are global IDE options that relate to the browser. You control how various items are colored in browser windows, and the time delay before the [Context Pop-Up Menu](#) appears:

- [Browser Coloring](#)
- The Flashing Delay is the amount of time the CodeWarrior Editor displays and highlights an item. It is measured in 60ths of a second. This option is for balancing punctuation. To learn more about balancing punctuation, refer to [“Balancing Punctuation” on page 141](#).

## Navigating Code in the Browser

There are many ways to move around in code with the browser.

### Using the Context Pop-Up Menu

Perhaps most powerful and flexible is the [Context Pop-Up Menu](#). You see this menu when you click and hold on any symbol for which there is data in the browser database. This includes class names, routine names, global variables, class data members, and much more. To learn more refer to [“Context Pop-Up Menu” on page 205](#).

In the [Multi-Class Browser Window](#) and [Single-Class Browser Window](#), simply selecting a class, routine, or data member displays the associated code in the window's [Source pane](#).

### Go Back and Go Forward

The browser fully supports the [Go Back](#) and [Go Forward](#) items in the [Search Menu](#). No matter what views, windows, or code you



have looked at, you can always go back to what you had been viewing earlier.

---

**TIP:** If you have the [Go Back](#) and [Go Forward](#) tools in the , click and hold on the tool icon to see a pop-up menu of all the locations in the go-back queue. You jump directly to any view in the queue.

---

These commands allow you to go backward or forward in a series of changes you made. For example, say you use the Browser to look at your project and make changes to a file. Then, you switch files and make more changes. You may do this many times. You use the [Go Back](#) command to go back one or more actions you have performed. Even if you didn't make any changes to the file, but looked at it (or a specific class or method), you can go back to that action. Similarly, once you've gone back, you can use the [Go Forward](#) command to return you to where you started.

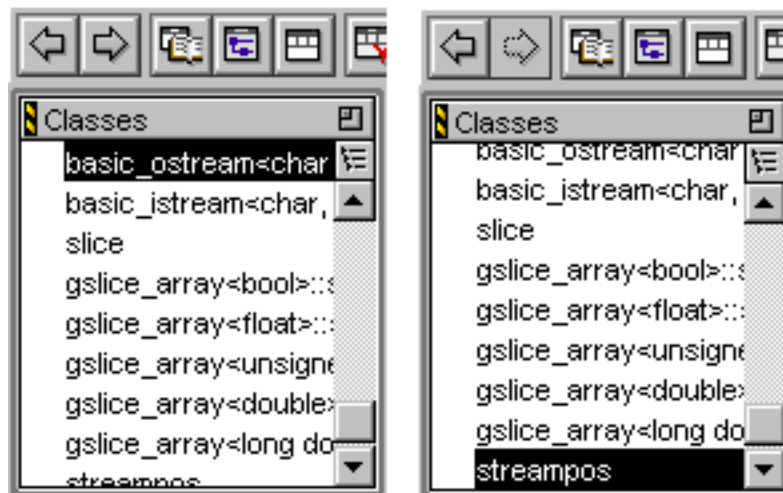
If you add these commands to the CodeWarrior, pressing and holding the mouse button down will show a pop-up menu of the actions you have performed ([Figure 7.14 on page 210](#)).

## Browsing Source Code

Using the Browser

---

**Figure 7.14** Go Back and Go Forward toolbar buttons



Choose any item from the menu to go to that action. If you choose an action out of sequence, CodeWarrior will go to that action *without* going through any previous action.

---

**NOTE:** Go Back and Go Forward do not undo any actions you performed. They allow for a more flexible method of moving around to specific places you have been in the Browser window.

---

## Opening a Source File

There are at least three quick methods you use to open the source file you want to see.

In the [Multi-Class Browser Window](#) or [Single-Class Browser Window](#), click the [Open File button](#) when portions of the file you wish to see are displayed in the window's [Source pane](#).

Click on a symbol used in the file, and hold the mouse button. Then use the [Context Pop-Up Menu](#) to open the desired file or see a particular routine.

## Seeing a Declaration

There are several methods used to see a declaration. The methods vary depending upon the kind of symbol you are investigating.

If you select a class name or data member name in a [Multi-Class Browser Window](#) or [Single-Class Browser Window](#), and the declaration appears in the [Source pane](#). To open the file that contains the declaration, double-click the name. (If you select or double-click a routine name, you see the definition).

If you click the [Show Declaration button](#) in the [Single-Class Browser Window](#), you see a class declaration.

If you click and hold on a name in any window. Then use the [Context Pop-Up Menu](#) to open the declaration. Use this technique to see a routine declaration.

## Seeing a Routine Definition

There are several methods you can use to see a routine definition.

In the [Multi-Class Browser Window](#) or [Single-Class Browser Window](#), select the routine in the [Member Functions pane](#). The definition appears in the [Source pane](#). To open the file that contains the definition, double-click the routine.

Click and hold on the routine name in any editor or browser window. Then use the [Context Pop-Up Menu](#) and you jump to the particular routine definition.

Choose the [Go Back](#) item in the [Search Menu](#).

Alt-double-click a function name in any source view to open the [Symbol Window](#) with all implementations of the function. You can use Control-double-click to do the same thing.

In the [Multi-Class Hierarchy Window](#) or the [Single-Class Hierarchy Window](#), click and hold on a class name. Then use the [Context Pop-Up Menu](#) to jump to the particular routine definition.

### Editing Code in the Browser

Any code visible in a [Source pane](#) is fully editable. Locate the definition you wish to work with. When the code appears in the [Source pane](#), use the same techniques you would in any CodeWarrior Editor window.

For more information about the CodeWarrior Editor, see [“Source Code Editor Overview” on page 117](#).

### Analyzing Inheritance

Use the [Multi-Class Hierarchy Window](#) or the [Single-Class Hierarchy Window](#). Look for the little triangle on the left of a class name that indicates the class has multiple ancestors. Use the associated [Ancestor Class pop-up menu](#) to jump to any ancestor class to study its descendants (or ancestors).

Use the [Hierarchy expansion triangle](#) to expose or conceal subclasses.

### Finding Functions That Are Overrides

Use the [Single-Class Browser Window](#). Turn off the display of inherited functions. This turns off the display of unchanged inherited functions. It does not turn off functions you have overridden.

Then look for functions that are marked as virtual with an [Identifier icon](#). Most of these functions are likely to be overrides of inherited functions, although some could be functions you declared in this class that were not inherited from an ancestor.

To open a [Symbols pane](#) window for any routine, click and hold on a routine name, and then choose the “Find all implementations of” item. Combined with a hierarchy view, you see precisely who overrides the routine, and where they are in the class hierarchy.

## Seeing MFC Classes

If you want the browser to display MFC classes, you have to include MFC headers in such a way that the compiler sees them.

You should use precompiled headers to speed compilation time. If you simply use the precompiled headers directly, however, the compiler does not see MFC classes and does not generate information for the symbol database.

The way around this problem is to include a renamed copy of the source file used to create the precompiled headers in your project. These files have a name like `myFile.mch`.

This creates a project-specific precompiled header. By including the source file, CodeWarrior builds the precompiled header from within your project, exposing MFC symbols to the compiler, which generates information for the browser database.

The reason you want to include a renamed copy is to avoid problems if you have multiple projects that use the same precompiled headers. When one project updates the headers, it will be marked as changed for all other projects, which then rebuild the precompiled headers. This defeats the purpose of the precompiled header.

## Saving a Default Browser

The browser windows all have various settings that you can modify. You can preserve these as your default settings.

Simply set up a browser window the way that you like. For example, in a [Multi-Class Browser Window](#), set the orientation, the size of each pane, and the size and location of the window. Then choose [Save Default Window](#) from the [Window Menu](#). The next time you open a multi-class browser window, it will take on the attributes you just set.

You can do the same for any of the browser windows. You must save each window's setup individually, while that window is the active window.

## Customizing the Browser

This feature was under construction at the time this manual was printed. Check the release notes for the latest information.

To learn more about customizing the toolbar icons that are visible on the browser windows, refer to [“Toolbar Customization” on page 269](#).



# Configuring IDE Options

---

This chapter discusses the many options available in the CodeWarrior IDE's [Preferences](#) and [Target Settings](#) dialog boxes, found on the [Edit Menu](#).

## Configuring IDE Options Overview

You control many features of the CodeWarrior IDE. The CodeWarrior IDE has two dialog boxes for handling options that you can configure. One dialog box handles global preferences. The other handles target-specific settings. This chapter discusses all the options available in these dialog boxes.

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 CodeWarrior target within your project that you are working with, you choose the [Target 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 CodeWarrior Editor. Another panel controls the target settings for your compiled code.

To learn about how to create Project stationery using the [Target Settings](#) menu command, so that your favorite option settings will be used when you create a new project, refer to [“About Project Stationery” on page 46](#).

The topics in this chapter include:

- [Option Dialog Boxes Guided Tour](#)

- [Choosing Preferences](#)
- [Choosing Target Settings](#)
- [Toolbar Customization](#)

## Option Dialog Boxes Guided Tour

The various options available to you for configuring the CodeWarrior IDE settings are found on the [Edit Menu](#), using the [Preferences](#) and [Target Settings](#) commands.

The topics in this section are:

- [Options Panels](#)
- [Dialog Box Buttons](#)

### Options Panels

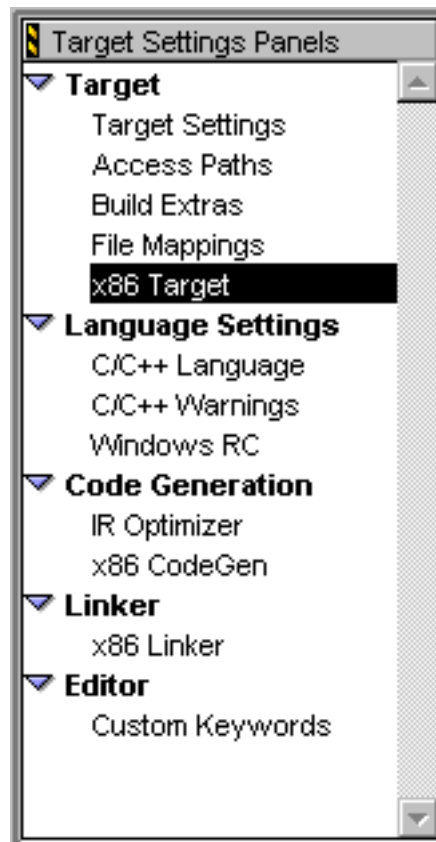
To see an options panel that you configure to set your CodeWarrior IDE options, choose the [Target Settings](#) command from the [Edit Menu](#) (the name of the command that appears in the menu may vary with the name of your targets, but ends with the word “Settings...” next to it). The actual options panels available to you may vary, depending upon the CodeWarrior IDE product you are using.

A typical [Target Settings](#) panel for Windows-based platform development, for example, might allow you to configure C/C++ Warnings, IR Optimizer, x86 CodeGen, x86 Linker, Custom Keywords, Access Paths, File Mappings, Build Extras, WinRC Resource Compiler options, and C/C++ Compiler options.

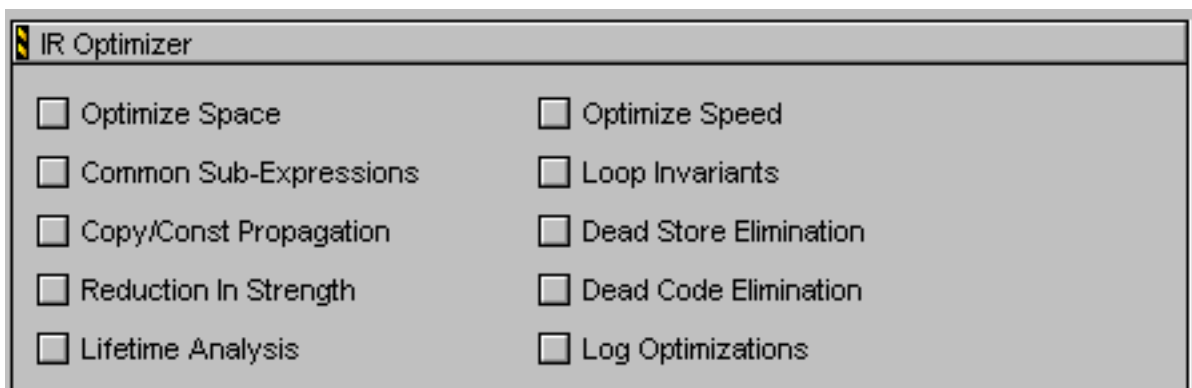
To select one of these to configure, click on the name of the panel in the list box on the left side of the dialog box, for the panel of interest to you. Refer to [Figure 8.1](#) for an example that shows the [Target Settings](#) panel list you can choose from. If you click on IR Optimizer, you will see a panel resembling [Figure 8.2](#).



**Figure 8.1**    **Selecting an Options Panel**



**Figure 8.2**    **An Options Panel**



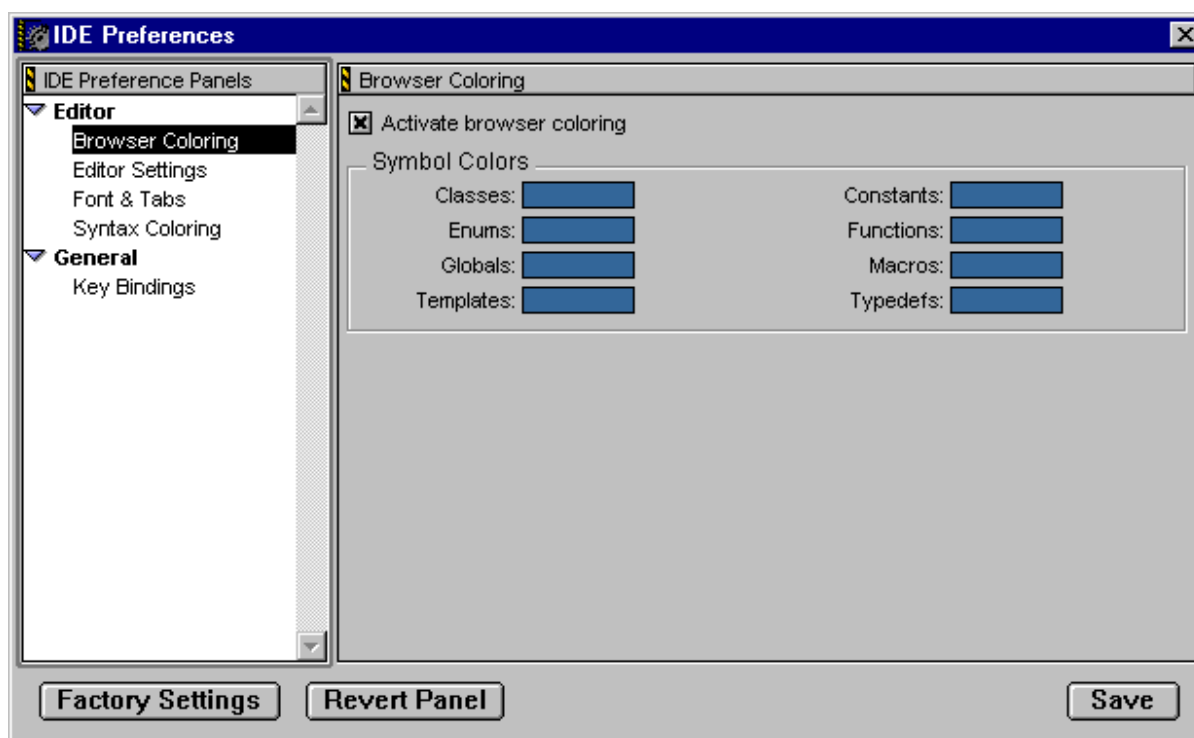
## Configuring IDE Options

### *Option Dialog Boxes Guided Tour*

---

To set global preferences for all projects, you use the [Preferences](#) menu command under the [Edit Menu](#). This command causes the dialog box shown in [Figure 8.3](#) to appear. This dialog box operates similarly to the [Target Settings](#) dialog box, but is used instead for changing preferences that are global to all projects in the IDE, not just local to a project that is currently open.

**Figure 8.3** Preferences Dialog Box



You can create project stationery that contains your favorite preferences, so that the factory default preferences will not be used when you create a new project. To learn more about this topic, refer to the discussion [“Creating Your Own Project Stationery” on page 52.](#)

## Dialog Box Buttons

There are several buttons in the dialog box to control how a panel's options are used and applied.

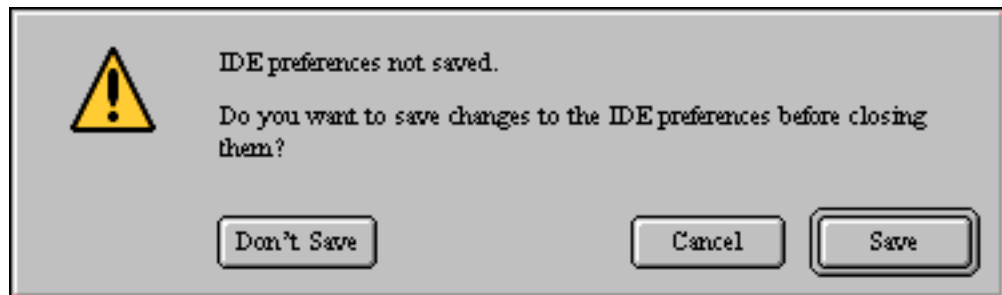
The topics in this section are:

- [Discarding Changes](#)
- [Factory Settings Button](#)
- [Revert Panel Button](#)
- [Save Button](#)

### **Discarding Changes**

If you make changes on a panel, then change your mind about them, you can click the close box of the dialog box to close the dialog box. A dialog box similar to that shown in [Figure 8.4](#) may be shown. To discard your changes, click the Don't Save button. To keep your changes, click the Save button. To keep the preferences dialog box open so that you may continue making changes, click Cancel.

**Figure 8.4 Preferences Confirmation Dialog Box**



### **Factory Settings Button**

The Factory Settings button causes the dialog box panel to revert to the settings that the CodeWarrior IDE uses as the defaults. If you click this button, you will reset the setting in the panel to a known state. Settings in other panels of this dialog box are not affected by this button. Only the settings for the options you are currently viewing will be reset.

### **Revert Panel Button**

The Revert Panel button allows you to reset the state of the current panel you're viewing to the settings it had when you first viewed it.

## Configuring IDE Options

### Choosing Preferences

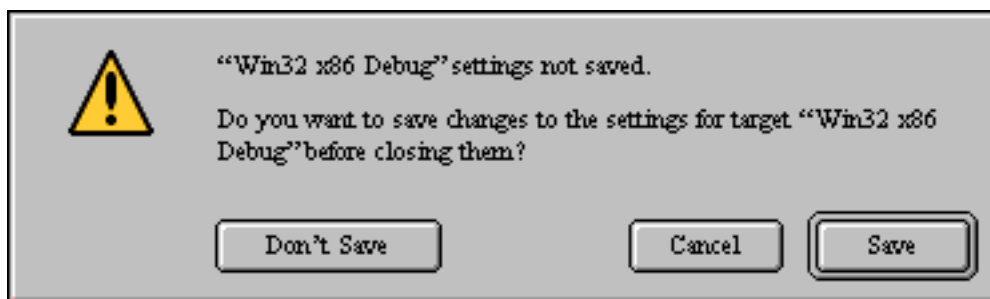
---

This is useful if you start making changes to a panel, then decide you don't want to commit them.

#### Save Button

The Save button commits any changes you made in any of the options panels. If you changed an option that will require the project be recompiled, you will see the dialog box similar to that shown in [Figure 8.5](#). Click Save, Don't Save, or Cancel depending on whether you want to keep your changes or not.

**Figure 8.5** Save Changes Dialog Box



To close the dialog [Preferences](#) or [Target Settings](#) box, click the close box in the upper left corner of the dialog box.

## Choosing Preferences

This section discusses setting preferences for your project. Here you will learn how to configure preferences for the Editor, and for things such as the user interface key bindings that will invoke a given menu command, or perform a command in the Editor.

To configure the preferences panels, first select the [Preferences](#) command from the [Edit Menu](#). Then, select the category of the preferences you want to configure, by clicking on the name of the desired preferences group in the list box on the left side of the dialog box.

#### Editor Preferences

- [Editor Settings](#)

- [Fonts and Tabs](#)
- [Syntax Coloring](#)
- [Browser Coloring](#)

### General Preferences

- [Key Bindings](#)

## Editor Settings

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 8.6](#).

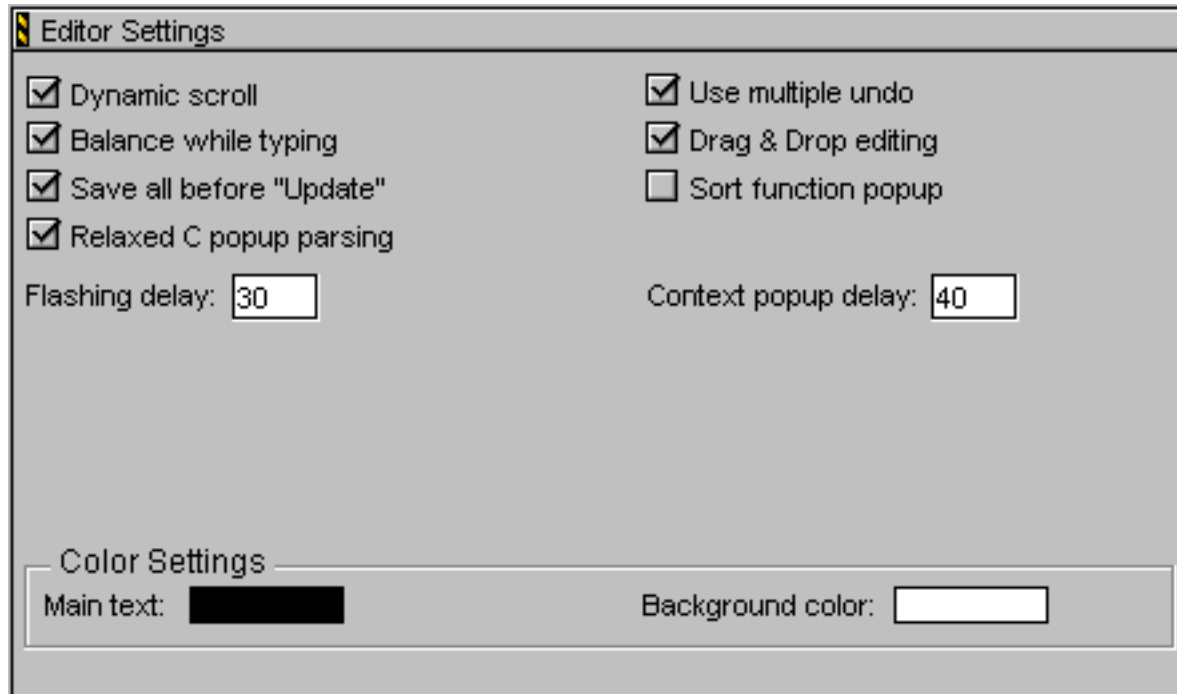
This preference panel has three areas of options, miscellaneous editor settings, Remember, and Color Settings. Editor settings (such as Dynamic Scroll and Balance While Typing) control how the Editor works. The Remember group of preferences determines what state information is saved for editor windows.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

## Configuring IDE Options

### Choosing Preferences

---

**Figure 8.6 Editor Settings 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 scrollbar and jumping to the new location when you release the thumb.

#### Balance While Typing

When the Balance While Typing option is enabled, the CodeWarrior IDE 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.

To learn more about [Flashing Delay](#), refer to [“Flashing Delay” on page 224](#).

---

**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.

### Relaxed C Popup Parsing

Lets the [Routine Pop-Up Menu](#) parser recognize K&R style functions. To learn more about this feature, refer to [“Routine Pop-Up Menu” on page 121](#).

There are some commonly used macros, particularly in Win32 / MFC programming but also elsewhere, that sometimes confuse the routine pop-up parser. With Relaxed C Popup Parsing turned off, the parser usually won't get confused by the macros, but will not recognize K&R (Kernighan and Richey) old-style C functions. With this option turned on, it will recognize them, but may show some aberrant results if you have functions defined using complex macros.

A K&R function definition looks like:

---

```
int func(x)
int x;
{
}
```

---

which is a different form than the ANSI-standard way:

## Configuring IDE Options

### *Choosing Preferences*

---

---

```
int func(int x)
{
}
```

---

### Use Multiple Undo

Enable this option if you want to use the multiple undo feature. When active, the [Undo](#) command and [Redo, Multiple Undo, and Multiple Redo](#) command are separate items with separate command keys. If this option is off, the Undo command works as normal. See [“Redo, Multiple Undo, and Multiple Redo” on page 313](#) 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 60ths of a second. This option is for balancing punctuation. To learn more about balancing punctuation, refer to [“Balancing Punctuation” on page 141](#).

### 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. This value is in “ticks” which are the same as 1 / 60th of a second (16.67 milliseconds).

---

**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 205](#) and [“Using the Context Pop-Up Menu” on page 208](#).



### **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 136](#).

### **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 121](#).

### **Main Text Color**

This option configures the color of any text not colored by the [Browser Coloring](#), [Syntax Coloring](#), or [Custom Keywords](#) color sets. Click the color area to change the color, and you will see with the color changing dialog as shown in [Figure 8.9 on page 228](#).

### **Background Color**

Click the color area to change the background color of the Editor and Browser windows. You will see the color changing dialog as shown in [Figure 8.9 on page 228](#).

### **Fonts and Tabs**

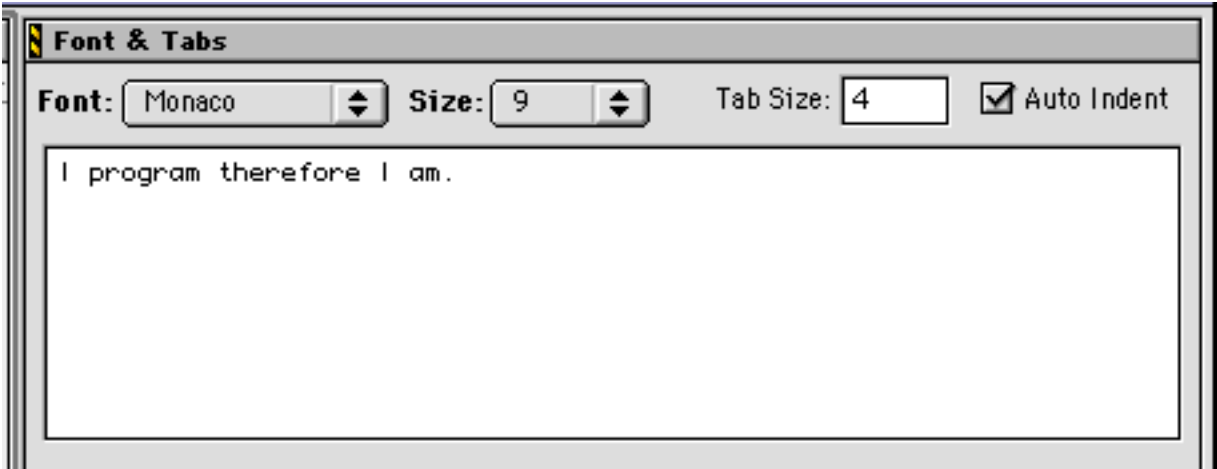
To change the settings for Fonts and Tabs you use the preference panel shown in [Figure 8.7](#). This preference panel sets the font and tab information for the active Editor window.

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.

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 the CodeWarrior IDE inserts to make up a ‘tab’ character using the Tab key.

Figure 8.7    Fonts and Tabs Options Panel

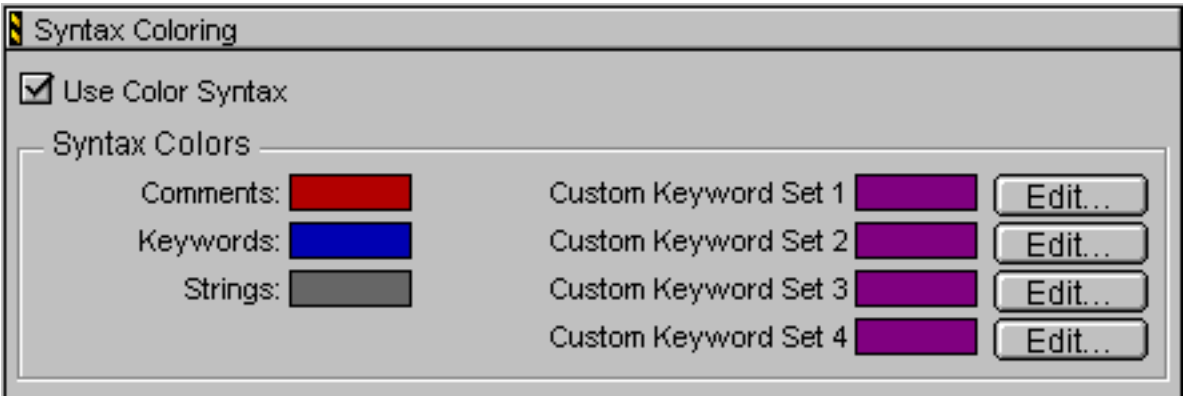


**Syntax Coloring**

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 8.8](#).

Figure 8.8    Syntax Coloring Options Panel



[Table 8.1](#) describes each element of text that the CodeWarrior Editor 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.
Comments	Code comments. In Java, C or C++, a comment is text enclosed by <code>/*</code> and <code>*/</code> or text from <code>//</code> to the end of the line. In Pascal, a comment is text enclosed by <code>{</code> and <code>}</code> or <code>(*</code> and <code>*)</code> .
Keywords	The language's keywords. It does not include any macros, types, or variables that you or the system interface files define.
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.

### **Changing syntax highlighting colors**

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

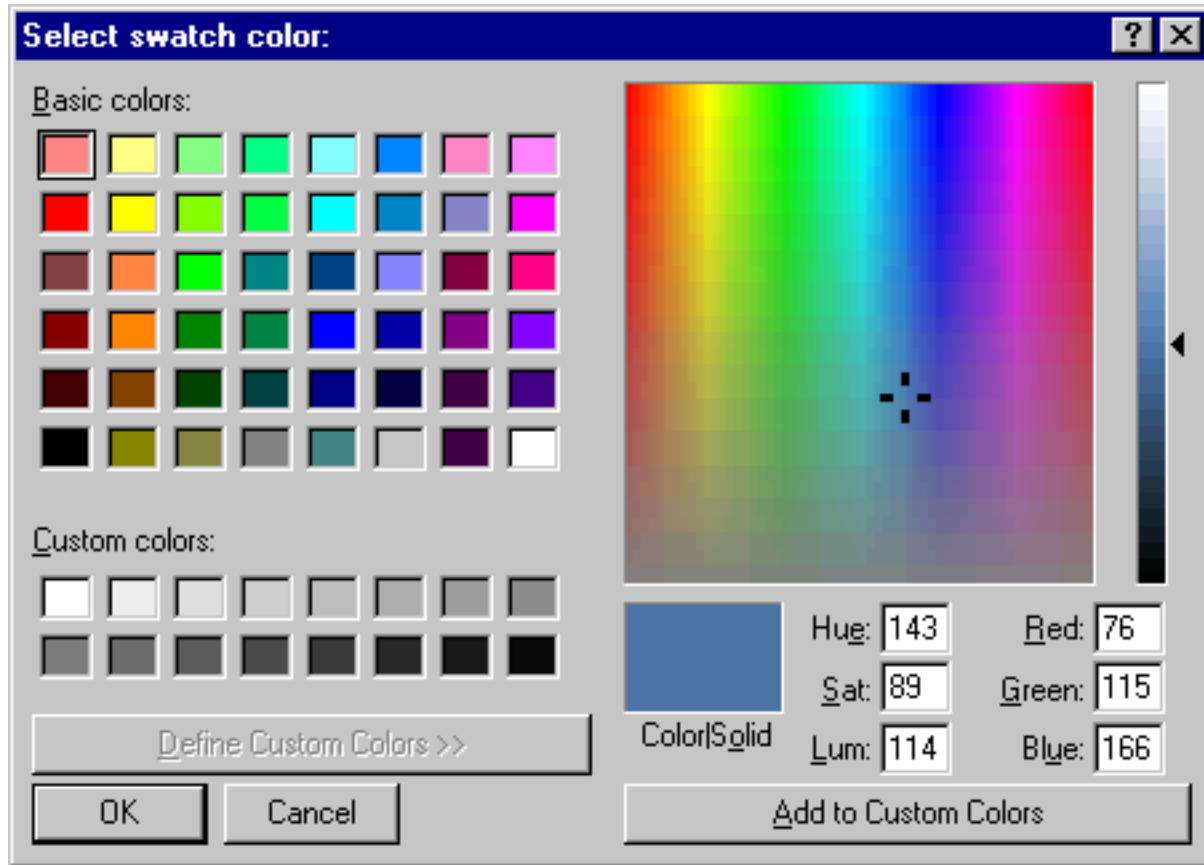
the CodeWarrior IDE uses different colors for each type of text. To change these colors, click on the color sample beside the name. the CodeWarrior IDE displays a dialog box ([Figure 8.9](#)) you use to select string color. The next time you view a text file, the CodeWarrior IDE uses the new color.

## Configuring IDE Options

### Choosing Preferences

---

**Figure 8.9** Select color dialog box



### Controlling syntax highlighting within a window

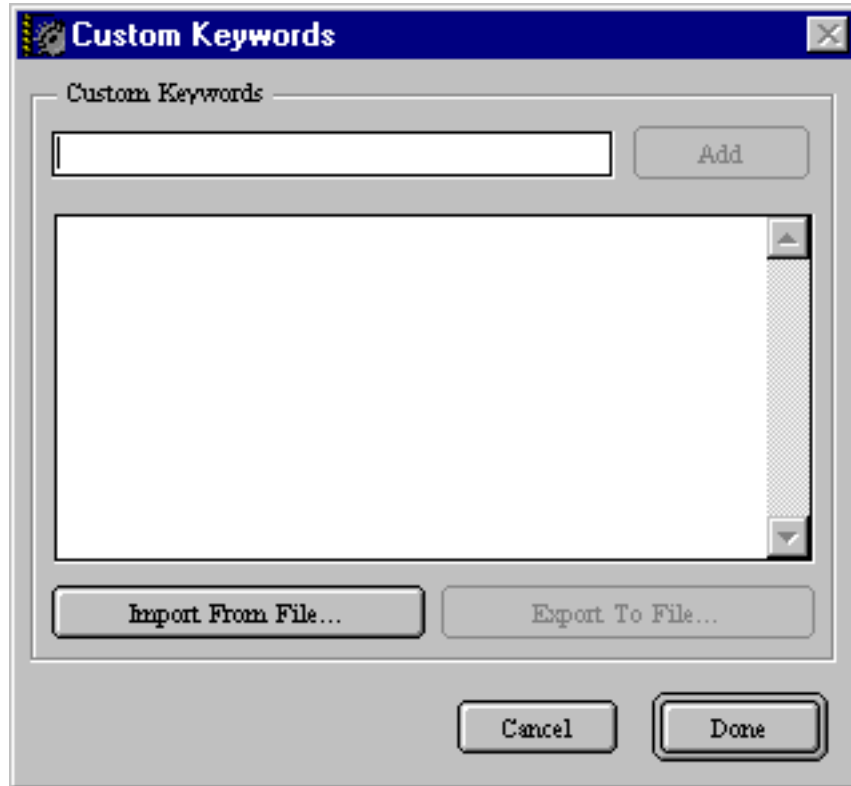
You can learn how to turn syntax coloring on or off by reading [“Adding, Removing, and Selecting a Marker” on page 144.](#)

### 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 the CodeWarrior IDE and will apply to every project. See also [“Custom Keywords” on page 245.](#)

In the options panel, click the Edit button to the right of the Custom Keyword Set you want to modify. The CodeWarrior IDE displays the Custom Keywords List dialog box, shown in [Figure 8.10](#).

**Figure 8.10** Custom keyword dialog box



Type a keyword in the Add field, then click Add. the CodeWarrior IDE 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. the CodeWarrior IDE removes the keyword from the Custom Keywords List.

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

## Configuring IDE Options

### *Choosing Preferences*

---

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

To retrieve or save an entire group of keywords, use the Import from File and Export to File buttons.

Click the Edit button to the left of the appropriate Custom Keyword Set (shown in [Figure 8.8 on page 226](#)).

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

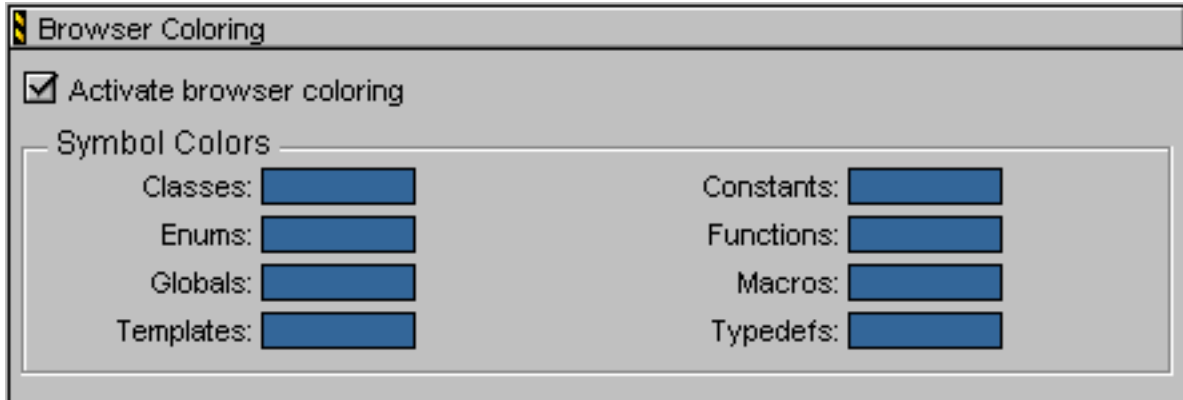
the CodeWarrior IDE adds or subtracts the custom keywords from that file to or from your Custom Keyword Set.

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

#### **Browser Coloring**

The Browser Coloring preferences are shown in [Figure 8.11](#). 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 box (shown in [Figure 8.9 on page 228](#)) to change the color.

**Figure 8.11 Browser highlighting options**



### **Enable Automatic Toolbar Help**

This option turns on Balloon Help for the toolbar icons. When you move the mouse over an icon and leave it there for a second or two, a balloon will pop up and tell you what command the icon represents.

## **Key Bindings**

The CodeWarrior IDE has customizable key bindings for menu, keyboard, and editor commands. You can configure the IDE to suit your personal working style using this feature. This feature will allow you to change the keys that activate many commands in the CodeWarrior IDE to key bindings that you find convenient for your working style.

For example, you may wish to change the menu command for Save from Control-S to Control-Alt-S. You could change this using the key bindings feature.

The key bindings for the menu commands are configurable via the File Menu, Edit Menu, Search Menu, Project Menu, and Window Menu categories in the options panel shown in [Figure 8.12](#). There are other key bindings that can be configured, using the Misc, Editor Commands, and Prefix Keys categories in the options panel.

The topics discussed in this section are:

## Configuring IDE Options

### *Choosing Preferences*

---

- [Restrictions for Choosing Key Bindings](#)
- [What is a Prefix Key?](#)
- [What is a Quote Key?](#)
- [Modifying Key Bindings](#)
- [Setting the Prefix Key Timeout](#)
- [Exporting Key Bindings](#)
- [Importing Key Bindings](#)
- [Configuring Misc Bindings](#)
- [Configuring Prefix Keys Bindings](#)
- [Configuring the Quote Key Binding](#)

### **Restrictions for Choosing Key Bindings**

When you are customizing key bindings, you need to be aware of some restrictions for keys that can and can not be used. These restrictions are listed here.

This section is under construction. As this manual went to press, some of the limitations of this feature were not yet characterized.

The following represents possible advice for using this feature:

Key bindings that use printing characters must use at least the Command or Control key in the key binding. Printable characters such as the following are not valid for key binding assignment:

- 'a'
- Shift-'a'

This printable character restriction does not apply for the second key of a two-key sequence.

The Alt key is not valid for use in Key Bindings.

The Return and Tab keys require at least the Control or Shift key. This restriction does not apply for the second key of a two-key sequence.

The Escape and Space keys are always invalid for key bindings.



Function keys and the Clear key are valid for creating key bindings.

### **What is a Prefix Key?**

Prefix Keys are for supporting multiple-keystroke key bindings, such as those of the Emacs text editor available on many different computer platforms. For example, the key sequence in Emacs to save a file is Control-X followed by Control-S.

To emulate this Emacs key binding in the CodeWarrior IDE, first set one of the Prefix Keys to be Control-X, and then set the key binding for the [Save](#) menu command to Control-X Control-S.

To learn how to configure Prefix Keys, refer to [“Configuring Prefix Keys Bindings” on page 243](#). You can also adjust the maximum time to wait for a key press after a Prefix Key is entered. To learn how to do this, refer to [“Setting the Prefix Key Timeout” on page 236](#).

### **What is a Quote Key?**

Information on this topic was not available at the time this manual was prepared for publication. Check the release notes for the latest information.

### **Modifying Key Bindings**

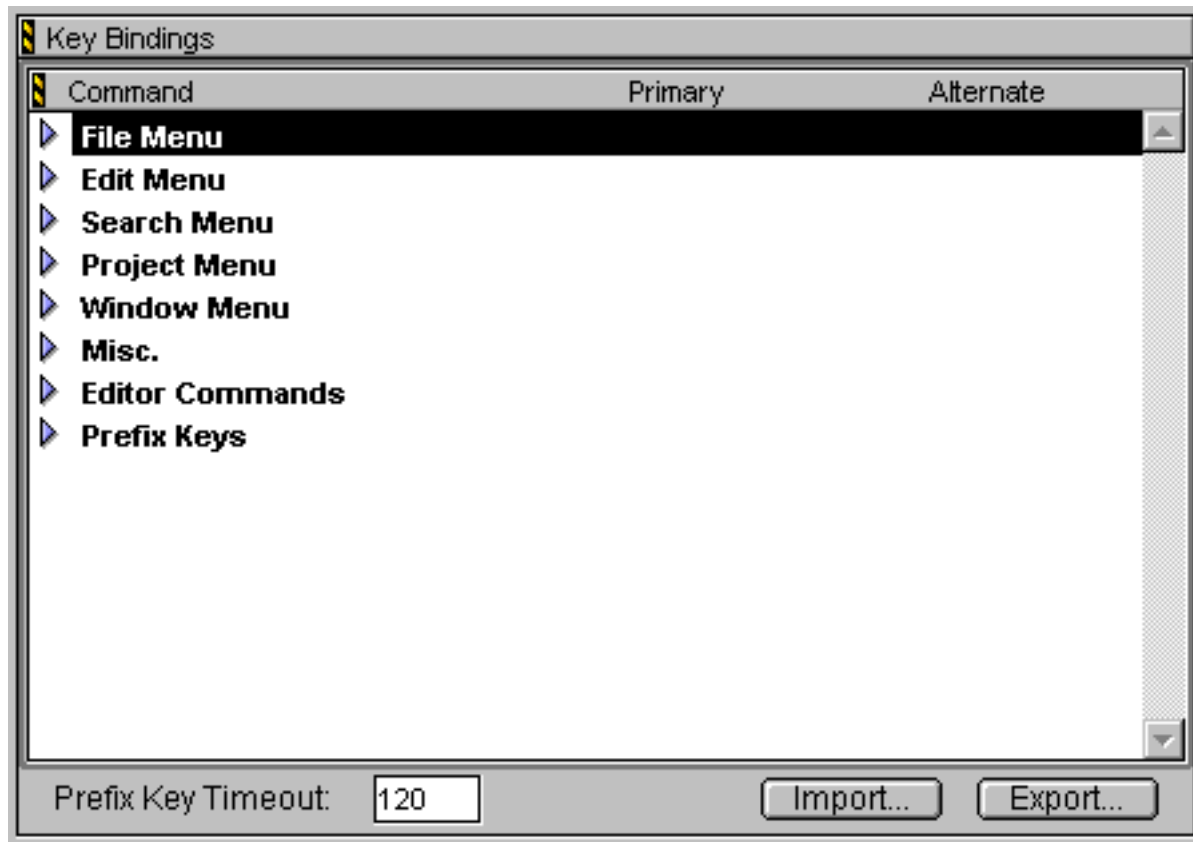
When you select the Key Bindings command from the pop-up menu, you see the Key Bindings options panel shown in [Figure 8.12](#).

## Configuring IDE Options

### *Choosing Preferences*

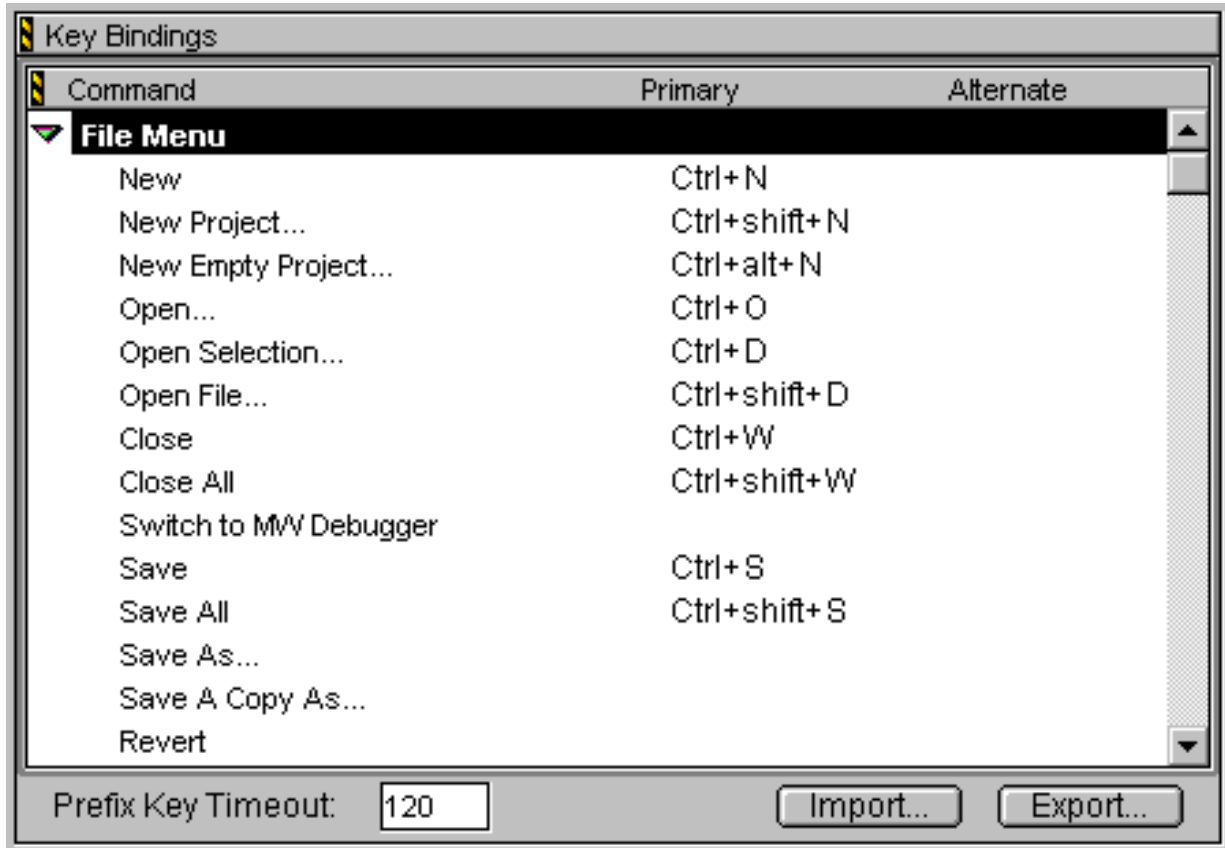
---

**Figure 8.12** Key Bindings Options Panel



To change the key binding that will activate a given IDE operation, you must first select the operation you wish to modify. To do this, click the disclosure triangle next to the Command category to show the individual operations. For example, clicking on the disclosure triangle next to File Menu changes the view of [Figure 8.12](#) to look like the view of [Figure 8.13](#).

**Figure 8.13** File Menu Key Bindings



If you wanted to change the key binding for the New command, double-click the line that contains the command in the window. The dialog box shown in [Figure 8.14](#) appears. Once you see this dialog, hold down the key combination that you want to be used for the command. For example, if you wanted to make the key binding for this command Control-8, you would press the Control key and the 8 key at the same time.

You may also select an additional key binding that can be used for the command, so that each command can have 2 different key bindings for it. To create an additional key binding, click on the Alternate rectangle so that it is selected. Then, hold down the keys that you want to be used for the key binding.

## Configuring IDE Options

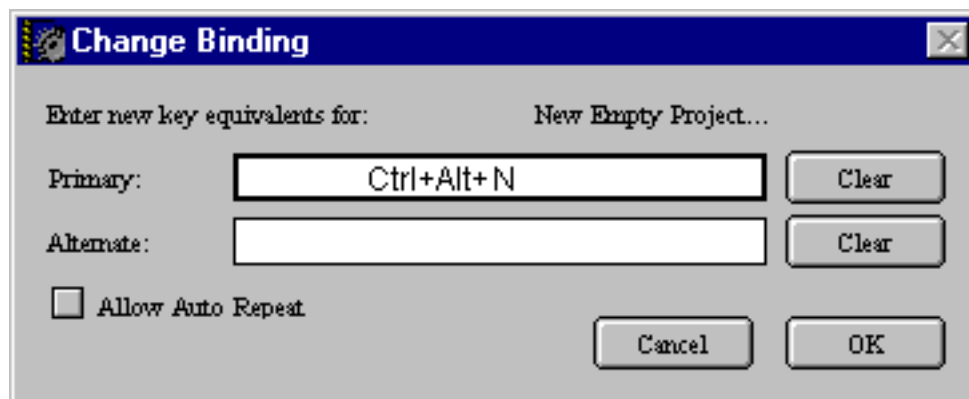
### Choosing Preferences

---

If you wish to clear the Primary or Alternate key bindings, click the Clear button to the right of the appropriate field.

The Allow Auto Repeat option is used to make a key binding repeat if you continue to depress the key combination. A good key binding to use here as an example is the [Find Next](#) command. If you click Allow Auto Repeat for the [Find Next](#) key binding, then you can just hold down the key combination while you watch the search engine find all the text matching your search criteria in the currently-open file. This is a quick way to jump through a file, finding all the patterns you are looking for and showing them quickly. If you did not configure this feature to Allow Auto Repeat, then you would have to release the keys and depress them again every time.

**Figure 8.14**    **Modify Key Binding Dialog Box**



When you are finished changing key bindings, click the OK button to dismiss the dialog box, saving your changes. If you do not wish to save your changes, click the Cancel button.

### Setting the Prefix Key Timeout

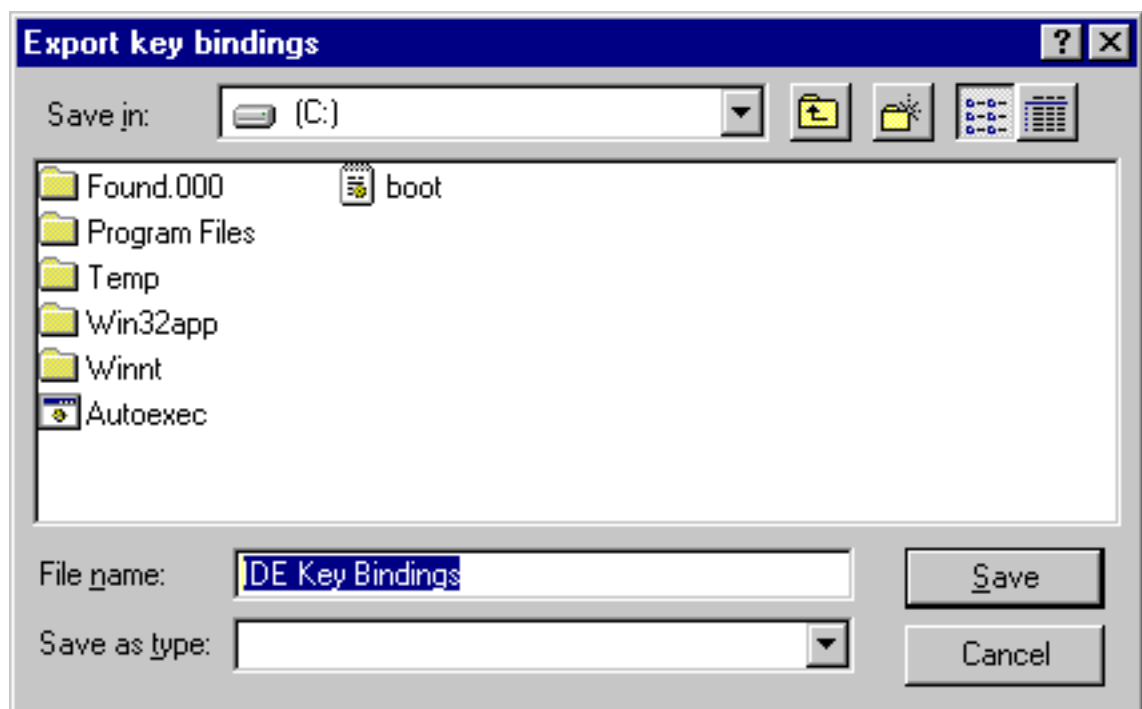
The Prefix Key Timeout field sets how long the IDE waits for the second key after a prefix key is depressed. Larger values mean that the IDE will wait longer for the second key to be depressed.

This value is in “ticks” which are the same as 1 / 60th of a second (16.67 milliseconds). Legal values are in the range of 1 to 999. The default is 120.

### Exporting Key Bindings

If you wish to save your key bindings in a file so that you can later import them into the IDE at another time, you use the Export button. When you click this button, the dialog box shown in [Figure 8.15](#) appears. Navigate to the place on your hard disk that you would like to save the key bindings file to, and click Save.

**Figure 8.15** Key Bindings Export Dialog Box



### Importing Key Bindings

If you wish to import saved key bindings from a previously-exported file, you use the Import button.

This section of the manual is under construction. This feature was not available at the time this manual was written. Check the release notes for the latest information.

## Configuring IDE Options

### Choosing Preferences

---

#### Configuring Misc Bindings

[Table 8.2](#) shows your options for configuring the Misc Key Bindings.

This section describes how to configure the key bindings that are associated with miscellaneous CodeWarrior IDE features. Refer to [Table 8.2](#) to see what the default command key bindings are for the feature you are interested in.

**Table 8.2** Misc Key Bindings

Command	Default Key Binding	Description
Go to header/source file	Control-Tab	This command switches between a source and header file of the same name. To learn more about this feature, refer to <a href="#">“Opening a Related File” on page 147</a> .
Toggle balloon help	Shift-Control-/	This command toggles balloon help on and off.
Go to previous error message	Alt-Control-Up Arrow	This command moves to the previous error message in the Message window. To learn more about stepping through messages in the Message window, refer to <a href="#">“Stepping Through Messages” on page 297</a> .
Go to next error message	Alt-Control-Down Arrow	This command moves to the next error message in the Message window. To learn more about stepping through messages in the Message window, refer to <a href="#">“Stepping Through Messages” on page 297</a> .

### Configuring Editor Commands Bindings

This section describes how to configure the key bindings that are associated with CodeWarrior IDE Editor features. Refer to [Table 8.3](#) to see what the default command key bindings are for the feature you are interested in.

To learn more about text navigation features of the Editor, refer to [“Navigating in Text” on page 143](#).

**Table 8.3 Editor Command Key Bindings**

Command	Default Key Binding	Description
Move character left	Left Arrow	Pressing this key moves the text insertion point one character to the left.
Move character right	Right Arrow	Pressing this key moves the text insertion point one character to the right.
Move word left	Alt-Left Arrow	Pressing these keys moves the text insertion point one word to the left.
Move word right	Alt-Right Arrow	Pressing these keys moves the text insertion point one word to the right.
Move sub-word left	Control-Left Arrow	Pressing these keys moves the text insertion point to the left of the next capital letter or start of a word on the left.
Move sub-word right	Control-Right Arrow	Pressing these keys moves the text insertion point to the left of the next capital letter or start of a word on the right.

## Configuring IDE Options

### *Choosing Preferences*

---

Command	Default Key Binding	Description
Move to start of line	Control-Left Arrow	Pressing these keys moves the text insertion point to the start of the current line.
Move to end of line	Control-Right Arrow	Pressing these keys moves the text insertion point to the end of the current line.
Move line up	Up Arrow	Pressing this key moves the text insertion point one line up.
Move line down	Down Arrow	Pressing this key moves the text insertion point one line down.
Move to top of page	Alt-Up Arrow	Pressing these keys moves the text insertion point to the top of the current view.
Move to bottom of page	Alt-Down Arrow	Pressing these keys moves the text insertion point to the bottom of the current view.
Move to top of file	Control-Up Arrow	Pressing these keys moves the text insertion point to the top of the current file.
Move to bottom of file	Control-Down Arrow	Pressing these keys moves the text insertion point to the bottom of the current file.
Delete character left	Backspace	Pressing this key erases the last character typed.
Delete character right	Del (Delete key)	Pressing this key deletes the character immediately to the right to the text insertion point.



<b>Command</b>	<b>Default Key Binding</b>	<b>Description</b>
Delete to end of file	Control-Backspace	Pressing these keys deletes the characters from the text insertion point to the end of the file.
Character select left	Shift-Left Arrow	Pressing these keys adds the character to the left of the text insertion point to the current selection.
Character select right	Shift-Right Arrow	Pressing these keys adds the character to the right of the text insertion point to the current selection.
Select word left	Alt-Shift-Left Arrow	Pressing these keys selects the full word to the left of the text insertion point.
Select word right	Alt-Shift-Right Arrow	Pressing these keys selects the full word to the right of the text insertion point.
Select sub-word left	Control-Shift-Left Arrow	Pressing these keys selects the partial word to the left of the text insertion point, up to the first capital letter or white space, whichever comes first.
Select sub-word right	Control-Shift-Right Arrow	Pressing these keys selects the partial word to the right of the text insertion point, up to the first capital letter or white space, whichever comes first.

## Configuring IDE Options

### *Choosing Preferences*

---

Command	Default Key Binding	Description
Select line up	Shift-Up Arrow	Pressing these keys selects all text from the current location of the text insertion point to a position one line directly above.
Select line down	Shift-Down Arrow	Pressing these keys selects all text from the current location of the text insertion point to a position one line directly below.
Select to start of line	Shift-Control-Left Arrow	Pressing these keys selects all text left of the text insertion point through the beginning of the current line.
Select to end of line	Shift-Control-Right Arrow	Pressing these keys selects all text right of the text insertion point through the end of the current line.
Select to start of page	Alt-Shift-Up Arrow	Pressing these keys selects all text left of the text insertion point up to the top of the current view of the file.
Select to end of page	Alt-Shift-Down Arrow	Pressing these keys selects all text right of the text insertion point down to the bottom of the current view of the file.
Select to start of file	Shift-Control-Up Arrow	Pressing these keys selects all text left of the text insertion point up to the top of the file.

Command	Default Key Binding	Description
Select to end of file	Shift-Control-Down Arrow	Pressing these keys selects all text right of the text insertion point up to the top of the file.
Scroll line up	Control-Up Arrow	Pressing these keys scrolls the view up one line.
Scroll line down	Control-Down Arrow	Pressing these keys scrolls the view down one line.
Scroll page up	PgUp (Page Up Key)	Pressing these keys scrolls the view up one page.
Scroll page down	PgDn (Page Down Key)	Pressing these keys scrolls the view down one page.
Scroll to top of file	Home	Pressing this key scrolls the view up to the top of the file.
Scroll to end of file	End	Pressing this key scrolls the view down to the bottom of the file.

### Configuring Prefix Keys Bindings

Configuring Prefix Keys is an easy process. To learn about what a Prefix Key is and for an example of what you can do with one, refer to [“What is a Prefix Key?” on page 233](#).

To configure a Prefix Key, just follow the procedure for [“Modifying Key Bindings” on page 233](#). The only difference in configuring a Prefix Key is that you will not be permitted to assign an Alternate key combination.

### Configuring the Quote Key Binding

Configuring the Quote Key is an easy process. To learn about what a Quote Key is and for an example of what you can do with one, refer to [“What is a Quote Key?” on page 233](#).

## Configuring IDE Options

### *Choosing Target Settings*

---

To configure the Quote Key, just follow the procedure for [“Modifying Key Bindings” on page 233](#). The only difference in configuring the Quote Key is that you will not be permitted to assign an Alternate key combination.

## Choosing Target Settings

You can change many different settings to configure the CodeWarrior IDE to build your project and target the way you want. This section discusses the changes you can make to [Target Settings](#) specific to your project.

Depending on the type of project you are working with, some of the material in this chapter may not be visible in the dialog boxes. For example, if you are working with an x86 project, you will not see any dialog boxes for Motorola 68K configuration.

The options panels discussed in this section are in several different categories: Editor, Target, Language Settings, Code Generation, and Linker.

### **Editor**

- [Custom Keywords](#)

### **Target**

- [68K Target](#)
- [Access Paths](#)
- [Build Extras](#)
- [Java Project](#)
- [PPC Target](#)
- [File Mappings](#)
- [Target Settings](#)
- [x86 Target](#)

### **Language Settings**

- [C/C++ Compiler](#)

- [C/C++ Warnings](#)
- [Pascal Compiler](#)
- [Pascal Warnings](#)
- [PPCAsm](#)
- [Rez](#)
- [WinRC Resource Compiler](#)

### Code Generation

- [68K Processor](#)
- [68K Disassembler](#)
- [IR Optimizer](#)
- [Java VM](#)
- [PPC Processor](#)
- [PPC Disassembler](#)
- [x86 CodeGen](#)

### Linker

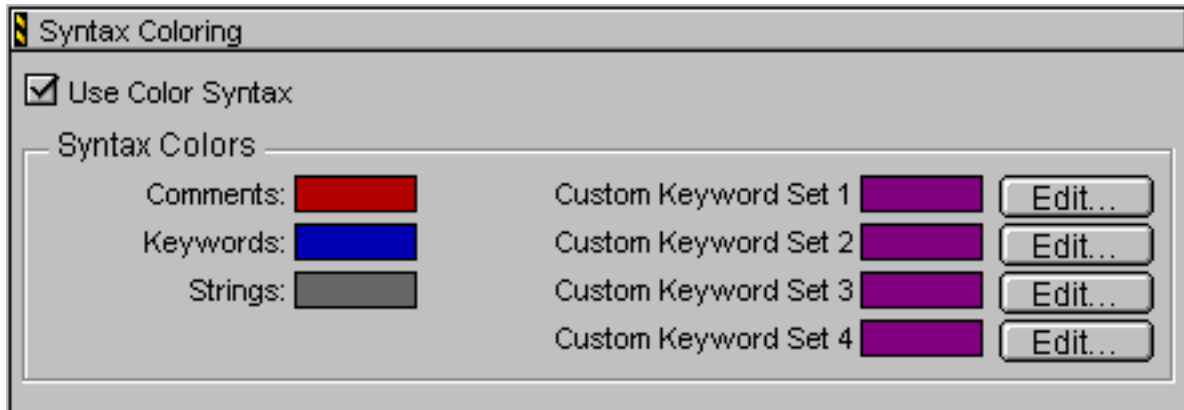
- [68K Linker](#)
- [CFM68K Linker](#)
- [Java Linker](#)
- [PPC Linker](#)
- [PPC PEF](#)
- [x86 Linker](#)

## Custom Keywords

The Custom Keywords options panel, shown in [Figure 8.16](#), 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 the CodeWarrior IDE. See also [“Using color for custom keywords” on page 228](#).

To change a keyword set, click the Edit button and make the appropriate entries in the dialog box that is presented.

Figure 8.16 Custom Keywords Options Panel

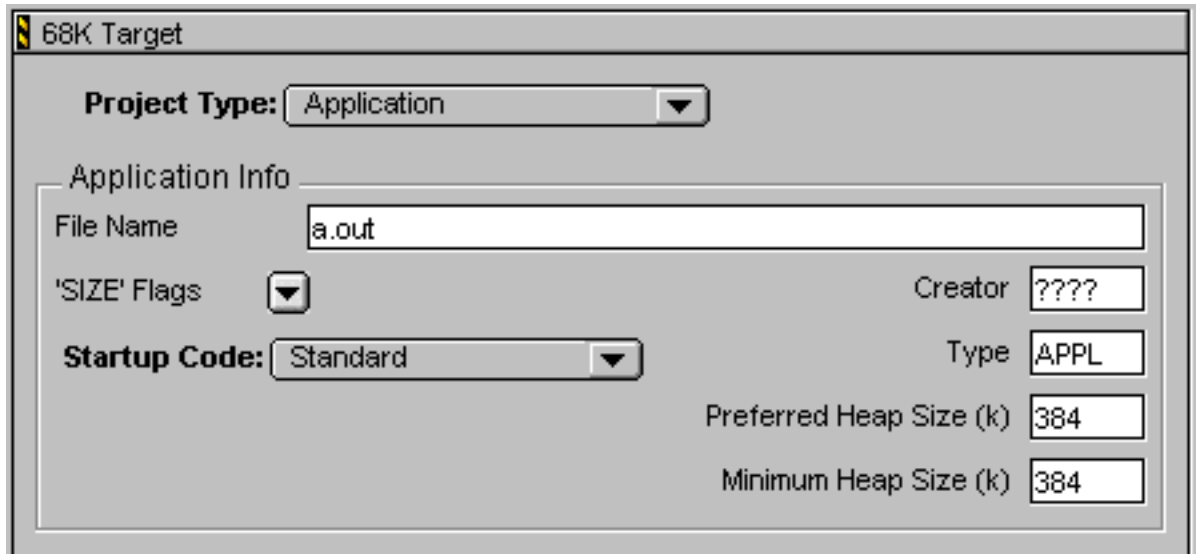


68K Target

The 68K Target options panel, shown in [Figure 8.17](#), allows you to configure settings for Motorola 680x0 target projects. Mac OS projects use these configuration options.

To learn more about Mac OS programming, refer to *Targeting Mac OS* on the CodeWarrior CD.

Figure 8.17 68K Target Options Panel



## Access Paths

If you need to define additional access paths for the CodeWarrior IDE to search while compiling and linking your project, you would use the Access Paths options panel, shown in [Figure 8.18](#).

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, the CodeWarrior IDE performs a recursive search on the path. That is, the CodeWarrior IDE 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 that contains your files to either the System Include Path or User Include Path panes, you will speed compilation of your project.

---

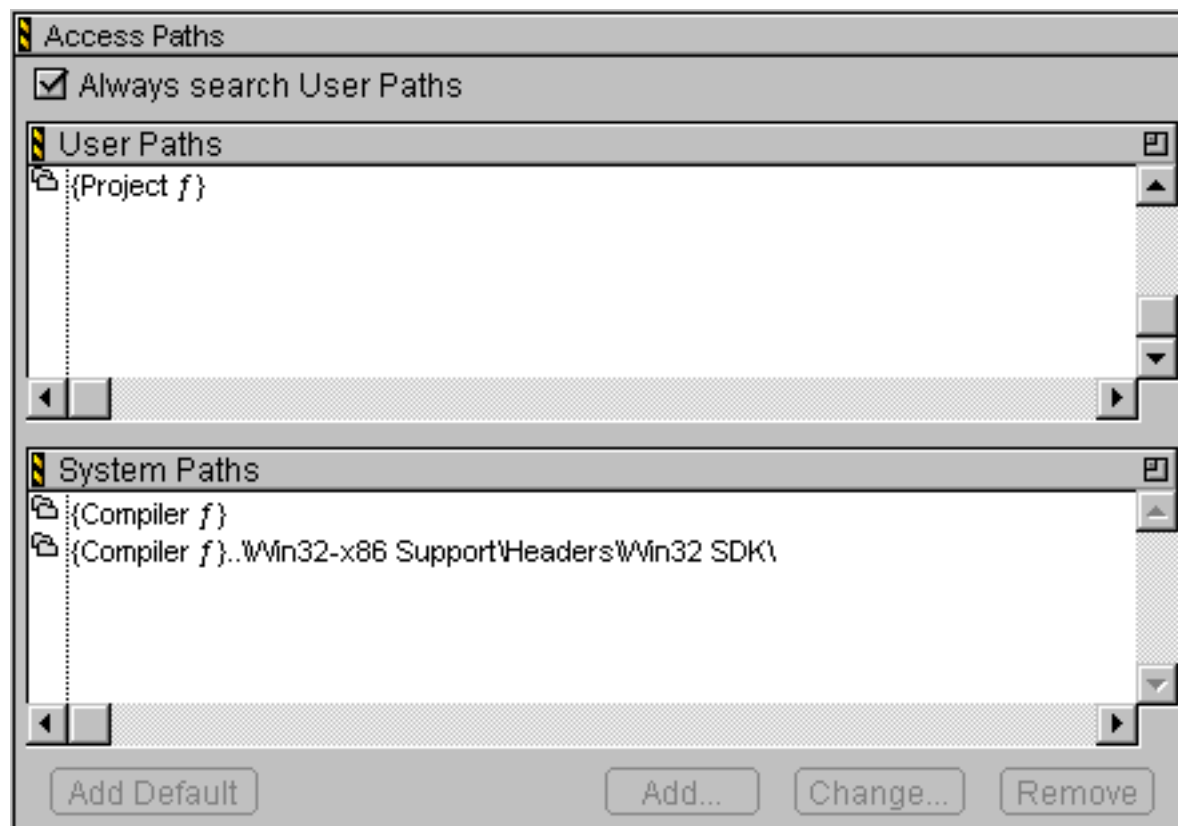
If your project's files or libraries are not in either of the default access paths, the CodeWarrior IDE will not find them when compiling, linking, or running your project. You must add their access path to tell the environment where to look.

## Configuring IDE Options

### Choosing Target Settings

---

**Figure 8.18** Access Paths Options Panel



#### Always Search User Paths

To search for system header or interfaces 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 the CodeWarrior IDE.

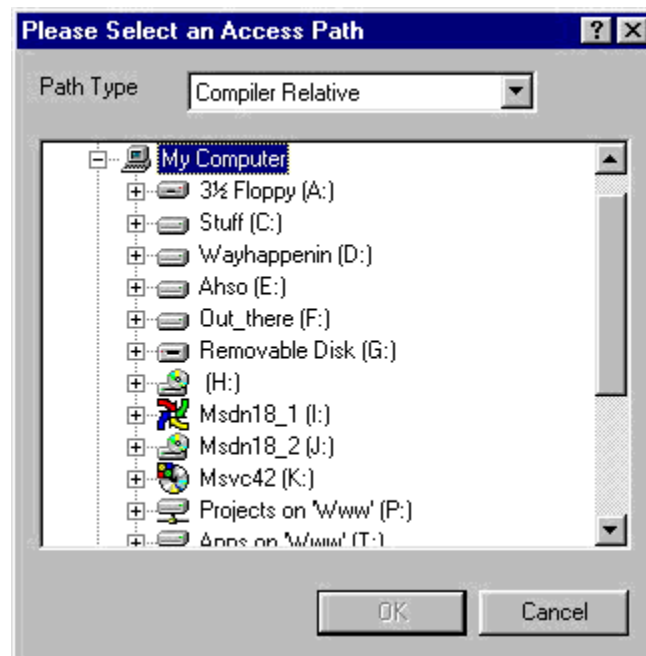
### **Add Default**

The CodeWarrior IDE 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. the CodeWarrior IDE 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 box shown in [Figure 8.19](#) appears.

**Figure 8.19 Access Path Selection Dialog Box**



Use the standard Windows tree controls to navigate to the folder you want to add to the Access Paths.

## Configuring IDE Options

### *Choosing Target Settings*

---

You can choose how CodeWarrior will store the access path with the pop-up menu at the top of the dialog. The four commands in the menu 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 the CodeWarrior IDE, 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 the CodeWarrior IDE 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 box shown in [Figure 8.19 on page 249](#) appears.

Use this dialog box to navigate to the location of the access path you want to change to.

To learn more information about the options in the dialog box, refer to [“Add” on page 249](#).

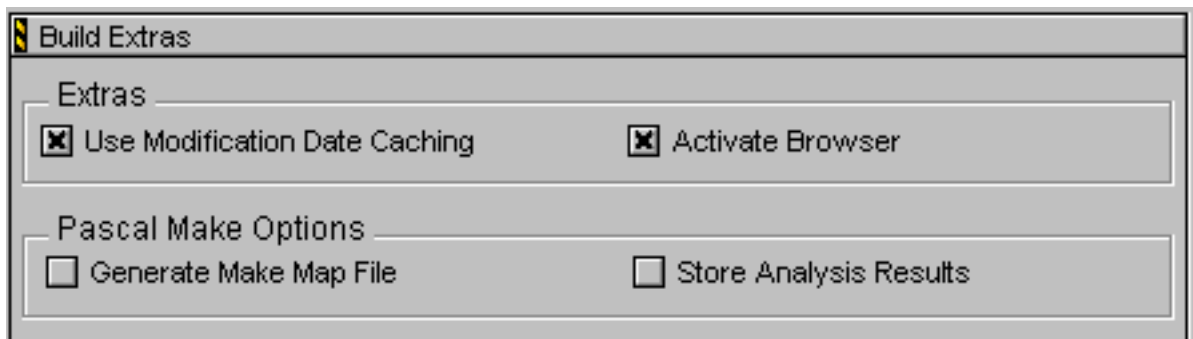
### **Remove**

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 8.20](#).

**Figure 8.20 Build Extras Options Panel**



### **Use Modification Date Caching**

Before making a project, the CodeWarrior IDE checks the modification dates of the files to see if you’ve changed them outside the CodeWarrior IDE. If you edit files with the CodeWarrior Editor only, turn on the Use Modification Date Caching option to shorten compilation time.

## Configuring IDE Options

### Choosing Target Settings

---

#### Activate Browser

If you activate the Browser for a project, you must rebuild your project (see [“Making a Project” on page 276](#)) so the compiler can generate the necessary information for each file.

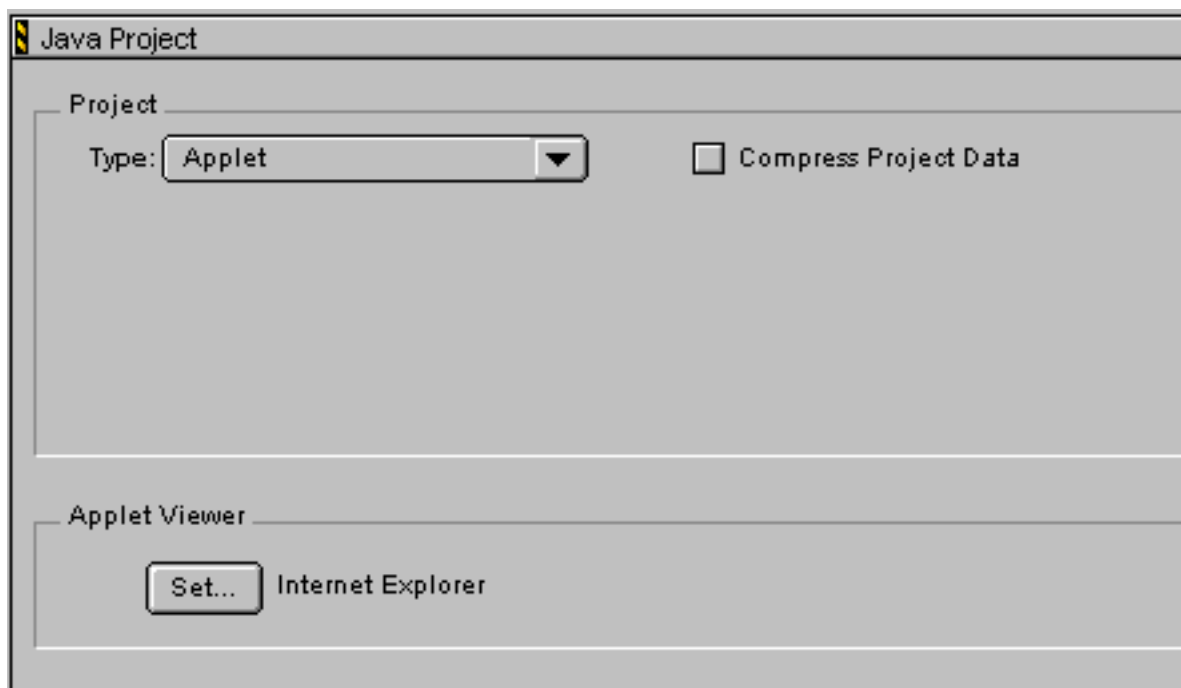
For more information on browser settings and options, see [“Browser Overview” on page 183](#).

#### Java Project

The Java Project options panel, shown in [Figure 8.21](#), allows you to configure settings for Java target projects.

To learn more about configuring the options in this panel for Java projects, refer to *Targeting Java* on the CodeWarrior CD.

**Figure 8.21** Java Project Options Panel



## PPC Target

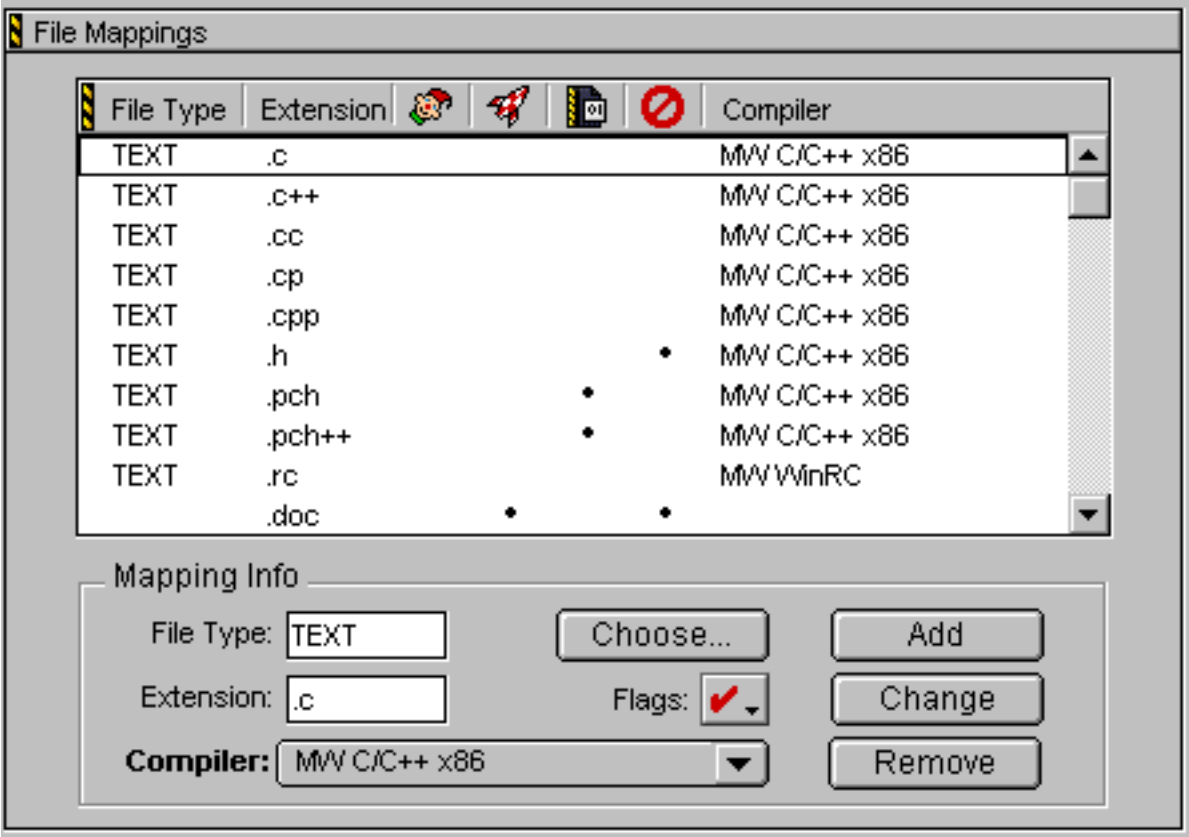
The PPC Project options panel allows you to configure settings for PowerPC (PPC) target projects. Mac OS and BeOS projects use these configuration options.

To learn more about Mac OS and BeOS programming, refer to *Targeting Mac OS* and *Targeting BeOS* on the CodeWarrior CD.

## File Mappings

The File Mappings options panel, shown in [Figure 8.22](#), is used to associate a file name extension such as .c or .p with a plugin compiler. This tells the CodeWarrior IDE which compiler to use when a file with a certain name is encountered.

Figure 8.22 File Mappings Options Panel



**File Mappings List**

The File Mappings List contains a File Type, associated Extension, and compiler choice for each file name extension in the list. This list tells the CodeWarrior IDE 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 [Precompiled](#) fields, and click the Add button. You can choose values for the Flags before clicking the Change 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 flag allows you to enter a file name extension such as `.c` or `.h` for a File Type you are working with in the File Mappings List.

#### **Compiler**

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

#### **Precompiled**

This flag means to compile these files before other files. This is useful if these files create documents that other source files or compilers use. For example, this option lets you create a compiler that translates a file into a C source code file and then compile the C file. YACC (Yet Another Compiler Compiler) files are treated as precompiled files since YACC generates C source code to be compiled by a C compiler.

#### **Launchable**

This flag means open the source code file with the application that created it when you double-click it in a project window.

#### **Resource File**

This flag means include the resources from these files in your finished product.

#### **Ignored by Make**

This flag means 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.

### **Target Settings**

The CodeWarrior IDE allows you to define multiple targets in your project file, so that you can have the same project file build different variations of your code. For example, you can define a debug target

## Configuring IDE Options

### *Choosing Target Settings*

---

and a shipping target in your project. You can then switch between these targets to build them when you want them. To learn more information about using targets in your projects, refer to [“Creating Complex Projects” on page 79](#).

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.

---

The Target Settings options panel, shown in [Figure 8.23](#), allows you to configure the name of your target, as well as which linker and post linker plugins to use for the target. To learn more about which plugins to select for your linker and post linker, refer to the targeting manual for your selected target code, as defined in [Table 1.2 on page 25](#).

### **Linker and Post Linker**

Some targets have post linkers that perform additional work (such as a data format conversion) on the final executable. Refer to information in the individual Targeting manual for your target.

### **Target Name**

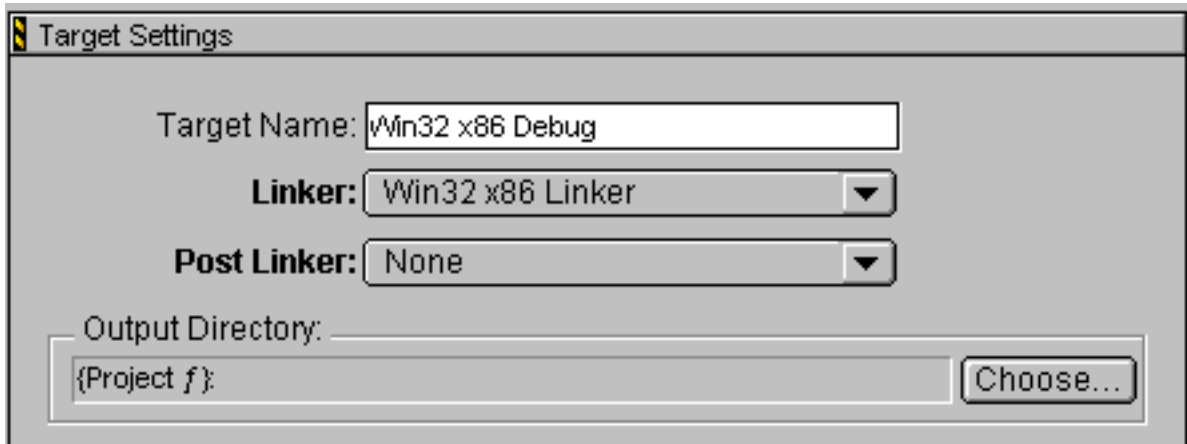
If you'd like to change the name of a target, change the text in the Target Name editable text field of this options panel. When you use the Targets category tab in the project window, you will be able to view the name that you have set.

### **Output Directory**

Click on the Choose button and select a directory where your final linked output file should be written to.



**Figure 8.23** Target Settings Options Panel



## x86 Target

The x86 Target options panel, shown in [Figure 8.24](#), allows you to configure your project binary options depending on whether you are creating an application, library, or shared library (DLL). This panel is used for Intel x86-architecture processor targets.

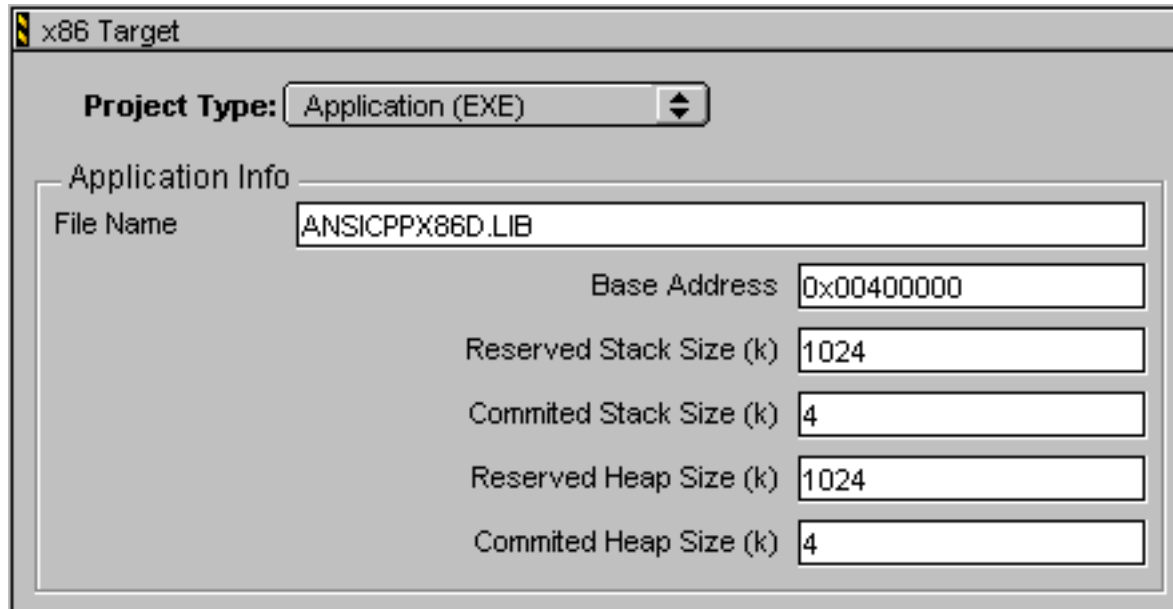
For a detailed explanation of how to configure these options, refer to the *Targeting Win32* manual on the CodeWarrior CD.

## Configuring IDE Options

### Choosing Target Settings

---

**Figure 8.24** x86 Project Options Panel



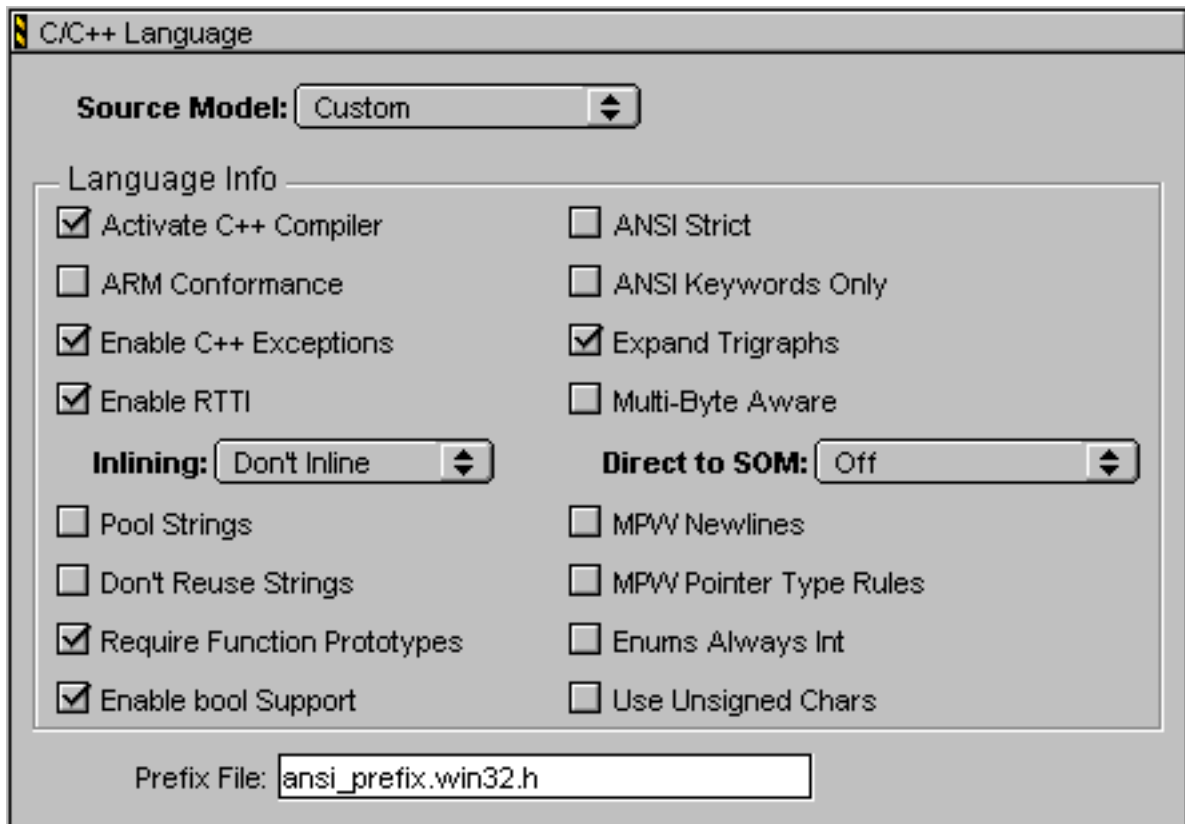
The screenshot shows the 'x86 Target' options panel. At the top, the 'Project Type' is set to 'Application (EXE)'. Below this, the 'Application Info' section contains the following settings:

Field	Value
File Name	ANSICPPX86D.LIB
Base Address	0x00400000
Reserved Stack Size (k)	1024
Committed Stack Size (k)	4
Reserved Heap Size (k)	1024
Committed Heap Size (k)	4

## C/C++ Compiler

There are several settings you can configure to tailor the C/C++ compilation of your project's source code files, as shown in [Figure 8.25](#). You may refer to the *C Compiler Guide* documentation for detailed information on these options.

Figure 8.25 C/C++ Compiler Options Panel



## C/C++ Warnings

The C/C++ Warnings options panel, shown in [Figure 8.26](#), allows you to configure the warning messages that are emitted during compilation of your C and C++ code.

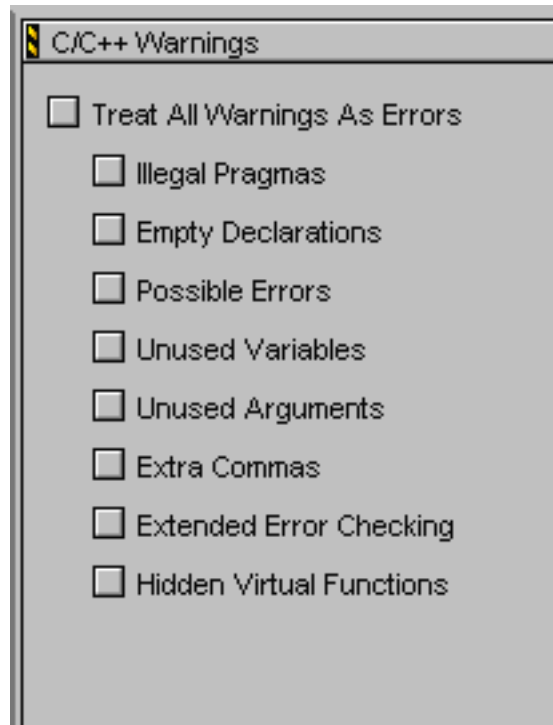
To learn more information about how to configure these options, refer to the *C Compiler Guide* documentation.

## Configuring IDE Options

### Choosing Target Settings

---

**Figure 8.26** C/C++ Warnings Options Panel



## Pascal Compiler

The Pascal Compiler options panel 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.

## Pascal Warnings

The Pascal Warnings options panel 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.

## PPCAsm

The PPCAsm options panel allows you to configure options for using PowerPC inline assembly code in your CodeWarrior IDE source files.

To learn more about using the options in this panel, refer to the book *Targeting Mac OS* on the CodeWarrior CD. To learn more about writing PowerPC assembly code, refer to the MPW documentation on the CodeWarrior CD, and the *C Compiler Guide*.

## Rez

The Rez options panel allows you to configure options for the MPW Rez resource compiler.

To learn more about setting these options for the Mac OS, refer to the *Targeting Mac OS* documentation on the CodeWarrior CD.

## WinRC Resource Compiler

The WinRC Resource Compiler options panel, shown in [Figure 8.27](#), allows you to configure options for the Win32 resource compiler.

For a detailed explanation of how to configure these options for your project, refer to the *Targeting Win32* manual on the CodeWarrior CD.

**Figure 8.27 WinRC Resource Compiler Options Panel**



## 68K Processor

The 68K Processor options panel, shown in [Figure 8.28](#), allows you to configure the options for Motorola 68K targets, such as Mac OS.

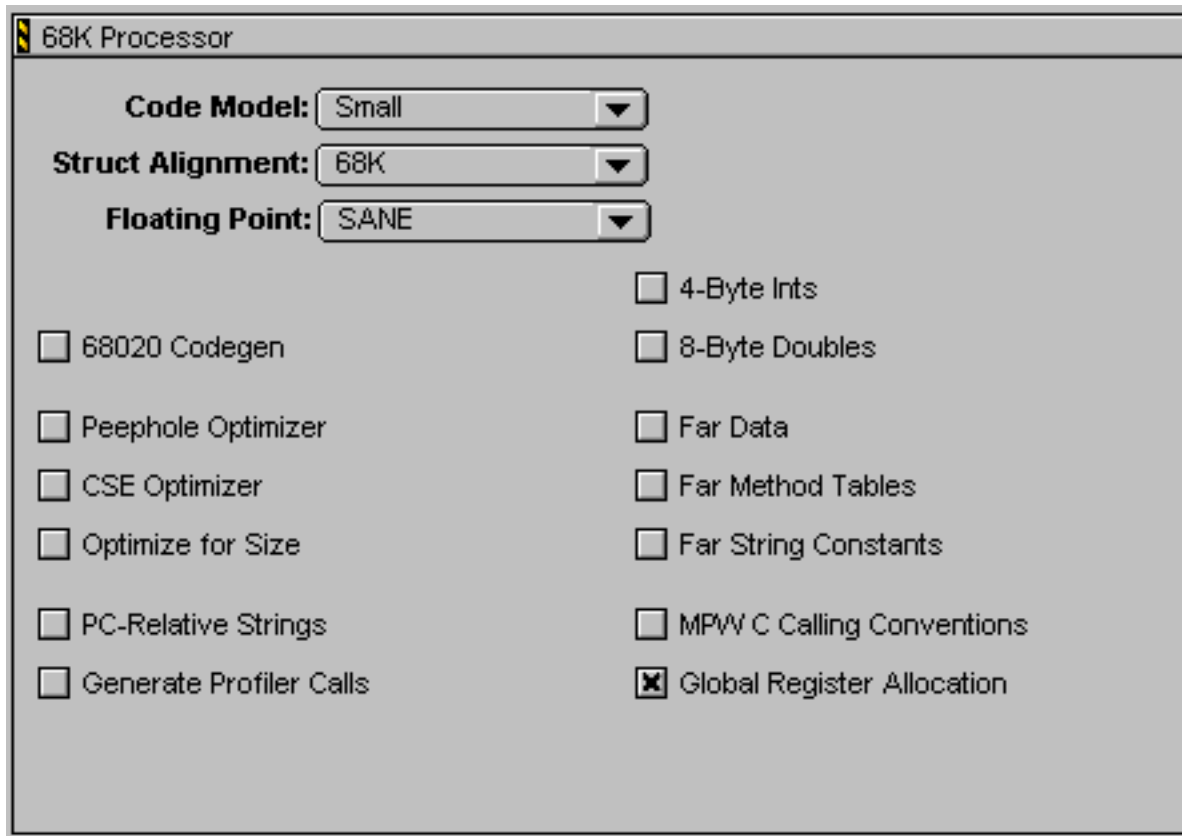
## Configuring IDE Options

### Choosing Target Settings

---

To learn more about setting these options for the Mac OS, refer to the *Targeting Mac OS* documentation on the CodeWarrior CD.

**Figure 8.28 68K Processor Options Panel**

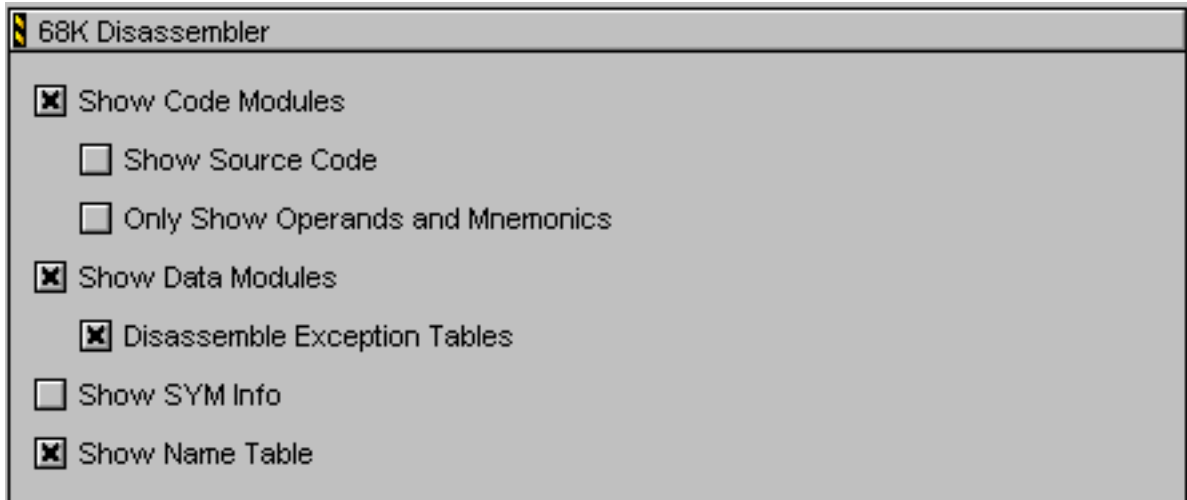


## 68K Disassembler

The 68K Disassembler options panel, shown in [Figure 8.29](#), allows you to configure options for disassembly of Motorola 68K code.

For more information about configuring the options in this panel, refer to the *Targeting Mac OS* manual on the CodeWarrior CD.

**Figure 8.29 68K Disassembly Options Panel**

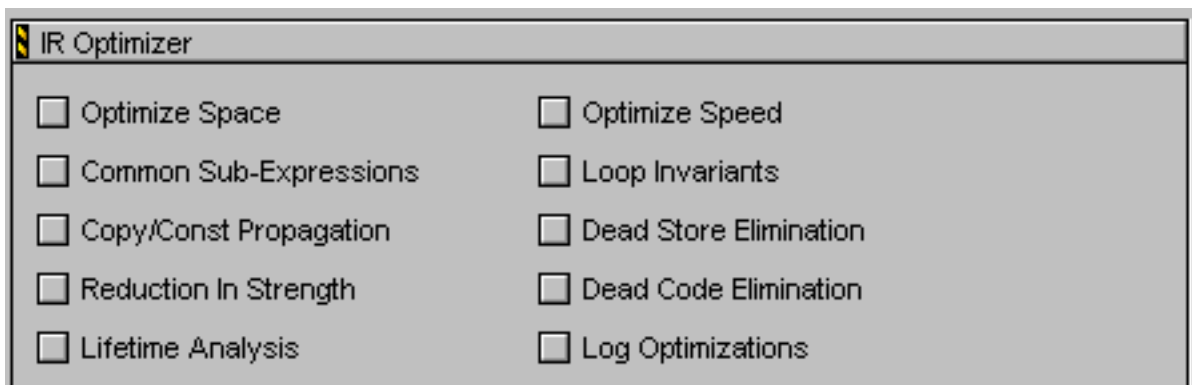


## IR Optimizer

The IR Optimizer options panel, shown in [Figure 8.30](#), 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 8.30 IR Optimizer Options Panel**



## Configuring IDE Options

### Choosing Target Settings

---

#### Java VM

The Java VM options panel allows you to configure code generation settings for Java target projects.

To learn more about configuring the options in this panel for Java projects, refer to *Targeting Java* on the CodeWarrior CD.

#### PPC Processor

The PPC Processor options panel allows you to configure the options for PowerPC targets, such as Mac OS, PowerTV, and BeOS.

To learn more about settings these options for the Mac OS, refer to the *Targeting Mac OS* documentation on the CodeWarrior CD. Refer to *Targeting PowerTV* to learn more about creating PowerTV code.

#### PPC Disassembler

The PPC Disassembler options panel allows you to configure options for disassembly of PowerPC code.

For more information about configuring the options in this panel, refer to the *Targeting Mac OS* book on the CodeWarrior CD.

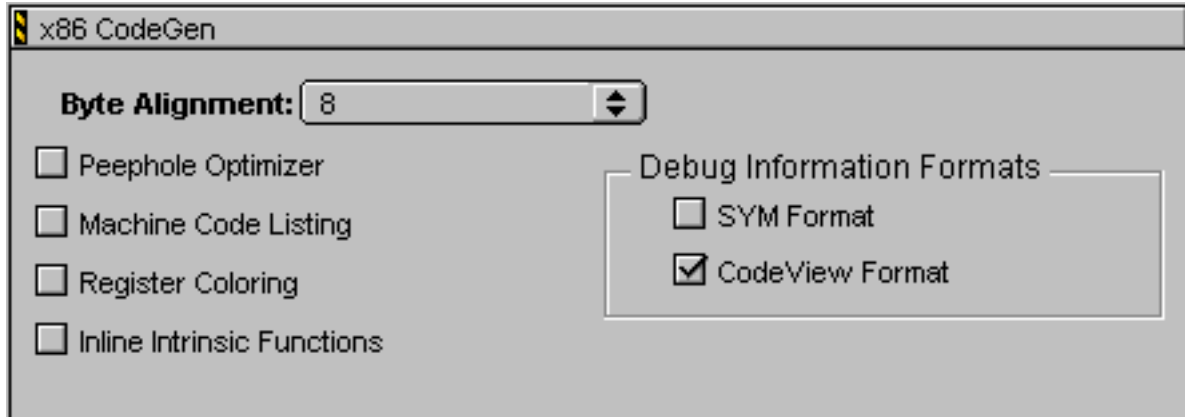
#### x86 CodeGen

The x86 CodeGen options panel, shown in [Figure 8.31](#), allows you to configure the Intel x86-compatible 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 8.31** x86 CodeGen Options Panel



## 68K Linker

The 68K Linker options panel, shown in [Figure 8.32](#), allows you to configure the Motorola 68K linker options for your project.

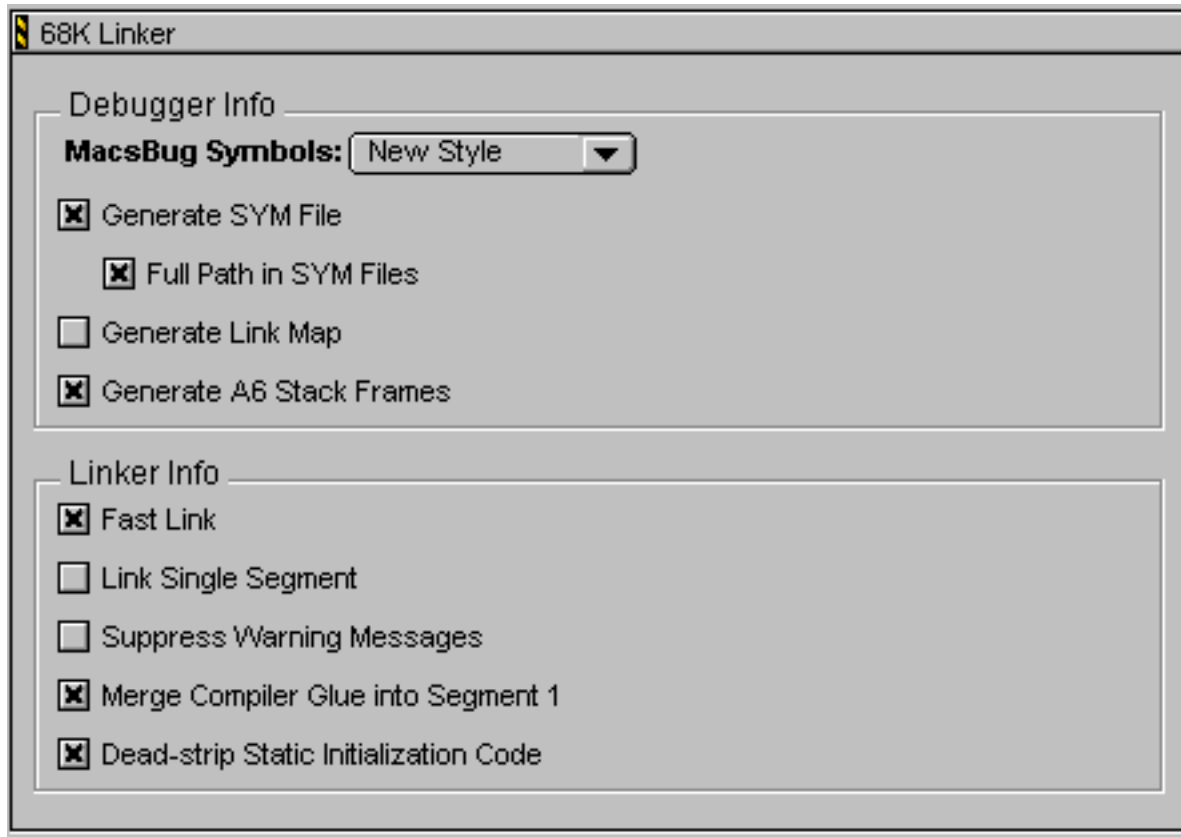
To learn more about setting these options for the Mac OS, refer to the *Targeting Mac OS* documentation on the CodeWarrior CD.

## Configuring IDE Options

### Choosing Target Settings

---

**Figure 8.32** 68K Linker Options Panel

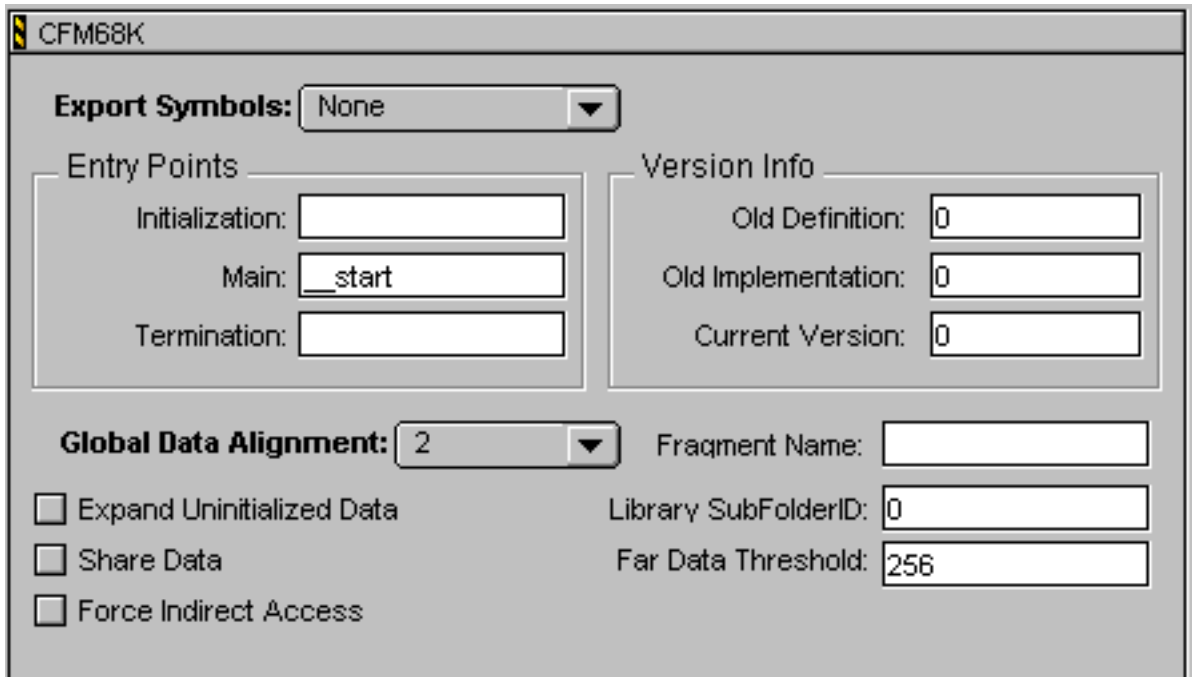


## CFM68K Linker

The CFM68K linker options panel, shown in [Figure 8.33](#), allows you to configure the Motorola 68K linker options for your Mac OS project.

To learn more about setting these options for the Mac OS, refer to the *Targeting Mac OS* documentation on the CodeWarrior CD.

**Figure 8.33 CFM68K Linker Options Panel**



## Java Linker

The Java Linker options panel allows you to configure linker settings for Java target projects.

To learn more about configuring the options in this panel for Java projects, refer to *Targeting Java* on the CodeWarrior CD.

## PPC Linker

The PPC Linker options panel allows you to configure the PowerPC linker options for your project.

To learn more about setting these options for the Mac OS, refer to the *Targeting Mac OS* documentation on the CodeWarrior CD. Refer to *Targeting Power TV* to learn more about creating code for the Power TV.

## Configuring IDE Options

### Choosing Target Settings

---

## PPC PEF

The PPC PEF options panel allows you to configure the PowerPC PEF, or Program Executable Format, options for your project.

To learn more about setting these options for the Mac OS, refer to the *Targeting Mac OS* documentation on the CodeWarrior CD.

## x86 Linker

The x86 Linker options panel, shown in [Figure 8.34](#), 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.

**Figure 8.34** x86 Linker Options Panel

The screenshot shows the 'x86 Linker' options panel. It is organized into several sections:

- Entry Point:** Includes a dropdown for 'Entry Point Usage' (set to 'Default') and a text field for 'Name'.
- SubSystem & IDs:** Includes a dropdown for 'SubSystem' (set to 'Windows GUI'), a text field for 'SubSystem ID' (set to '4.00'), and a text field for 'User ID' (set to '0.00').
- Options:** Includes three checkboxes: 'Generate Link Map' (checked), 'Generate SYM File' (unchecked), and 'Generate CV Info' (checked).
- Command File:** A text field for specifying the linker command file.

## Toolbar Customization

---

**NOTE:** As this manual went to press, the toolbar customization features were not fully functional. Check the release notes for the latest information.

---

## Configuring IDE Options

### *Toolbar Customization*

---



# Compiling and Linking

---

This chapter discusses how to control compilation, linking and running a CodeWarrior project.

## Compiling and Linking 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 45](#), [“Working with Files Overview” on page 101](#), [“Source Code Editor Overview” on page 117](#), and [“Configuring IDE Options Overview” on page 215](#).

You should also be familiar with features such as moving files in the Project window, the Project window's columns, and Project window pop-up menus. To learn more about these things, refer to [“Projects Overview” on page 45](#).

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 the CodeWarrior IDE 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 25](#).

The CodeWarrior IDE can only compile and link files belonging to an open project. That is, you should have a project open before trying to compile any files.

The topics in this chapter are:

## Compiling and Linking

### *Choosing a Compiler*

---

- [Choosing a Compiler](#)
- [Compiling and Linking a Project](#)
- [Using Precompiled or Preprocessed Headers](#)
- [Preprocessing Source Code](#)
- [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 associate a file suffix with 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.

Plugin compilers usually have default target settings to help the CodeWarrior IDE decide which project files a plugin handles. During regular compile and link operations, the IDE assigns files to the proper plugin automatically.



## Setting a File Extension

To associate a plugin compiler with a given file, you set the [File Mappings](#) options. For a description of how to configure these options to set a compiler for your source code files, refer to [“File Mappings” on page 253](#).

## 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 . Depending on your project type, a few of these commands may be disabled or renamed. For example, you cannot [Run](#) a shared library, 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.

The CodeWarrior IDE can only compile and link files belonging to an open project. That is, you should have a project open before trying to compile any files.

If you have multiple projects open at the same time, you may want to learn about how to select a default project before compiling and linking. To learn about this topic, refer to [“Choosing a Default Project” on page 58](#).

The topics in this section are:

- [Touching and Untouching Files](#)
- [Compiling Files](#)
- [Updating a Project](#)
- [Making a Project](#)
- [Enabling Debugging](#)
- [Running a Project](#)
- [Debugging a Project](#)
- [Generating a Link Map](#)

- [Synchronizing Modification Dates](#)
- [Removing Objects](#)
- [Advanced Compile Options](#)

## 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 not compile that file the next time the project is built. A newly-created file, or a file that has never been compiled, can't be untouched.

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 76](#).

## 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.

You may want to switch between multiple targets in a project when compiling files. To learn how to do this, refer to [“Setting the Current Build Target” on page 88](#).

The [Compile](#) command on the [Project Menu](#) is dimmed when:

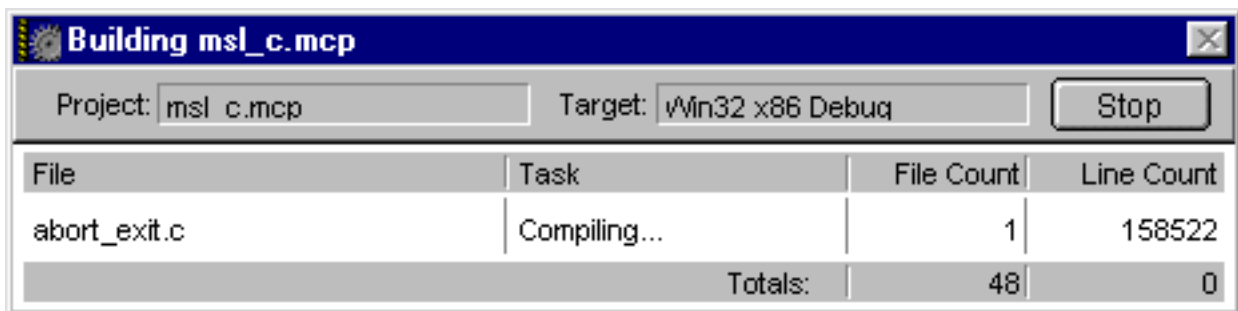
- There is no open project.
- The active Editor window does not have a source code file name extension.
- The active window's source code file is not included in your project.
- The binary file, such as the application you created from the project, is running under the CodeWarrior debugger.

- 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 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 67](#).

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 several 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 67](#).

### 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 274](#).

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.

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 274](#).

You may want to switch between multiple targets in a project when updating a project. To learn how to do this, refer to [“Setting the Current Build Target” on page 88](#).

### 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 some example project types and what is built when the [Make](#) command is executed. To find a full list of the types

of software products you can produce, refer to the targeting guide for your target. Refer to [Table 1.1 on page 19](#) to find out which guide to read for your target.

**Table 9.1 Example Products for Certain Targets**

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.

You may want to switch between multiple targets in a project when making a project. To learn how to do this, refer to [“Setting the Current Build Target” on page 88](#).

## Enabling Debugging

When the [Enable Debugger](#) option is chosen 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 96](#).

To learn more about running your project, refer to [“Running a Project” on page 278](#).

## 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.

If the current project is for a library or shared library (DLL), the [Run](#) command is not available.

When compiling and linking is successful, CodeWarrior saves a new application on your hard disk. It is named according to options you set. If you would like to change these options, refer to [“Choosing Target Settings” on page 244](#).

## 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 277](#).

The second step is for you to start the debugger with your compiled application as the debug target. You can do this by choosing 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 277](#) to learn how to remedy this.

Once you have chosen 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 277](#). 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, or the application is already running.

---

**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 to the debugger while you are in a CodeWarrior Editor window, use the [Switch to MW 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” on page 268](#). 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 79](#).

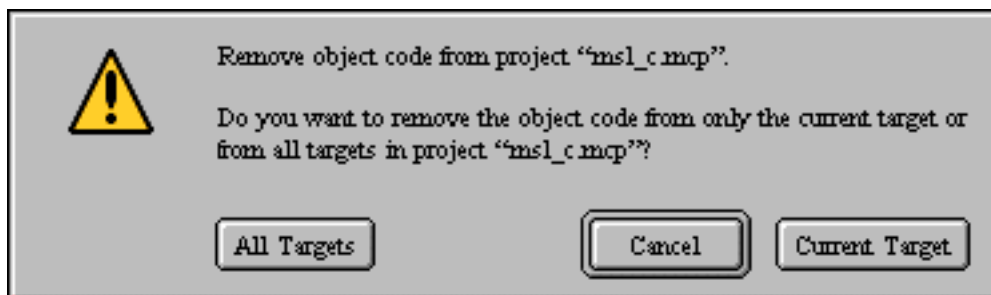
### Removing Objects

When you compile your project, the CodeWarrior IDE adds the object code from each source file to 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 object code, select the [Remove Object Code](#) command from the [Project Menu](#). The dialog box shown in [Figure 9.2](#) appears.

**Figure 9.2** Remove Objects dialog box



Clicking All Targets removes all object code data for all targets in the project, resetting the Code and Data size of each file in the project window to zero. Current Target removes the objects for the current target only, and leaves the objects in place for all other targets. Cancel aborts the operation so none of your object code is removed.

To learn how to change the current target of your project, refer to [“Setting the Current Build Target” on page 88](#).

### Advanced Compile Options

This section describes two options that either speed up your project build times or alert you when a build is completed.



### **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 251](#).

### **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 251](#).

## **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. Many programming languages support precompiled headers, including C and C++.

To shorten the time spent compiling and recompiling a header file, use the [Precompile](#) command 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.

---

## Compiling and Linking

### *Using Precompiled or Preprocessed Headers*

---

---

**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 386 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:

---

```
#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 .mch extension, such as MyHeader.mch, 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 .mch file, without the .mch extension. For example, myHeader.mch would be precompiled and saved as myHeader on your hard disk.

---

**WARNING!** If you don't use the .mch suffix, you may encounter a performance penalty when using precompiled headers.

---

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 . 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 299](#).

## Compiling and Linking

### *Using Precompiled or Preprocessed Headers*

---

#### Automatic updating

During a [Make](#) or [Bring Up To Date](#) operation, the CodeWarrior IDE automatically updates a precompiled header if the source has been modified.

If CodeWarrior encounters a .mch 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 282](#).

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 .mch filename extension.

Now add the source file to the open project with the [Add Window](#) command in the [Project Menu](#).

Whenever the .mch 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” on page 258](#).

---

**NOTE:** Do not use the .mch source file in `#include` directives; use the name of the resulting precompiled header file. Although using the .mch 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](#) 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 [Target Settings](#) from the [Edit Menu](#), and select the [C/C++ Compiler](#) 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 corresponding to the needs of your source code.

## Compiling and Linking

### *Using Precompiled or Preprocessed Headers*

---

Use .mch as a filename extension. For example, save this file as "MyPrecomp.mch".

Choose [Add Window](#) from the [Project Menu](#) to add this precompiled header file to your project.

Choose [Target Settings](#) from the [Edit Menu](#) and select the [C/C++ Compiler](#) settings panel.

In the Prefix File field, enter your precompiled file's name, in this example "MyPrecomp.mch". 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.

### **1. Create a New Source Code File**

Open your project and create a new source code file with the [New](#) command.

This new text file will contain your compiler directives.

### **2. Open the Pascal Compiler settings panel.**

Choose [Target Settings](#) from the [Edit Menu](#), and select the [Pascal Compiler](#) settings panel.

### **3. Get the Prefix File name**

If there is a filename in the Prefix File box, [Copy](#) it into the Clipboard, then click OK.

**4. Paste the Prefix File name**

In the new editor window, paste (from the clipboard) 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}
```

**5. Type in all your own `{SETC}`, `{I}`, and other preprocessor directives.**

This file cannot contain any source code that generates data or executable code.

**6. [Save](#) this file**

Save it as an ordinary Pascal source code file in the same folder as your project.

For example, save this file as “MyPrecomp.pas”.

**7. Open the Pascal Compiler settings panel.**

Choose [Target Settings](#) from the [Edit Menu](#), and select the [Pascal Compiler](#) settings panel.

**8. Set the New Prefix File**

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

The preprocessor prepares source code for the compiler. It interprets directives beginning with the “#” and “\$” symbols (such as `#define`, `$pragma` and `#ifdef`), removes extra spaces and blank lines, and removes comments (such as `/*...*/` and `//`). You might want to preprocess a file if you want to see what the code looks like just before compilation.

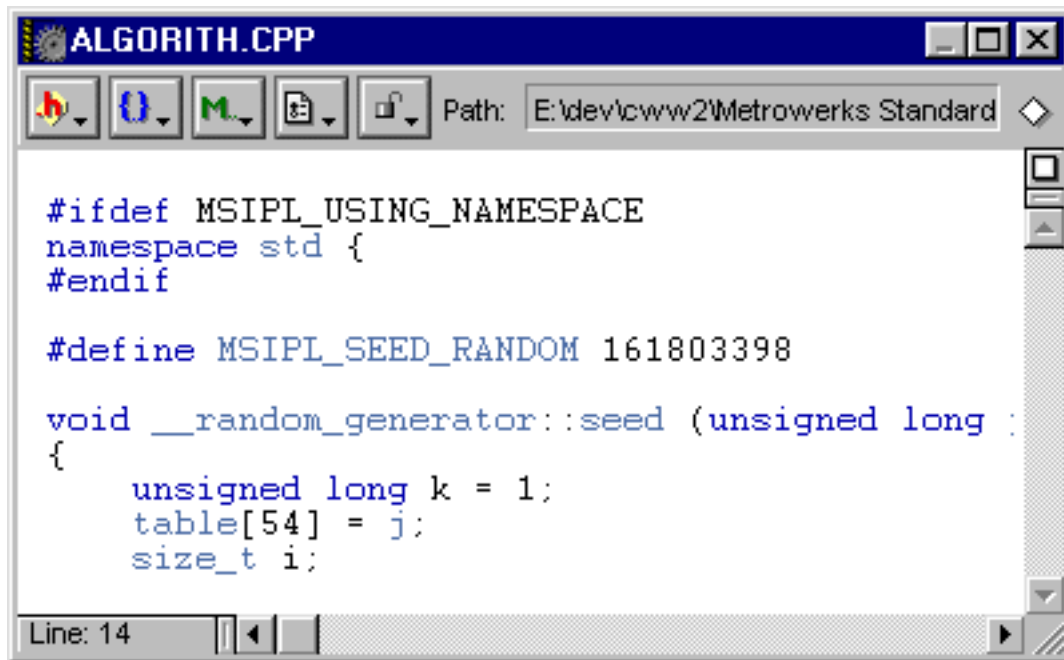
## Compiling and Linking

### Preprocessing Source Code

---

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

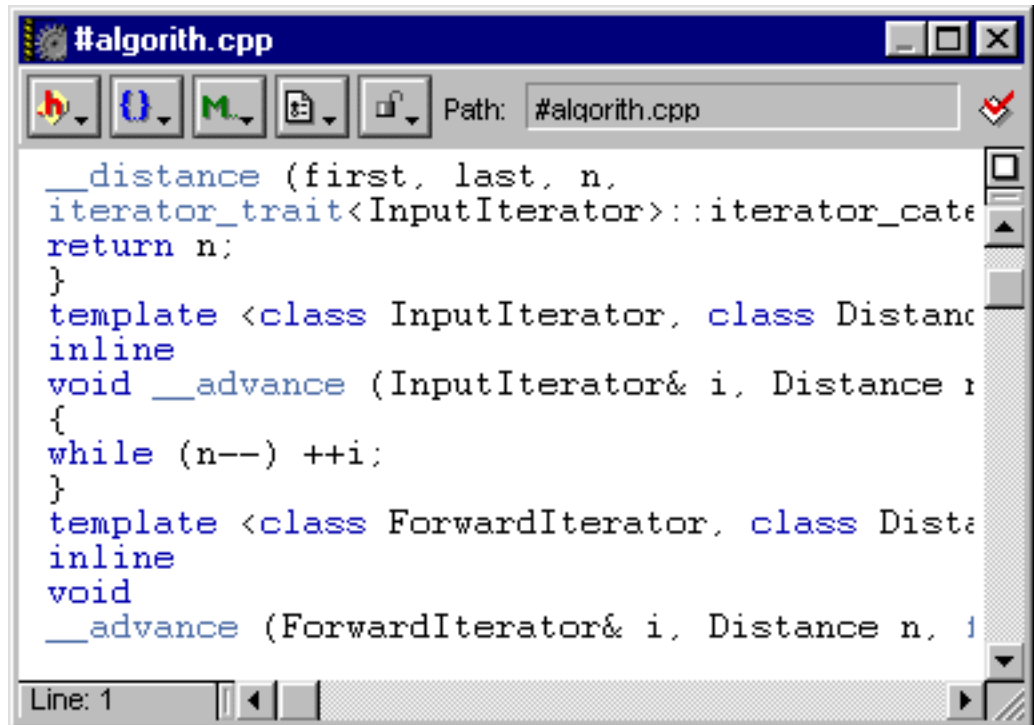


Open a file that you want to preprocess, or select a file in your currently-open Project window. To preprocess a file, select the [Preprocess](#) command on the [Project Menu](#). The results of the [Preprocess](#) command are stored in a new file named after the source code file that was preprocessed 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](#).



Figure 9.4 Preprocessor output



```
#algorithm.cpp

__distance (first, last, n,
iterator_trait<InputIterator>::iterator_cate
return n;
}
template <class InputIterator, class Distanc
inline
void __advance (InputIterator& i, Distance r
{
while (n--) ++i;
}
template <class ForwardIterator, class Dist:
inline
void
__advance (ForwardIterator& i, Distance n, 1
```

Line: 1

## 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 code that is being executed when your source code is executed. In addition, the disassembled code can be a model for writing your own assembly routines. Library files may also be examined using this command.

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](#)).

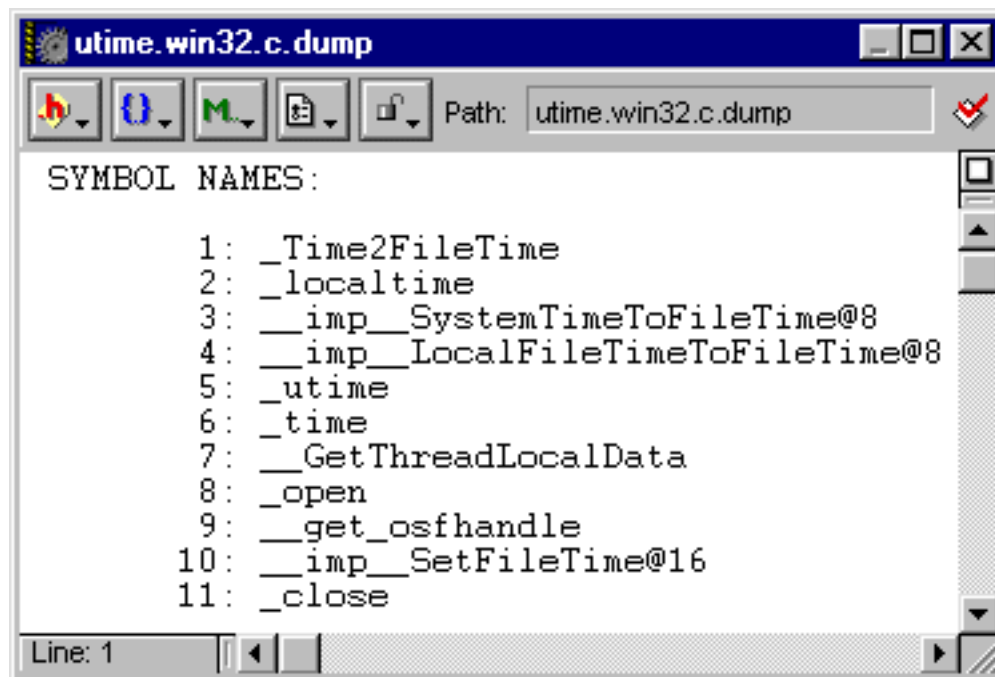
To save the contents of the “.dump” window, choose one of the save commands in the [File Menu](#).

## Compiling and Linking

### *Guided Tour of the Message Window*

---

**Figure 9.5** Disassembling a selected source code file



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

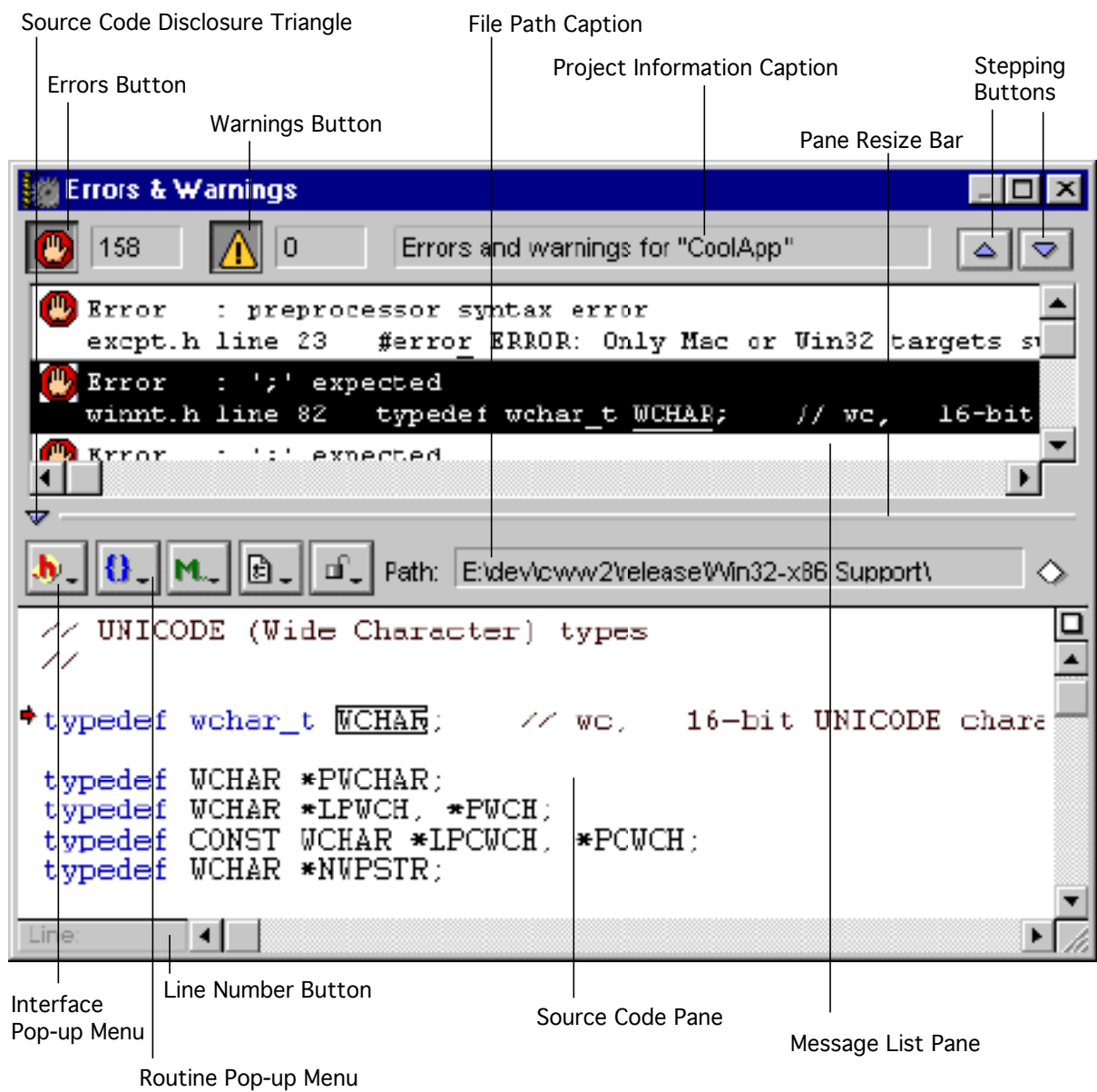
The Message window is used to display messages about events that have occurred when compiling, linking, or searching files. There are a number of elements in the window that are useful for accomplishing certain tasks, such as navigating to error locations and scrolling to see all messages for a project.

There are some user interface items in the Message window that are not discussed here. To learn about what the [Marker Pop-Up Menu](#), [Options Pop-Up Menu](#), [Permissions Pop-up Menu](#), and [File Path Caption](#) do, refer to [“Guided Tour of the Editor Window” on page 117](#).

The topics in this section include:

- [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 296](#).

## Warning Button



The Warning Button in the Message window, shown in [Figure 9.6 on page 292](#), 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 296](#).

## Project Information Caption

The Project Information Caption, shown in [Figure 9.6 on page 292](#), 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 292](#), 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 297](#).

## Message List Pane

The Message List Pane, shown in [Figure 9.6 on page 292](#), allows you to view your messages.

To learn more about seeing messages in the Message window, refer to [“Seeing Errors and Warnings” on page 296](#).

## Source Code Disclosure Triangle

The Source Code Disclosure Triangle, shown in [Figure 9.6 on page 292](#), 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 296](#).

## 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 126](#).

## 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 119](#).

## 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 121](#).

## **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 124.](#)

## **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 125.](#)

# **Using the Message Window**




While compiling your project, the CodeWarrior IDE 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, and 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 the CodeWarrior IDE from creating a binary. However, they indicate potential problems during run time. 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.



- During a Find when the [Batch Checkbox](#) is selected in the Find dialog box.

## Stepping Through Messages

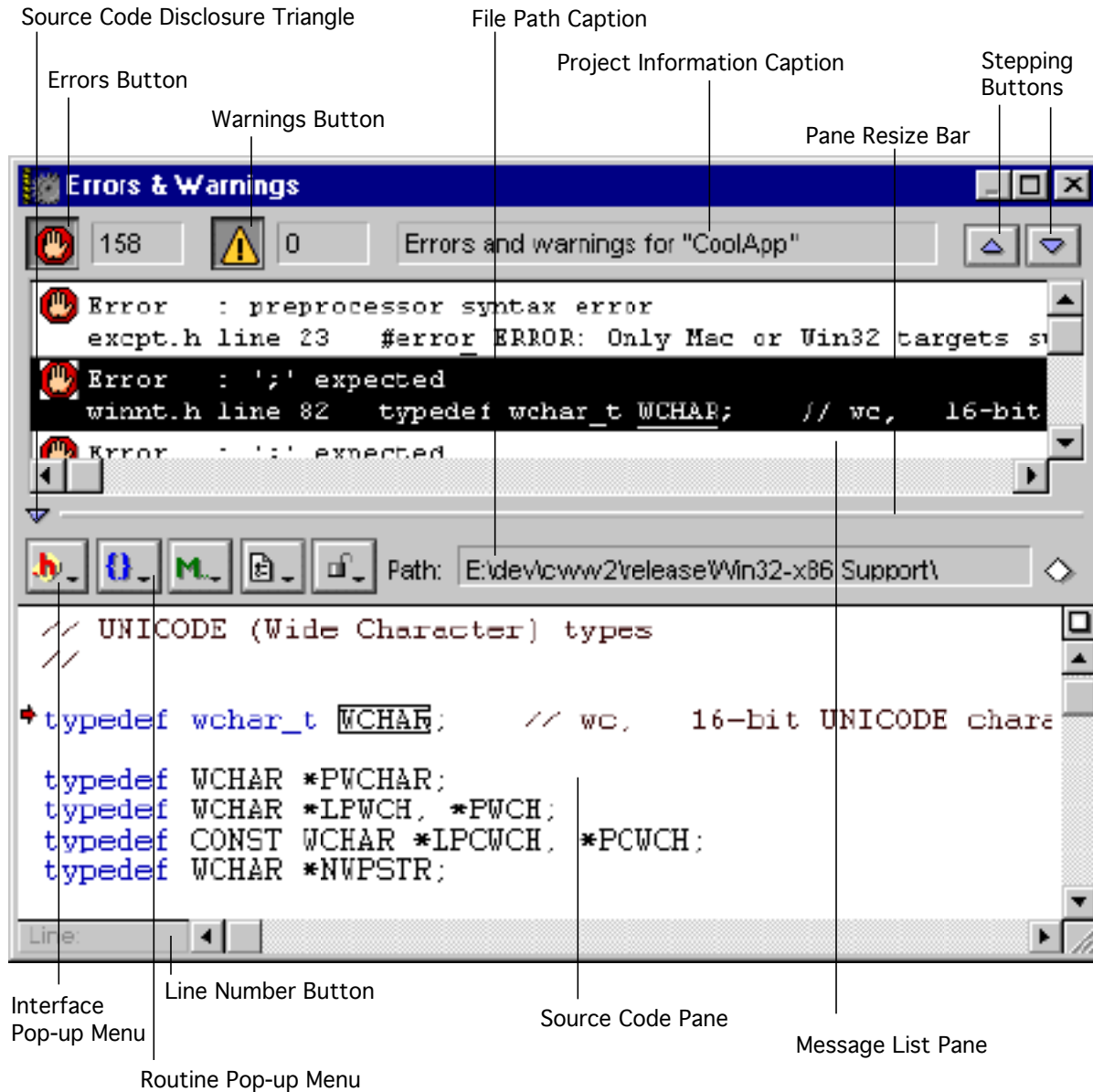
When the compiler finds errors during a build, or the CodeWarrior IDE search command finds text you asked it to look for when [Using Batch Searches](#), you'll see the message window, as shown in [Figure 9.7](#). The window is divided into two panes:

- [Message List Pane](#), which lists the messages, or
- [Source Code Pane](#), which displays the source code for the selected message.

## Compiling and Linking

### Using the Message Window

**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 117](#).

## 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 294](#) 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.

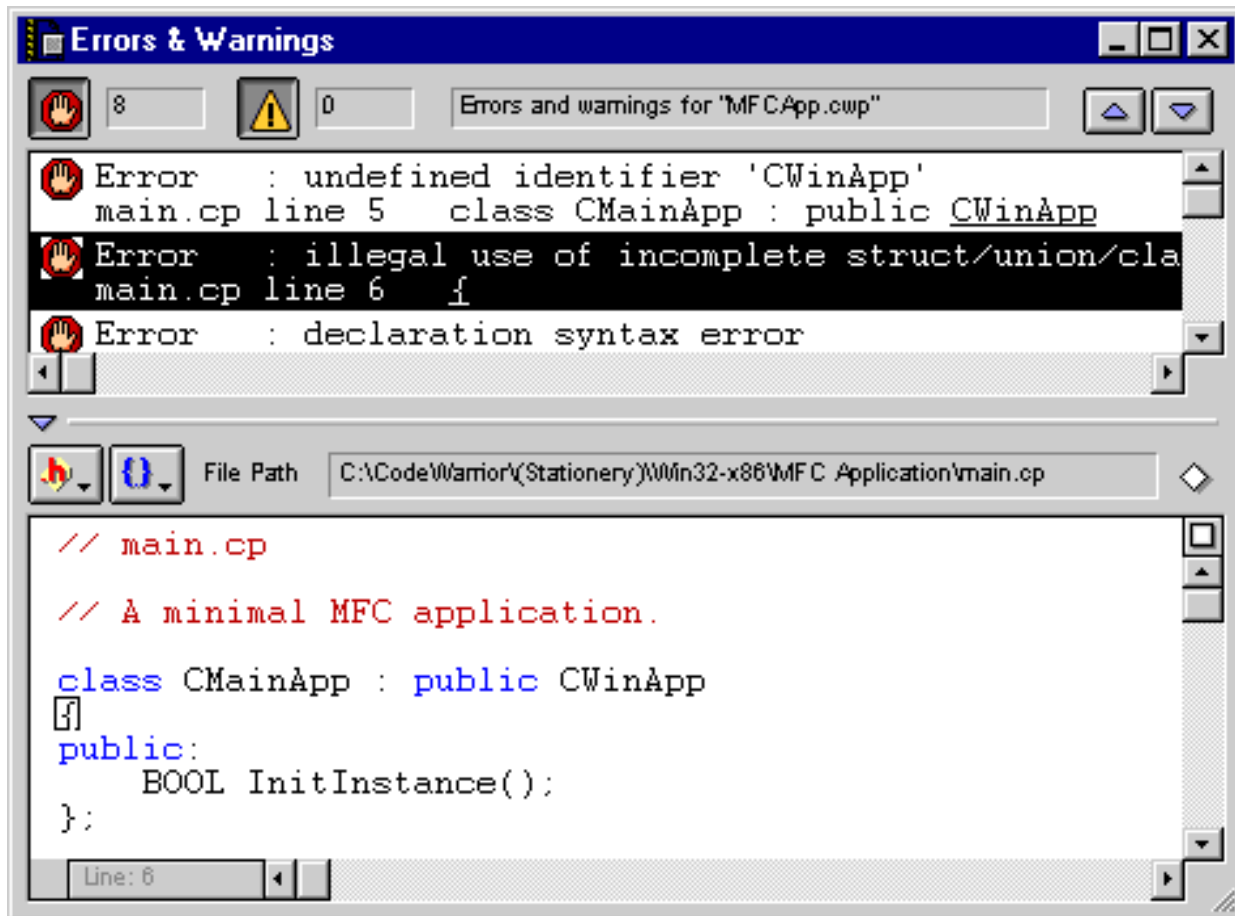
When you do, the file containing the error is brought to the front. The corresponding error is selected in the [Source Code Pane](#), as shown in [Figure 9.8](#).

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 117](#).

## Compiling and Linking

### *Using the Message Window*

**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

When your project is linked, any errors that may occur can be viewed and corrected.

## Viewing Linker Errors

If the linker encounters any errors while linking your project, the Message window appears indicating these errors. This window can be scrolled through using the scroll bar or [Stepping Buttons](#).

To learn about how to scroll through messages in the Message window, refer to [“Stepping Buttons” on page 293](#). To learn about changing the view of messages in the Message window, refer to [“Seeing Errors and Warnings” on page 296](#).

Since Linker errors are a result of problems in the object code, the CodeWarrior IDE cannot show their corresponding errors in the project’s source code files.

## Why Linker Errors Occur

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 25](#). 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

## Compiling and Linking

### *Using the Message Window*

---

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; INTERFACE USES C;	UNIT B; INTERFACE USES C;	UNIT C; INTERFACE
TYPE A_type = ...	TYPE B_type = ....	TYPE C_type = ...
IMPLEMENTATION USES B; ....	IMPLEMENTATION USES A; ...	IMPLEMENTATION 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.**
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.

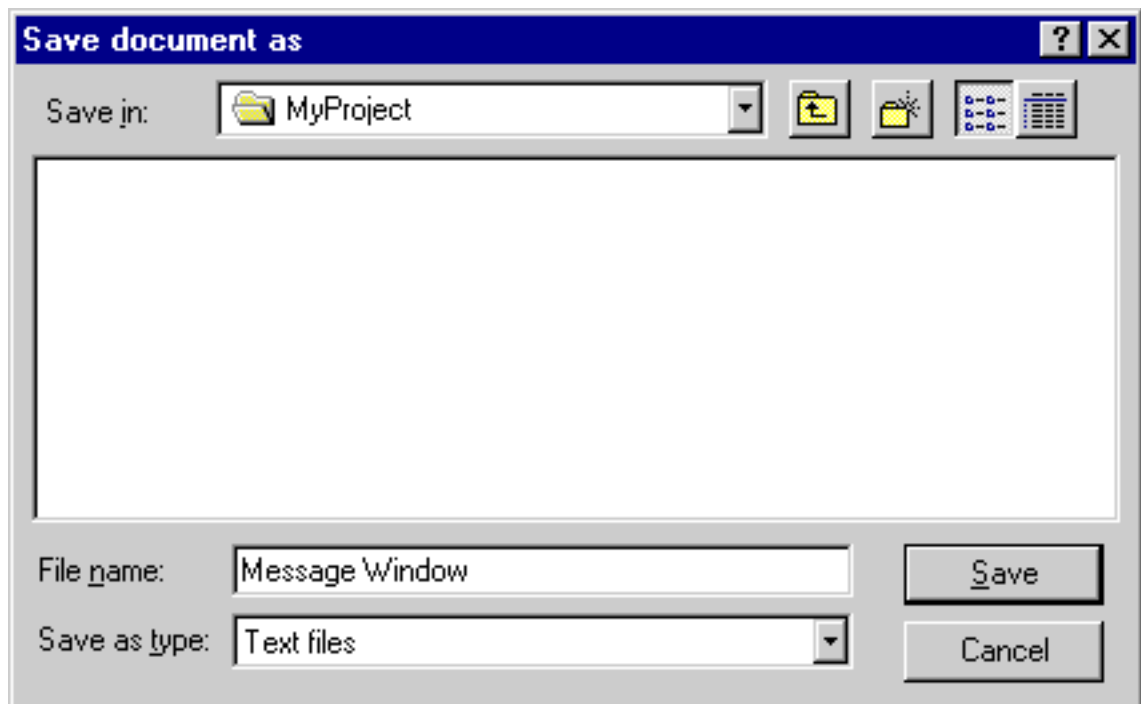
- 1. Make the Message window active.**

To accomplish this, either click on the deactivated Message window, or select the [Errors & Warnings Window](#) command from the [Window Menu](#).

- 2. 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 9.9](#))

**Figure 9.9 Saving the Message window**



## Compiling and Linking

### *Using the Message Window*

---

3. **Specify the name and the location in the dialog box, as shown in [Figure 9.9](#).**

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.

To learn more about printing, refer to the documentation that came with your printer.

## 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.





# Configuring Version Control Software

---

This chapter explains how to use the CodeWarrior IDE version control integration facilities to control your source code.

This chapter is under construction.

## Version Control System Overview

The CodeWarrior IDE includes features for integrating your projects with revision control systems, such as Metrowerks CodeManager and Microsoft Visual SourceSafe.

## Using Source Code Control with Files

The Windows-platform CodeWarrior IDE does not yet support revision control systems, so these features are not yet available.

## **Configuring Version Control Software**

*Using Source Code Control with Files*

---



# 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 menu bar:

- [File Menu](#)
- [Edit Menu](#)
- [Search Menu](#)
- [Project Menu](#)
- [Window Menu](#)
- [Toolbar Submenu](#)
- [Help Menu](#)

The [File Menu](#), [Edit Menu](#), [Search Menu](#), [Project Menu](#), [Window Menu](#), and [Help Menu](#) are visible at all times.

Many menu commands can also have button equivalents on the toolbars. To learn more about how to customize the toolbars, refer to [“Toolbar Customization” on page 269](#).

## File Menu

The File Menu, shown in [Figure 11.1](#), contains commands you use when opening, creating, saving, closing, and printing existing or

# IDE Menu Reference

## File Menu

---

new source code files and projects. The File Menu also provides a few different methods of saving edited files.

**Figure 11.1    The File Menu**



### New

Creates a new editable text file.

To learn more about this command, refer to [“Creating a New File” on page 101](#) for more information.

### New Project

Creates a new project file.

To learn more about this command, refer to [“Creating a Simple Project” on page 46](#).

**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 102](#).

**Open Recent**

The Open Recent command exposes a submenu of projects and files that have recently been opened. You may choose a file from the submenu to instantly open one of these items.

To learn more about this command, refer to [“Opening Files with the File Menu” on page 102](#).

**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 102](#).

**Open Selection**

This menu item allows you to open an existing text file, using the currently-selected text in the editor window as the target file name.

This menu item toggles between Open File... and Open Selection.

[Open Selection](#) appears only when text is selected in the active editor window.

See [“Opening Files with the File Menu” on page 102](#) for more information.

**Close**

Closes the active window whether it is the Project window, the Message Window, or a source code window.

See [“Closing a File” on page 112](#) for more information.

## IDE Menu Reference

### File Menu

---

To learn how to close all open Editor windows, refer to [“Closing All Files” on page 113.](#)

#### **Switch to MW Debugger**

Gives control to the debugger. The line containing 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 the *CodeWarrior Debugger Manual*.

#### **Save**

Saves the contents of the active Editor window to disk.

For more information on this topic, refer to [“Saving One File” on page 108.](#)

#### **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 109.](#)

#### **Save A Copy As**

Saves the active Editor window, Message 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 110.](#)

#### **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 115.](#)

**Print Setup**

Sets the options used when printing files from the CodeWarrior IDE.

For more information about Print Setup, see [“Setting Print Options” on page 114.](#)

**Print**

Prints files from the CodeWarrior IDE on your printer.

For more information on printing files, see [“Printing a Window” on page 114,](#) 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 exiting.

# Edit Menu

The Edit menu, shown in [Figure 11.2](#), contains all the customary editing commands and some CodeWarrior additions, including the commands that open the Preferences and Project Settings dialogs.

Figure 11.2 The Edit Menu

Edit	
Can't Undo	Ctrl+Z
Can't Redo	Ctrl+Shift+Z
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Clear	
Select All	Ctrl+A
Balance	Ctrl+B
Shift Left	Ctrl+[
Shift Right	Ctrl+]
Insert Reference Template	
Preferences...	
Win32 x86 Debug Settings...	

## 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 executed. 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 142](#), and [“Undoing and redoing multiple edits” on page 142](#).

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” on page 221](#).

### **Redo, Multiple Undo, and 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](#), 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 142](#), and [“Undoing and redoing multiple edits” on page 142](#).

To learn about how to configure multiple undo, refer to [“Editor Settings” on page 221](#).

### **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.

## IDE Menu Reference

### *Edit Menu*

---

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 135](#).

#### **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 141](#).

#### **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 141](#).

#### **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 141.](#)

### **Insert Reference Template**

This command is not available on the Windows platform.

### **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 220.](#)

### **Target Settings**

Use this command to change settings for the active target. Note that the name of this menu command will vary depending on the name of your current target.

To learn more about configuring Project Settings, refer to [“Choosing Target Settings” on page 244.](#) To learn how to change the current target, refer to [“Set Current Target” on page 326.](#)

## **Search Menu**

The Search Menu contains all the commands used to find and replace text.

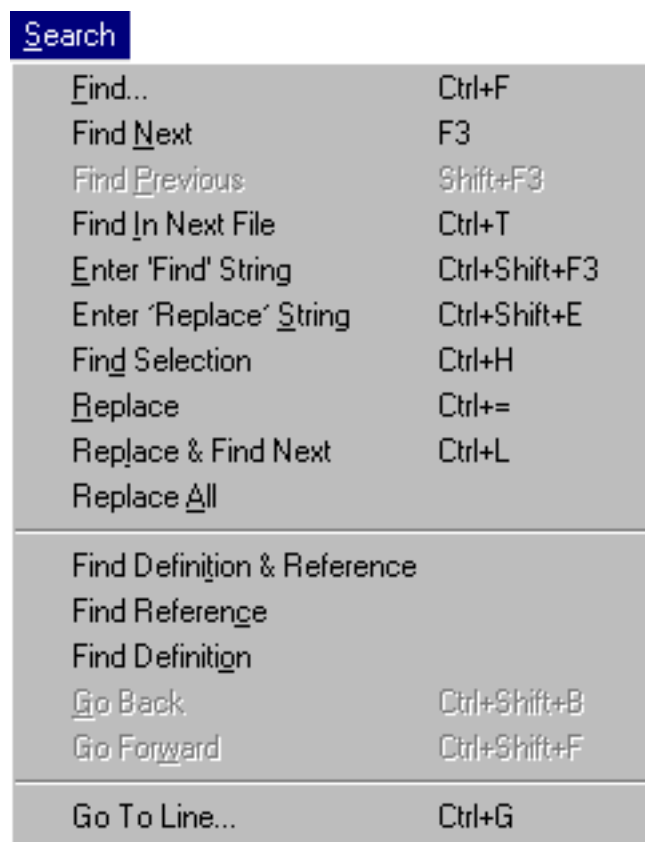
## IDE Menu Reference

### Search Menu

---

There are also some commands for code navigation. Refer to [Figure 11.3](#) to see what this menu looks like when you click on it.

**Figure 11.3 The Search Menu**



### 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 Dialog Box” on page 151.](#)

### 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 165](#).

### 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 165](#).

### 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 158](#), this command is dimmed.

To learn more about this feature, refer to [“Searching and Replacing Text in Multiple Files” on page 171](#).

### 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 135](#).

### 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 168.](#)

#### 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 165.](#)

#### 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 168.](#)

To learn how to select text, refer to [“Selecting Text” on page 135.](#)

#### 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 168.](#)

To learn how to select text, refer to [“Selecting Text” on page 135](#).

### **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**

This feature was not be available on the Windows platform, at the time of this writing. Check the release notes for the latest information.

### **Find Definition and Reference**

This command searches for the definition of the routine name selected in the active window. Searching is done in the source files belonging to the open project, followed by the on-line reference databases.

If no definition is found, a system beep sounds.

This feature was not available on the Windows platform, at the time of this writing. Check the release notes for the latest information.

### **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 208](#).

### **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 208](#).

#### **Go To Line**

Opens a dialog box (in which you enter the line number) and then moves the text insertion point to the line.

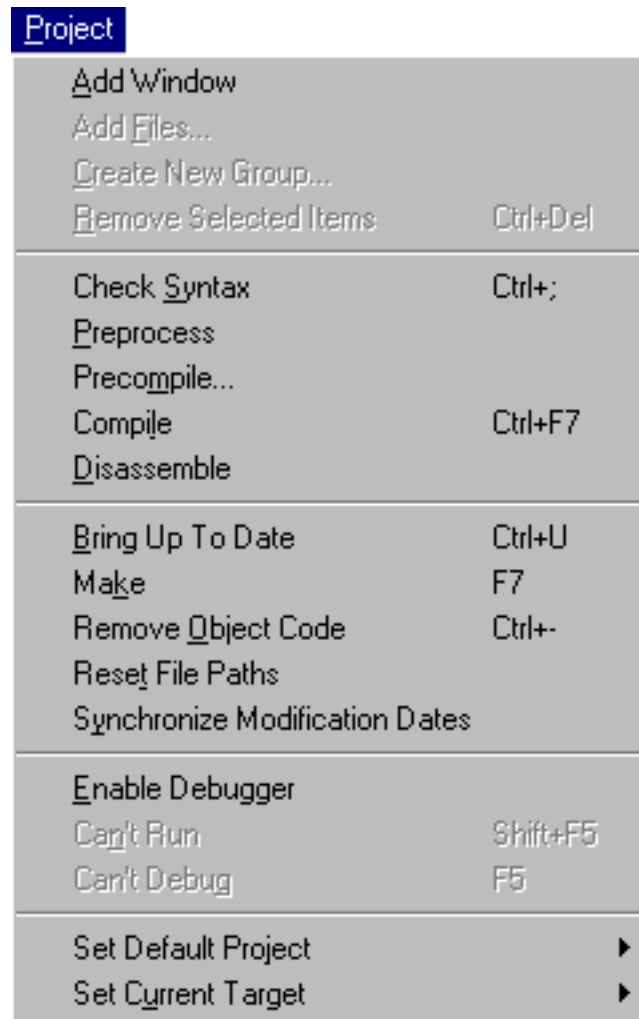
For more information about this feature, refer to [“Going to a Particular Line” on page 149](#).



## Project Menu

The Project menu, shown in [Figure 11.4](#), 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.

**Figure 11.4 The Project Menu**



#### **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 73.](#)

#### **Add Files**

This menu command adds files to the Project window.

To learn more about this feature, refer to [“Using the Add Files Command” on page 70.](#)

#### **Create New Group**

This menu command allows you to create a new group in the current project. This command is in the Project Menu if the Files category is selected in the current project window.

For more information about creating groups, refer to [“Creating Groups” on page 74.](#)

#### **Create New Target**

This menu command allows you to create a new target for the current project. This command is in the Project Menu if the Targets category is selected in the current project window.

For more information about creating targets, refer to [“Creating Complex Projects” on page 79.](#)

#### **Create New Segment**

This menu command allows you to create a new segment (also referred to as a group of files) in the current project. This command is in the Project Menu if the Segments category is selected in the current project window.

For more information about managing segments, refer to [“Managing Files in a Project” on page 65.](#)

### **Remove Selected Items**

This menu command removes the items that are currently-selected from the Project window.

To learn more about removing items from the Project window, refer to [“Managing Files in a Project” on page 65.](#)

---

**WARNING!** This command cannot be undone.

---

### **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 299.](#)

### **Preprocess**

This command performs preprocessing on selected source code files in any language that has a preprocessor, including C, C++, and Pascal.

To learn more about this command, refer to [“Preprocessing Source Code” on page 287.](#)

### **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 Pre-processed Headers” on page 281.](#)

### **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 273.](#)

### **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 289.](#)

### **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 276.](#)

### **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 276.](#)

### **Remove Object Code**

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 Objects” on page 280](#).

### **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, the CodeWarrior IDE marks it for recompilation.

To learn more about this topic, refer to [“Synchronizing modification dates” on page 79](#).

### **Enable Debugger**

This command changes settings to allow your project to be debugged. After using 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 96](#).

### **Disable Debugger**

After using this command, the debugger can not be launched to debug your project.

To learn more about this topic, refer to [“Controlling Debugging in a Project” on page 96](#).

## IDE Menu Reference

### Window Menu

---

#### 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 278.](#)

#### Debug

This menu 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*.

#### Set Default Project

This menu command selects which project is the default project. To learn more about what a default project is, refer to [“Choosing a Default Project” on page 58.](#)

#### Set Current Target

This menu command allows you to choose a different target within the current project to work with. This menu command might be useful if you want to switch between multiple targets in a project and do a build for each one.

## Window Menu

The Window menu, shown in [Figure 11.5](#), includes commands that tile open editor windows, switch between windows, and reopen

previously opened projects. There is also a submenu for customizing the toolbars.

**Figure 11.5 The Window Menu**



### 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 in a single row.

This command is dimmed when the active window is the Project window or Message window.

#### Zoom Window

This menu 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, the CodeWarrior IDE opens it with the saved settings. This command works with Browser windows, Message windows, and Editor windows.

To learn more about this command, refer to, [“Saving Window Settings” on page 131](#), or [“Saving a Default Browser” on page 213](#).

#### Toolbar

This menu item causes the Toolbar submenu to appear. To learn more about this submenu, refer to [“Toolbar Submenu” on page 330](#).

#### 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 189](#). To learn how to activate the Browser, refer to [“Activating the Browser” on page 184](#).



### 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 199](#).

To learn how to activate the Browser, refer to [“Activating the Browser” on page 184](#).

### 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 191](#).

To learn how to activate the Browser, refer to [“Activating the Browser” on page 184](#).

### Build Progress Window

This menu command brings the progress window for builds, as shown in [Figure 9.1 on page 275](#), to the front.

### 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 290](#). Also, refer to [“Using Batch Searches” on page 169](#).

### Project Inspector

This menu command allows you to view information about your project, and also enable debug information generation.

## IDE Menu Reference

### *Toolbar Submenu*

---

To learn more about this command's window, refer to [“Guided Tour of the Project Window” on page 58](#).

#### **Other Window Menu Items**

The other Window Menu items depend solely on which project, source files, header or interfaces 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, do one of the following:

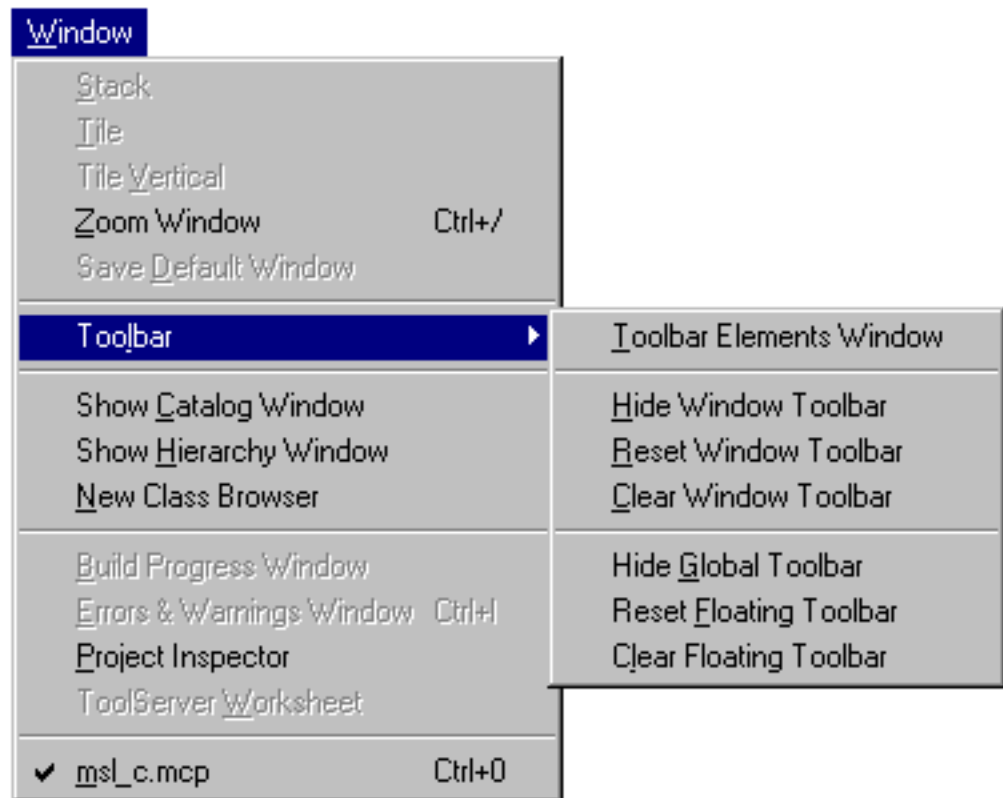
- Click in its window.
- Select it from the [Window Menu](#).
- Use the key equivalent shown in the [Window Menu](#).

## **Toolbar Submenu**

The [Window Menu](#) has another submenu under it for the Toolbar command, as shown in [Figure 11.6](#). The Toolbar submenu contains all the commands used to customize the toolbars that appear in CodeWarrior IDE windows.

To learn more about how to customize the toolbars, read the information in [“Toolbar Customization” on page 269](#).

**Figure 11.6 Toolbar Submenu**



### Toolbar Elements Window

The menu command shows the Toolbar Elements window. From this window you can customize the toolbars by adding icon shortcuts to better suit the way you work.

### Show Window Toolbar and Hide Window Toolbar

These menu commands cause the toolbar in the active window to disappear or reappear. The actual command shown in the menu will toggle between Show Window Toolbar and Hide Window Toolbar, depending on whether the active window's toolbar is visible or not.

#### **Reset Window Toolbar**

This menu command causes the toolbar in the active window to reset to a default state. You should use this menu command if you want to return the Editor window toolbar to the original default state.

#### **Clear Window Toolbar**

This menu command causes the toolbar in the active Editor, Project, or Browser window to have all icons removed from it. Once all the icons have been removed, you can add icons using the Toolbar Elements window.

Use the Reset Window Toolbar command to cause all the default icons to come back.

#### **Show Global Toolbar and Hide Global Toolbar**

These menu commands cause the Global Toolbar to appear or disappear. The actual command shown in the menu will toggle between Show Global Toolbar and Hide Global Toolbar, depending on whether the Global Toolbar is already visible or not.

#### **Reset Floating Toolbar**

This menu command causes the Global Toolbar to return to its default state. You should use this menu command if you want to return the Global Toolbar to the original default state.

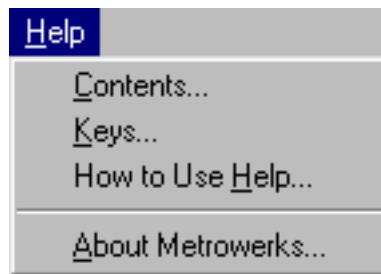
#### **Clear Floating Toolbar**

This menu command causes the Global Toolbar to have all icons removed from it. Once all the icons have been removed, you can add icons using the Toolbar Elements window.

## Help Menu

The Help menu, shown in [Figure 11.7](#), 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.

**Figure 11.7**    **Help Menu**



### **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 283  
.cwp 54  
.dump 289

## Numerics

68K 19  
80x86 20

## A

About Metrowerks command 333  
Add Default button 249  
Add File command 322  
Add Files command 322  
Add Window command 73, 322  
Alert Yourself After Build option 281  
Arrow Keys 134  
arrow keys 59  
Automatic updating 284

## B

Background Color 225  
backup files 110  
Balance command 141, 314  
Balance While Typing option 222  
Balancing punctuation 141  
Batch search 169  
BeOS 20, 25  
boxes 199  
Bring Up To Date command 281, 324  
browser 183–213  
    activating 184, 252  
    analyzing inheritance 212  
    base classes in hierarchy 202  
    base classes in single-class 199  
    catalog view 185, 189  
    classes pane 195  
    customizing windows 213  
    data members pane 195  
    declaration button 199  
    editing code 212  
    file button 194  
    finding function overrides 212

    hierarchy view 187  
    identifier icon 196  
    interface 188–207  
    lines in hierarchy 201  
    member functions pane 195  
    multi-class 191–196  
    multi-class hierarchy 199–202  
    navigating code with 205  
    opening source file 210  
    orientation button 193  
    resize bar 196  
    saving windows 213  
    seeing declaration 211  
    seeing function definition 211  
    seeing PowerPlant in 213  
    show hierarchy button 199  
    showing data in single-class 199  
    showing subclasses in hierarchy 201  
    single-class 197  
    single-class hierarchy 202  
    source pane 196  
    strategy 184–188  
    symbol window 204  
    synchronized class selection 195, 201  
    using 207–213  
    viewing options 184  
browser view 186  
Build Progress Window command 329

## C

C 20  
C++ 20  
catalog view 185  
catalog window  
    browser 189  
category pop-up menu, in category window 190  
Check Syntax command 323  
classes pane in browser 195  
Clear command 140, 314  
Clear Floating Toolbar command 332  
Clear Window Toolbar command 332  
Close All command 113  
Close command 309  
Code Column 62

## Index

---

- code disassembly
  - 68K 262
  - PPC 264
- code generation
  - 68K 261
  - PowerPC 264
  - x86 options 264
- code navigation pop-up menu 205
- CodeWarrior IDE 23
- comments, coloring 227
- Compile column 76
- Compile command 324
- Compiler 255
- Compiling 271–304
  - project 273
- compiling
  - 68K options 261
  - BeOS options 264
  - Mac OS options 264
  - Power TV options 264
  - source files 276
- Compiling and Linking
  - choosing a compiler 272
  - compiling files 274–276
  - debugging 278
  - disassembling 289
  - guided tour 290–295
  - link map 279
  - making a project 276
  - options 280
  - overview 271
  - plugin compilers 272
  - precompiling headers 281–287
  - preprocessing 287
  - removing binaries 280
  - removing object code 280
  - running 278
  - setting file extension 273
  - speeding 281
  - Synchronizing Modification Dates 279
  - touch and untouch 274
- compiling one file 275
- compiling selected files 275
- Contents command 333
- Context Popup Delay option 224
- conventions 21

- Copy command 140, 313
- Create Folder 49
- Create New Group command 322
- Create New Segment command 322
- Create New Target command 322
- custom keyword, coloring 227, 228
- Cut command 140, 313

## D

- Data Column 63
- data members pane, in browser 195
- date caching 281
- Debug Column 63, 97
- Debug Info Marker 97
- debugger
  - generating information 97
- Debugging configuration 277
- Defining Symbols for C/C++ 285
- Defining Symbols for Pascal 286
- Disable Debugger command 325
- Disassemble command 324
- Disassembling source code 289
- disassembly
  - 68K 262
  - PPC 264
- documentation
  - viewers 24
- DOS text files 111
- Drag & Drop editing support 225
- Drag and Drop text 136–138
- Dynamic Scroll option 222

## E

- Edit Menu 38, 311–315
  - Balance command 314
  - Clear command 314
  - Copy command 313
  - Cut command 313
  - Insert Reference Template command 315
  - Multiple Redo 313
  - Multiple Undo 313
  - Paste command 313
  - Preferences command 315
  - Redo command 313
  - Select All command 314



- Shift Left command 314
- Shift Right command 314
- Target Settings command 315
- Undo command 312
- editing
  - in browser 212
- editing, redoing 142
- editing, undoing 142
- Editor 117–149
  - adding text 134
  - balancing punctuation 141
  - color syntax 143
  - configuration 126–132
  - deleting text 135
  - drag and drop 225
  - finding a routine in 144
  - font 127
  - Go Back and Go Forward 149
  - go to line number 149
  - guided tour 117–126
  - markers 144–147
  - moving text 136–138
  - navigating text 143–149
  - opening related file 147
  - overview 117, 305
  - panes 129–131
  - saving window settings 131
  - selecting text 135
  - text editing 132–143
  - text size 127
  - undoing changes 142–143
  - user interface elements 117
- Enable Debugging command 325
- End Key 59
- End key 134
- Enter 'Find' String command 164, 317
- Enter 'Replace' String command 317
- Enter 'Find' String command 317
- Enter 'Replace' String command 317
- Entire Word Check Box 167
- Error Button 292
- Error Messages 296
- error messages
  - compiler 297
- Errors and Warnings Window command 329
- Exit command 311

extension 254

## F

- Factory Settings button 219
- file button, in browser 194
- File Column 62
- File extension 254
- File Menu 37, 307–311
  - Close command 309
  - Exit command 311
  - New command 308
  - New Project command 308
  - Open command 309
  - Open File command 309
  - Open Recent command 309
  - Open Selection command 309
  - Print command 311
  - Print Setup command 311
  - Revert command 310
  - Save A Copy As command 310
  - Save As command 310
  - Save command 310
  - Switch to MW Debugger command 310
- file name suffix 254
- File Path Caption 124, 295
- File Sets List 160
- File Sets Pop-up Menu 159
- Files 101–116
  - closing 112–113
  - creating 101
  - opening 147
  - opening existing 102–107
  - overview 101
  - printing 113–115
  - reverting to saved 115
  - saving 107–112
  - selecting 67
- files
  - selecting 68
- Files Sets
  - saving 176
- Find command 316
- Find Definition & Reference command 319
- Find in Next File command 317
- Find Next command 164, 317
- Find Previous command 164, 166, 317

## Index

---

Find Selection command 318  
Find Window  
    guided tour 151–162  
finding all implementations of function 204  
Flashing Delay option 224  
function, finding all 204

## G

Go Back command 319  
Go Forward command 319  
Go To Line command 320  
grep 178–182  
Group File Pop-up 64  
Group File Pop-up menu 77  
groups 61, 62, 64, 65–76  
    creating 74  
    moving 73  
    naming 76  
    removing 75  
    selecting 67, 68

## H

header files  
    opening 107  
    precompiling 281–287  
Help Menu 42, 333  
    About Metrowerks command 333  
    Contents command 333  
    How to Use Help command 333  
    Keys command 333  
Hide Global Toolbar command 332  
Hide Window Toolbar command 331  
hierarchy view  
    browser 187  
Home Key 59  
Home key 134  
How to Use Help command 333

## I

IDE 19, 23, 32  
    guided tour 35–43  
    installation 32  
    Menus 36  
    Toolbar 35, 42  
IDE Toolbar 36

identifier icon in browser 196  
Ignore Case Check Box 167  
Ignored by Make option 255  
Insert Reference Template command 315  
Installation 32  
Integrated Development Environment 19  
Interface Pop-up Menu 119, 294  
Interfaces File Pop-up 64  
Interfaces File Pop-up menu 77

## J

Java 20, 25  
Java projects 252, 264, 267

## K

Keys command 333  
keywords, coloring 227

## L

Launchable option 255  
line button in hierarchy browser 201  
Line Number Button 125, 295  
line number, going to 149  
Linking 271–304  
linking  
    Motorola 68K 265, 266  
    Power TV 267  
    PowerPC 267  
list button, in browser  
    browser  
        list button 194

## M

Macintosh 19  
Main Text Color 225  
Make 276  
Make command 281, 324  
manual style 21  
MAP 279  
Marker Pop-up Menu 122  
marker, adding to text 123  
Marking files for compilation 78  
member functions pane, in browser 195

Message List Pane 293  
Message Window  
    command 329  
    correcting compiler errors 299  
    error and warning messages 296  
    stepping through messages 297  
    using 295–304  
Modification dates, synchronizing 79  
Motorola 68K  
    code generation 261  
multi-class browser 191–196  
multi-class hierarchy, browser 199–202  
Multi-file searches 158–162  
multiple redo 313  
multiple undo 313  
multiple Undo command 142

## N

New Class Browser command 329  
New command 308  
New Project command 47, 54, 308  
number, going to line 149

## O

Open command 55, 309  
Open File command 309  
Open Recent command 309  
Open Selection command 106, 309  
opening file with browser 210  
optimization  
    IR options 263  
Options 215–261  
    advanced compile options 280  
    browser coloring 230  
    C/C++ Compiler panel 258  
    C/C++ Warnings Panel 259  
    Editor settings 221  
    fonts and tabs 225  
    guided tour 216  
    overview 215  
    Pascal Compiler panel 260  
    Pascal Warnings panel 260  
    preferences 220  
    syntax coloring 226–230  
    target settings 244–268

    x86 Linker panel 268  
    x86 Project panel 257  
options  
    68K code generation 261  
    68K disassembly 262  
    68K linker 265  
    Access Paths 247–251  
    Build Extras 251–252  
    CFM68K linker 266  
    Custom Keywords panel 245  
    IR Optimizer 263  
    Java projects 252, 264, 267  
    PowerPC code generation 264  
    PowerPC disassembly 264  
    PowerPC PEF 268  
    PPC linker 267  
    project settings 216  
    resource compiler 261  
    x86 code generation 264  
Options Pop-up Menu 123  
orientation button  
    in browser 193  
Others Button 162

## P

Page Down key 59, 134  
Page Up key 59, 134  
Palm OS 20, 25  
Pane Resize Bar 294  
Pane Splitter Controls 126  
Panels  
    in editor window 129–131  
Pascal 20  
Paste command 140, 313  
PEF 268  
PlayStation 20, 25  
Pop-up Menu Disclosure Button 126, 294  
Post Linker option 256  
PowerPC 19  
PowerTV OS 20, 25  
PPC  
    code generation  
        PPC 264  
Precompile command 283, 323  
precompile\_target, #pragma 283

## Index

---

- Precompiled 255
- Precompiled headers
  - automatic updating 284
  - creating 282
- precompiled headers
  - automatic updating of 284
- precompiling 281–287
- Preferences 53
  - choosing 220
- Preferences command 315
- Preprocess command 323
- Preprocessing code 287
- preprocessor 287
- Print command 114, 311
- Print Setup command 311
- printing 113–115
- Program Executable Format 268
- project
  - naming 49
  - opening 55
- Project Headers Check Box 162
- Project Information Caption 293
- Project Inspector command 329
- Project Menu 40, 321–326
  - Add File command 322
  - Add Window command 322
  - Bring Up To Date command 324
  - Check Syntax command 323
  - Compile command 324
  - Create New Group command 322
  - Create New Segment command 322
  - Create New Target command 322
  - Disable Debugger command 325
  - Disassemble command 324
  - Enable Debugging command 325
  - Make command 324
  - Precompile command 323
  - Preprocess command 323
  - Remove Binaries command 325
  - Remove Object Code command 325
  - Remove Selected Items command 323
  - Reset File Paths command 323, 325
  - Run command 326
  - Set Current Target command 326
  - Set Default Project command 326
  - Synchronize Modification Dates
    - command 325
- Project Settings 216
- Project stationery
  - creating 52
- project stationery 46, 52
  - choosing 48, 52
  - folder 52
- Project Stationery folder 53
- Project Switch List submenu 55
- Project Window
  - guided tour 58–64
  - navigating 59
  - user interface items 59
- Projects 45–100
  - adding files 69–73
  - building 52, 273
  - choosing stationery 47
  - compiling 273
  - creating 46–55
  - creating groups 74
  - debug enabling 277
  - debugging 278
  - Expanding and Collapsing Groups 66
  - expanding and collapsing groups 66
  - groups and segments 66
  - Items Saved with 57
  - making 276
  - managing files in 65–79
  - modifying 51
  - moving around 59
  - moving files and groups 73
  - new 48
  - open command 55
  - opening existing 55–56
  - opening with Project Switch List 55
  - removing files and groups 75
  - running 278
  - save as type options 57
  - saving 56–57
  - selecting files and groups 67–69
  - settings 244–268
  - stationery folder 52
  - stationery, about 46
  - touching and untouching files 76
  - updating 276
  - using stationery 47

## projects

- adding preprocessor symbols to 99
- building 281
- closing 57
- compiling 276, 281
- debug setup 96–98
- items saved with 57
- naming groups 76
- saving 56
- selecting files 67
- selecting groups 67
- settings 216
- switching between 55

**Q**

QuickStart 24, 32

**R**

Recent Strings pop-up menu 165, 168

recompiling 276

recompiling files 276

Redo command 142, 313

regular expressions 178–182

Release notes 21

Remove a file set command 177

Remove Binaries command 325

Remove Files command 75

Remove Object Code command 325

Remove Selected Items command 323

Replace & Find Next command 318

Replace All command 168, 319

Replace command 318

replacing

- in multiple files 171–178

Reset File Paths command 323, 325

Reset Floating Toolbar command 332

Reset Window Toolbar command 332

resize bar, in browser 196

Resource file option 255

resources

- WinRC compiler 261

Revert command 310

Revert Panel button 219

Routine Pop-up Menu 144, 294

routine pop-Up menu 121

Run command 326

**S**

Save A Copy As command 53, 110, 310

Save All Before "Update" option 223

Save As command 109, 310

Save command 310

Save Default Window command 328

Save this File Set command 176

Search Menu 39, 315–320

- Enter 'Find' String command 317
- Enter 'Replace' String command 317
- Find command 316
- Find Definition & Reference command 319
- Find in Next File command 317
- Find Next command 317
- Find Previous command 317
- Find Selection command 318
- Go Back command 319
- Go Forward command 319
- Go To Line command 320
- Replace & Find Next command 318
- Replace All command 319
- Replace command 318

Search Menu Find Selection command 318

searching 163

- for selection 164
- multi-file 158–162
- multiple files 171–178
- selected text 163–164

segments 62, 65, 66

Select All command 314

selected text search 163

selection

- finding 164

Set Current Target command 326

Set Default Project command 326

Shift Left command 141, 314

Shift Right command 141, 314

Show Catalog Window command 328, 329

Show Global Toolbar command 332

Show Hierarchy Window command 329

Show Window Toolbar command 331

single-class browser 197

single-class hierarchy, in browser 202

## Index

---

- Sort Function Pop-Up 225
- Source Code Disclosure Triangle 294
- Source Code Pane 294
- source files
  - compiling 276
  - precompiling 281–287
- source pane, in browser 196
- Sources Check Box 162
- Stack command 327
- stationery 46, 47
- Stepping Buttons 293
- Stop at EOF option 161, 178
- Switch to MW Debugger command 310
- symbol window, in browser 204
- Synchronize Modification Dates command 325
- Synchronizing modification dates 79
- Syntax coloring, table of 227
- System Headers Check Box 162
- System Include Path Pane 248
- System Requirements 31

## T

- Target Settings command 315
- Targets
  - documentation 24
  - IDE 25
  - settings 244–268
- Text
  - drag and drop of 136–138
- Text Editing Area 119
- text replace
  - multiple file 171–178
  - Replace All 168
  - replacing found text 168
  - selective replace 168
  - single file 164–170
- text search 151–182
  - activating multi-file 171
  - Batch search 169
  - choosing file sets 175
  - choosing files 172
  - controlling range 166
  - controlling search parameters 167
  - controlling search range 177
  - finding selection 164
  - finding text 165
  - for selection 164
  - multi-file 158–162
  - multiple file 171–178
  - overview 151
  - regular expressions 178–182
  - removing file sets 177
  - saving file sets 176
  - selected text search 163
  - single file 164–170
- Tile command 328
- Tile Vertical command 328
- Toolbar
  - using Shift and Option with 43
- Toolbar Elements Window command 331
- Toolbar Submenu 330–332
  - Clear Floating Toolbar command 332
  - Clear Window Toolbar command 332
  - Hide Global Toolbar command 332
  - Hide Window Toolbar command 331
  - Reset Floating Toolbar command 332
  - Reset Window Toolbar command 332
  - Show Global Toolbar command 332
  - Show Window Toolbar command 331
  - Toolbar Elements Window command 331
- Toolbar submenu 328
- Touch Column 63
- Touch command 78
- Treat #include as #include "..." option 248
- Turbo Pascal 20
- Tutorial Resources 24
- typographical conventions 21

## U

- Undo command 312, 313
- UNIX text files 111
- updating projects 276, 281
- Use Modification Dates Caching option 251
- Use Multiple Undo option 224
- User Include Path Pane 248

## V

- viewers 24

### W

Warning Button 293  
Warning Messages 296  
wildcard searching 178–182  
Win32/x86 20, 25  
Window Menu 41, 326–330

- Build Progress Window command 329
- Errors and Warnings Window command 329
- New Class Browser command 329
- Project Inspector command 329
- Save Default Window command 328
- Show Catalog Window command 328, 329

Show Hierarchy Window command 329  
Stack command 327  
Tile command 328  
Tile Vertical command 328  
Toolbar submenu 328  
Zoom Window command 328  
Windows 95 20, 31  
Windows NT 20, 31  
WinRC resource compiler 261

### Z

Zoom Window command 328





# CodeWarrior

## IDE User Guide

### Credits

**writing lead:** “BitHead” Magnuson

**other writers:** 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, Kevin Bell, Mark Anderson, Fred Peterson

**frontline warriors:** John Roseborough, Marc Paquette



# Guide to CodeWarrior Documentation

If you need information about...	See this
Installing updates to CodeWarrior	<i>QuickStart Guide</i>
Getting started using CodeWarrior	<i>QuickStart Guide</i> <i>Tutorials</i> (Apple Guide)
Using CodeWarrior IDE (Integrated Development Environment)	<i>IDE User Guide</i>
Debugging	<i>Debugger Manual</i>
Important last-minute information on new features and changes	<i>Release Notes</i> folder
Creating Mac OS software	<i>Targeting Mac OS, &amp; Mac OS</i> folder
Creating Win32/x86 software	<i>Targeting Win32, &amp; Win32/x86</i> folder
Creating Java software	<i>Targeting Java, &amp; Sun Java Documenta- tion</i> folder
Creating software for other targets, such as PlayStation game console and PalmPilot	the appropriate <i>Targeting</i> manual for the target of interest
Using ToolServer with the CodeWarrior editor	<i>IDE User Guide</i>
Controlling CodeWarrior through AppleScript	<i>IDE User Guide</i>
Using CodeWarrior to program in MPW	<i>Command Line Tools Manual</i>
Programming in C, C++, and inline assembly for 68K, PowerPC, MIPS, and x86	<i>C Compiler Guide, MSL C Reference, MSL C++ Reference</i>
Pascal or Object Pascal programming	<i>Pascal Compiler Guide, Pascal Lan- guage Manual, Pascal Library Reference</i>
Programming in assembly language	<i>Assembler Guide</i>
Fixing compiler and linker errors	<i>Errors Reference</i>
Fixing Mac OS memory bugs	<i>ZoneRanger Manual</i>
Speeding up your Mac OS programs	<i>Profiler Manual</i>
PowerPlant Programming	<i>The PowerPlant Book</i> <i>PowerPlant Advanced Topics</i> PowerPlant reference documents
Creating a PowerPlant visual interface	<i>Constructor Manual</i>
Creating a Java visual interface	<i>Constructor for Java Manual</i>
Creating a PalmPilot visual interface	<i>Constructor for PalmPilot Manual</i>
Learning how to program for Mac OS, Windows, or Java	<i>Discover Programming</i> series of CDs
Contacting Metrowerks about registration, sales, and licensing	<i>Quick Start Guide</i>
Contacting Metrowerks about problems and suggestions using CodeWarrior software	<i>email Report Forms</i> in the <i>Release Notes</i> folder
Sample programs and examples	<i>CodeWarrior Examples</i> folder <i>The PowerPlant Book</i> <i>PowerPlant Advanced Topics</i> <i>Tutorials</i> (Apple Guide)
Problems other CodeWarrior users have solved	<i>Internet newsgroup [docs]</i> folder