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.

Revised: 980304–Isf
# Table of Contents

## 1 Introduction

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction Overview</td>
<td>15</td>
</tr>
<tr>
<td>Read the Release Notes!</td>
<td>17</td>
</tr>
<tr>
<td>Manual Conventions</td>
<td>17</td>
</tr>
<tr>
<td>Typographical Conventions</td>
<td>17</td>
</tr>
<tr>
<td>Host Conventions</td>
<td>18</td>
</tr>
<tr>
<td>Figure Conventions</td>
<td>18</td>
</tr>
<tr>
<td>Keyboard Conventions</td>
<td>19</td>
</tr>
<tr>
<td>IDE User Guide Overview</td>
<td>20</td>
</tr>
<tr>
<td>About the CodeWarrior IDE</td>
<td>21</td>
</tr>
<tr>
<td>Where to Go From Here</td>
<td>22</td>
</tr>
<tr>
<td>QuickStart and Tutorials</td>
<td>22</td>
</tr>
<tr>
<td>Targeting Documentation</td>
<td>23</td>
</tr>
<tr>
<td>What’s New in This Release</td>
<td>24</td>
</tr>
<tr>
<td>Integrated Debugger</td>
<td>25</td>
</tr>
<tr>
<td>File Compare and Merge Tool</td>
<td>25</td>
</tr>
<tr>
<td>Project Manager Improvements</td>
<td>25</td>
</tr>
<tr>
<td>Editor Improvements</td>
<td>25</td>
</tr>
<tr>
<td>Version Control Support (Windows)</td>
<td>25</td>
</tr>
<tr>
<td>Online Help Support (Windows)</td>
<td>26</td>
</tr>
<tr>
<td>Native Plug-ins Support (Windows)</td>
<td>26</td>
</tr>
</tbody>
</table>

## 2 Getting Started

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting Started Overview</td>
<td>27</td>
</tr>
<tr>
<td>System Requirements</td>
<td>28</td>
</tr>
<tr>
<td>Windows Requirements</td>
<td>28</td>
</tr>
<tr>
<td>Mac OS Requirements</td>
<td>28</td>
</tr>
<tr>
<td>CodeWarrior IDE Installation</td>
<td>28</td>
</tr>
<tr>
<td>Programming Concepts</td>
<td>29</td>
</tr>
<tr>
<td>Creating Input Files</td>
<td>29</td>
</tr>
<tr>
<td>Generating the Software</td>
<td>31</td>
</tr>
<tr>
<td>Debugging and Refining</td>
<td>32</td>
</tr>
<tr>
<td>An Introduction to the CodeWarrior IDE</td>
<td>32</td>
</tr>
<tr>
<td>Projects and Targets</td>
<td>33</td>
</tr>
</tbody>
</table>
# Table of Contents

Source Code Editing and Browsing ........................................... 34  
Compiling and Linking ....................................................... 34  
Project Debugging ............................................................ 35  
Version Control ............................................................... 35  
Customizing the IDE ......................................................... 36  
Mac OS External Tool Support ............................................... 37  
Mac OS Scripting .............................................................. 37

## 3 Working with Projects ................................................. 39

Working with Projects Overview ............................................. 39  
Guided Tour of the Project Window ......................................... 40  
Navigating the Project Window ............................................. 41  
Project Window Toolbar ..................................................... 41  
File View ........................................................................... 42  
Link Order View .................................................................. 47  
Targets View ....................................................................... 47  
Creating a New Project ....................................................... 48  
Choosing the New Project Stationery File .............................. 48  
Naming Your New Project .................................................... 50  
Modifying Your New Project ................................................. 52  
Building Your New Project .................................................. 52  
Working with Project Stationery ............................................ 52  
About Project Stationery ...................................................... 53  
About the Project Stationery Folder ....................................... 53  
Creating Your Own Project Stationery ..................................... 54  
Opening an Existing Project ............................................... 56  
Using the Open Command .................................................... 57  
Using the Open Recent Command ......................................... 57  
Using the Project Window to Open Subprojects ..................... 58  
Opening Project Files Created on Other Host Platforms .......... 58  
Opening Project Files from Earlier IDE Versions ..................... 59  
Saving a Project ................................................................. 61  
Items Saved with Your Project .............................................. 61  
Saving a Copy of Your Project .............................................. 62  
Closing a Project .................................................................. 62
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choosing a Default Project</td>
<td>63</td>
</tr>
<tr>
<td>Managing Files in a Project</td>
<td>63</td>
</tr>
<tr>
<td>Expanding and Collapsing Groups</td>
<td>64</td>
</tr>
<tr>
<td>Selecting Files and Groups</td>
<td>65</td>
</tr>
<tr>
<td>Adding Files</td>
<td>67</td>
</tr>
<tr>
<td>Moving Files and Groups</td>
<td>73</td>
</tr>
<tr>
<td>Creating Groups</td>
<td>74</td>
</tr>
<tr>
<td>Removing Files and Groups</td>
<td>75</td>
</tr>
<tr>
<td>Renaming Groups</td>
<td>76</td>
</tr>
<tr>
<td>Touching and Untouching Files</td>
<td>77</td>
</tr>
<tr>
<td>Working with Complex Projects</td>
<td>79</td>
</tr>
<tr>
<td>What is a Target?</td>
<td>80</td>
</tr>
<tr>
<td>What is a Subproject?</td>
<td>80</td>
</tr>
<tr>
<td>Strategy for Creating Complex Projects</td>
<td>82</td>
</tr>
<tr>
<td>Creating a New Target</td>
<td>82</td>
</tr>
<tr>
<td>Changing a Target Name</td>
<td>84</td>
</tr>
<tr>
<td>Changing the Target Settings</td>
<td>84</td>
</tr>
<tr>
<td>Setting the Current Build Target</td>
<td>85</td>
</tr>
<tr>
<td>Creating Target Dependencies</td>
<td>85</td>
</tr>
<tr>
<td>Assigning Files to Targets</td>
<td>86</td>
</tr>
<tr>
<td>Creating Subprojects Within Projects</td>
<td>88</td>
</tr>
<tr>
<td>Examining Project Information</td>
<td>89</td>
</tr>
<tr>
<td>Moving a Project</td>
<td>90</td>
</tr>
<tr>
<td>Controlling Debugging in a Project</td>
<td>91</td>
</tr>
<tr>
<td>Activating Debugging for a Project</td>
<td>91</td>
</tr>
<tr>
<td>Activating Debugging for a File</td>
<td>91</td>
</tr>
<tr>
<td>Adding Preprocessor Symbols to a Project</td>
<td>94</td>
</tr>
<tr>
<td>4 Working with Files</td>
<td>95</td>
</tr>
<tr>
<td>Working with Files Overview</td>
<td>95</td>
</tr>
<tr>
<td>Creating a New File</td>
<td>95</td>
</tr>
<tr>
<td>Opening an Existing File</td>
<td>96</td>
</tr>
<tr>
<td>Opening Files with the File Menu</td>
<td>96</td>
</tr>
<tr>
<td>Opening Files from the Project Window</td>
<td>98</td>
</tr>
<tr>
<td>Opening Files from an Editor Window</td>
<td>100</td>
</tr>
</tbody>
</table>
## 5 Editing Source Code

Source Code Editor Overview .................................................. 121
Guided Tour of the Editor Window ............................................. 122
Text Editing Area ................................................................. 123
Interface Pop-Up Menu ......................................................... 124
Routine Pop-Up Menu .......................................................... 124
Marker Pop-Up Menu ........................................................... 126
Options Pop-Up Menu ........................................................... 126
VCS Pop-up Menu ............................................................... 127
File Path Caption ............................................................... 127
Dirty File Marker ............................................................... 128
Pane Splitter Controls ......................................................... 128
Line Number Button ............................................................ 128
Toolbar Disclosure Button ..................................................... 128
Mac OS Path Pop-Up Menu ................................................... 129
Editor Window Configuration ................................................ 129
Setting Text Size and Font .................................................... 130
Seeing Window Controls ....................................................... 130
Splitting the Window into Panes ............................................ 131
Saving Editor Window Settings ............................................... 132
## Table of Contents

Basic Text Editing ............................................. 133
Basic Editor Window Navigation ......................... 134
Adding Text .................................................. 135
Deleting Text ................................................ 135
Selecting Text ............................................... 136
Moving Text (Drag and Drop) ............................ 138
Using Cut, Copy, Paste, and Clear .................... 138
Balancing Punctuation .................................... 139
Shifting Text Left and Right ............................ 140
Undoing Changes .......................................... 140
Controlling Color ......................................... 141
Navigating in Text ........................................... 142
Finding a Routine .......................................... 142
Adding, Removing, and Selecting a Marker .......... 142
Opening a Related File .................................. 145
Going to a Particular Line ............................... 146
Using Go Back and Go Forward ....................... 146
Configuring Editor Commands ......................... 147
Opening a File’s Directory (Mac OS) .................. 147
Online References .......................................... 147
Finding Symbol Definitions ............................... 148
WinHelp (Windows) .......................................... 149
QuickView (Mac OS) ......................................... 150
THINK Reference (Mac OS) ............................... 151
Inserting Routine (Reference) Templates (Mac OS) . 151

### 6 Searching and Replacing Text 153

Searching and Replacing Text Overview ................ 153
Guided Tour of the Find Dialog Box .................. 153
Search and Replace Section ............................. 154
Multi-File Search Section ............................... 160
Searching for Selected Text ............................ 164
Searching and Replacing Text in a Single File ...... 166
Finding Search Text ....................................... 167
Controlling Search Range ............................... 168
# Table of Contents

Controlling Search Parameters ........................................... 169
Searching with Special Characters ...................................... 169
Replacing Found Text ....................................................... 170
Using Batch Searches ..................................................... 172
Searching and Replacing Text in Multiple Files ..................... 173
Activating Multi-File Search .............................................. 174
Choosing Files to be Searched .......................................... 175
Saving a File Set .......................................................... 179
Removing a File Set ....................................................... 180
Controlling Search Range ................................................ 181
Using Regular Expressions (grep) ........................................ 182

## 7 Browsing Source Code ................................................. 189

Browser Overview ........................................................ 189
Activating the Browser ................................................... 190
Understanding the Browser Strategy .................................... 190
Catalog View .............................................................. 192
Browser View ............................................................. 192
Hierarchy View ............................................................ 194
Guided Tour of the Browser .............................................. 195
Catalog Window ........................................................... 196
Multi-Class Browser Window ............................................ 197
Single-Class Browser Window ........................................... 203
Multi-Class Hierarchy Window .......................................... 206
Single-Class Hierarchy Window ........................................ 208
Symbol Window ........................................................... 209
Context Pop-Up Menu ..................................................... 211
Using the Browser ........................................................ 213
Setting Browser Options .................................................. 213
Identifying Symbols in the Browser Database ....................... 214
Navigating Code in the Browser ......................................... 214
Browsing Across Subprojects .......................................... 216
Completing Symbols ........................................................ 217
Opening a Source File ..................................................... 217
Seeing a Declaration ....................................................... 218
# Table of Contents

- Seeing a Routine Definition .................................... 218
- Editing Code in the Browser .................................... 219
- Analyzing Inheritance ........................................... 219
- Finding Functions That Are Overrides ......................... 219
- Windows MFC Class Viewing .................................... 220
- Mac OS PowerPlant Class Viewing ............................... 220
- Saving a Default Browser ....................................... 221

## 8 Configuring IDE Options 223

- Configuring IDE Overview ....................................... 223
- Preferences Guided Tour ......................................... 224
  - Preference Panels ............................................... 224
  - Dialog Box Buttons ............................................ 225
- Choosing Preferences ............................................ 226
- Debugger Preferences ............................................ 227
  - Display Settings Panel ........................................ 227
  - Global Settings Panel ......................................... 229
- MetroNub Settings Panel ........................................ 231
- Editor Preferences ............................................... 233
  - Browser Coloring ............................................... 233
  - Editor Settings ................................................. 234
  - Font and Tabs .................................................... 239
  - Syntax Coloring ................................................ 240
- General Preferences .............................................. 244
  - Build Settings Panel .......................................... 244
  - IDE Extras Panel ............................................... 246
  - Key Bindings Panel ............................................. 250
- Customizing Toolbars ........................................... 262
  - Kinds of Toolbars ............................................... 263
  - Toolbar Elements .............................................. 264
  - Showing and Hiding a Toolbar ................................ 265
  - Modifying a Toolbar ............................................ 267
  - Adding a Toolbar Element ..................................... 267
  - Rearranging Toolbar Elements ................................ 268
  - Removing a Toolbar Element .................................. 268
# Table of Contents

Removing All Toolbar Elements ........................................... 268
Restoring a Toolbar to Default Settings ................................. 269
Anchoring the Floating Toolbar (Mac OS) ............................... 269
To release the floating toolbar when it is currently anchored, choose the Unanchor Floating Toolbar command. ....................... 270

## 9 Configuring Target Options ........................................... 271
Configuring Target Options Overview ................................... 271
Target Settings Guided Tour ............................................. 272
  Panels ............................................................... 273
  Dialog Box Buttons ................................................ 273
Choosing Target Settings ................................................ 275
Target Configurations ................................................... 278
  Target Settings ..................................................... 278
  Access Paths ........................................................ 280
  Build Extras ........................................................ 288
  File Mappings ....................................................... 290
Editor Configurations ................................................... 292
  Custom Keywords ................................................... 292
  Global Optimizations .............................................. 293

## 10 Compiling and Linking .............................................. 297
Compiling and Linking Overview ....................................... 297
Choosing a Compiler ..................................................... 298
  Understanding Plug-in Compilers .................................. 298
  Setting a File Extension ........................................... 299
Compiling and Linking a Project ....................................... 299
  Compiling Files ..................................................... 300
  Setting Link Order .................................................. 301
  Updating a Project .................................................. 302
  Making a Project ..................................................... 303
  Enabling Debugging .................................................. 304
  Running a Project ................................................... 304
  Debugging a Project ................................................ 305
  Generating a Link Map .............................................. 306
  Synchronizing Modification Dates .................................. 306
Table of Contents

Removing Objects ............................................. 306
Advanced Compile Options ................................. 308
Using Precompiled or Preprocessed Headers ........... 308
  Creating Precompiled Headers......................... 309
  Defining Symbols For C/C++......................... 312
  Defining Symbols For Pascal......................... 313
Preprocessing Source Code ............................... 315
Disassembling Source Code ................................ 315
Guided Tour of the Message Window .................... 316
Using the Message Window .................................. 319
  Seeing Errors and Warnings.......................... 320
  Stepping Through Messages........................... 321
  Correcting Compiler Errors and Warnings ........... 321
  Correcting Linker Errors .............................. 323
  Correcting Pascal Circular References .............. 326
  Saving and Printing the Message Window ............ 327
  Locating Errors in Modified Files ................... 328
Mac OS Special Library Options ......................... 328
  Import Weak. .............................................. 330
  Initialize Before ....................................... 330
  Merge into Output ...................................... 330

11 Configuring Version Control Software 331
  Version Control System Overview ..................... 331
  Version Control System (VCS) Setup .................. 332
  Setting up a Project for Revision Control .......... 334
  Using Source Code Control with Files ............... 337
    Determining Version Control Status of a File ....... 338
    Modifying a Read Only Source Code File ............ 340
    Other Revision Control Operations ................. 341
    Viewing a File’s Status in the Project Window .... 343
    Mac OS ‘ckid’ Resources ............................. 343
    Advanced VCS Operations ............................. 344
  Windows VCS Features .................................. 345
  Mac OS VCS Features .................................... 347
# Table of Contents

**VCS Window** ................................................. 349

<table>
<thead>
<tr>
<th>12 IDE Menu Reference</th>
<th>351</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE Menu Reference Overview</td>
<td>351</td>
</tr>
<tr>
<td>File Menu</td>
<td>352</td>
</tr>
<tr>
<td>Edit Menu</td>
<td>356</td>
</tr>
<tr>
<td>Search Menu</td>
<td>360</td>
</tr>
<tr>
<td>Project Menu</td>
<td>366</td>
</tr>
<tr>
<td>Debug Menu</td>
<td>372</td>
</tr>
<tr>
<td>Data Menu</td>
<td>373</td>
</tr>
<tr>
<td>Window Menu</td>
<td>376</td>
</tr>
<tr>
<td>Version Control System (VCS) Menu</td>
<td>380</td>
</tr>
<tr>
<td>Help Menu</td>
<td>382</td>
</tr>
<tr>
<td>Windows Help Menu</td>
<td>382</td>
</tr>
<tr>
<td>Mac OS Help Menu</td>
<td>382</td>
</tr>
<tr>
<td>Toolbar Submenu</td>
<td>383</td>
</tr>
<tr>
<td>Apple Menu (Mac OS)</td>
<td>384</td>
</tr>
<tr>
<td>Tools Menu (Mac OS)</td>
<td>384</td>
</tr>
<tr>
<td>Scripts Menu (Mac OS)</td>
<td>387</td>
</tr>
<tr>
<td>Editor Extensions Menu (Mac OS)</td>
<td>389</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A Default CodeWarrior Key Bindings</th>
<th>391</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac OS Key Binding Legend</td>
<td>391</td>
</tr>
<tr>
<td>File Menu</td>
<td>392</td>
</tr>
<tr>
<td>Edit Menu</td>
<td>393</td>
</tr>
<tr>
<td>Search Menu</td>
<td>394</td>
</tr>
<tr>
<td>Project Menu</td>
<td>395</td>
</tr>
<tr>
<td>Debug.</td>
<td>397</td>
</tr>
<tr>
<td>Data</td>
<td>397</td>
</tr>
<tr>
<td>Window Menu</td>
<td>398</td>
</tr>
<tr>
<td>Misc.</td>
<td>401</td>
</tr>
<tr>
<td>Editor Commands</td>
<td>401</td>
</tr>
<tr>
<td>Prefix Keys</td>
<td>404</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B Mac OS CodeWarrior Scripting</th>
<th>405</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeWarrior Apple Events Overview</td>
<td>405</td>
</tr>
</tbody>
</table>
# Table of Contents

AppleScript Tools and Reference Material ........................................... 406
Writing Your First CodeWarrior IDE AppleScript ............................... 407
CodeWarrior IDE AppleScript Events .................................................. 408
  Processing Errors ........................................................................... 409
  Required Events ............................................................................. 411
  File Handling Events ................................................................. 412
  Building Events ............................................................................ 418
  Status/Query Events ...................................................................... 427
  Navigation Events .......................................................................... 435
CodeWarrior IDE AppleScript Classes ................................................ 437
  Project Classes ............................................................................. 438
  Compiler Classes ........................................................................... 449
  CodeGen Classes ............................................................................ 455
  Disassembler Classes ................................................................. 458
  Linker Classes ............................................................................... 459
  Build Classes ................................................................................ 465
  Browser Classes ............................................................................. 467
  Editor Classes ............................................................................... 468
  Object Classes ............................................................................... 475
  Misc Classes .................................................................................. 478
Coding with CodeWarrior IDE and Apple Events ................................ 484

C Mac OS CodeWarrior and MPW ToolServer ..................................... 487
  Using MPW ToolServer Overview .................................................. 487
  About ToolServer .......................................................................... 487
  ToolServer and MPW Shell Differences .......................................... 489
  Installing ToolServer ..................................................................... 489
  Starting ToolServer ....................................................................... 491
  Executing MPW Commands .......................................................... 493
    Using the ToolServer WorkSheet ............................................... 494
    Using the ToolServer Tools Menu .............................................. 495
    Using a Commando Dialog Box .................................................. 497
  Looking Up MPW 411 Documentation ............................................ 502
  Creating MPW Tool Projects ......................................................... 504
Introduction

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

NOTE: On occasion a CodeWarrior product ships with an earlier version of the IDE than reflected in this user guide. In that case, your IDE will not have the new features described in this manual. You can identify new features by referring to “What’s New in This Release” on page 24.

In some cases a patch may become available to update the tools. You can visit the Metrowerks website at http://www.metrowerks.com for information on available patches.

Introduction Overview

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

- Read the Release Notes!
- Manual 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 target systems listed in Table 1.1. To learn which manual to read for the target that you are interested in, refer to Table 1.2.
You use the same IDE when developing code for all these systems.

The version of CodeWarrior you have contains some or all of these programming languages for developing applications:

- C and C++, a compiler that implements templates, exception handling, run-time type information (RTTI), and inline assembly code.
- Java, for Java virtual machines.

### Table 1.1 CodeWarrior IDE Targets

<table>
<thead>
<tr>
<th>Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeOS</td>
<td>The Be operating system</td>
</tr>
<tr>
<td>Embedded PowerPC</td>
<td>Various embedded PowerPC processors</td>
</tr>
<tr>
<td>Java</td>
<td>The Java virtual machine</td>
</tr>
<tr>
<td>Mac OS</td>
<td>The Mac OS running on either the 680x0 (68K) or PowerPC family of microprocessors</td>
</tr>
<tr>
<td>Nucleus</td>
<td>A real-time, pre-emptive, multi-tasking kernal for embedded systems</td>
</tr>
<tr>
<td>OS-9</td>
<td>A real-time operating system for software running on embedded systems</td>
</tr>
<tr>
<td>PalmPilot OS</td>
<td>The operating system for the PalmPilot connected organizer</td>
</tr>
<tr>
<td>PlayStation</td>
<td>The Sony PlayStation game console</td>
</tr>
<tr>
<td>Win32/x86</td>
<td>The Win32 model for Windows NT and Windows 95 on Intel 80x86-class and Pentium-class processors</td>
</tr>
</tbody>
</table>

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

Read the Release Notes!

Object Pascal, a compiler for ANS Pascal and Object Pascal, which provides Turbo Pascal Input/Output routines, conditional compilation and macros, extended debugging features, and inline assembly code.

Assembly Language, a low-level machine code language.

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.

Manual Conventions

This section describes the different conventions used in this manual.

Typographical Conventions

This manual uses some style conventions to make it easier to read and use:

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!
Introduction

Manual Conventions

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 appears in a different typeface (as the word different does in this sentence), you are reading file or folder names, source code, keyboard input, or programming items.

Text formatted like this means that the text refers to an item on the screen, such as a menu command or control in a dialog box.

If you are using an on-line viewing application that supports hypertext navigation, such as Adobe Acrobat, text formatted like this means you can click on it to view another topic or related information. For example, click this text, “IDE User Guide Overview” on page 20, to jump to a section that describes the rest of this manual.

Host Conventions

CodeWarrior runs on several different host platforms, each of which may have several versions of its operating system. For purposes of this manual, a generic platform identifier is used to identify the host platform, regardless of operating system.

The specific versions of the operating system that host CodeWarrior are:

- Windows—desktop versions of the Windows operating system that are Win32 compliant, such as Windows 95 or Windows NT.
- Mac OS—desktop versions of Mac OS, System 7.1 or later.

Figure Conventions

CodeWarrior runs on several different host platforms. The visual interface on these hosts is nearly identical in all significant respects. When discussing a particular interface element such as a dialog box or window, the screenshot may come from any host. If you are using CodeWarrior on a different host, you should have no difficulty understanding the picture.
However, there are occasions when dialog boxes or windows are unique to a particular host. In that case, a screenshot from each unique host will appear and be clearly identified so that you can see how CodeWarrior works on your preferred host.

**Keyboard Conventions**

The default keyboard shortcuts for CodeWarrior on some platforms are very similar. However, keyboards and shortcuts do vary across host platforms. For example, a typical keyboard for a Windows machine has an Alt key, but a typical keyboard for a Mac OS computer calls the same key the Option key.

To handle these kinds of situations, CodeWarrior documentation identifies and uses the following paired terms in the text:

- **Enter/Return**—the “carriage return” or “end of line” key. This is not the numeric keypad Enter key, although in almost all cases that works the same way.

- **Backspace/Delete**—the Windows Backspace key and the Mac OS Delete key. In most cases, CodeWarrior maps these keys the same way. This is the key that (in text editing) causes the character before the insertion point to be erased. (This is not the Delete/Del, the “forward delete” key.

- **Ctrl/Command**—the Windows Ctrl (control) key and the Mac OS Command key. In most cases, CodeWarrior maps these keys the same way. (This is not the Mac OS keyboard Control key.)

- **Alt/Option**—the Windows Alt key and the Mac OS Option key. In most cases, CodeWarrior maps these keys the same way.

For example, you may encounter instructions such as “Press Ctrl/Cmd+S to save a file,” or “Alt/Option click the Function pop-up menu to see the functions in alphabetical order.” Use the appropriate key as it is labeled on your keyboard.

Some combinations of key strokes require multiple modifier keys. In those cases, key combinations are shown connected with hyphens. For example, Ctrl-Alt/Option-Shift-Command.
In certain cases the cross-platform variation in keyboard shortcuts is more complex. In those cases, you will see more detailed instructions on how to use a keyboard shortcut for your host platform. In all cases the host and shortcut will be clearly identified.

**IDE User Guide Overview**

There are several chapters in this User Guide to explain how to use the IDE. Each chapter begins with an overview of the topics discussed in that chapter. The chapter overviews are:

- **Introduction Overview** — (this chapter) an overview of the CodeWarrior IDE languages, targets, hosts, and documentation
- **Getting Started Overview** — system requirements, installation, guided tour of the user interface
- **Working with Projects Overview** — creating, configuring, and working with projects
- **Working with Files Overview** — opening, saving, backing up, comparing, and printing files
- **Source Code Editor Overview** — editing and navigating text and source code
- **Searching and Replacing Text Overview** — finding and replacing text
- **Browser Overview** — analyzing and navigating through a project from various views
- **Configuring IDE Overview** — customizing the IDE with the Preferences dialog box and using toolbars
- **Configuring Target Options Overview** — setting up and customizing a project and its targets
- **Compiling and Linking Overview** — compiling, linking, running, updating, preprocessing, and precompiling a project’s target and its files.
- **Version Control System Overview** — using revision control systems with the CodeWarrior IDE
- **IDE Menu Reference Overview** — lists menus and menu commands
Introduction

About the CodeWarrior IDE

- **Default CodeWarrior Key Bindings**—lists all the default key bindings used by the CodeWarrior IDE
- **CodeWarrior Apple Events Overview**—lists Apple Event and AppleScript support provided by the CodeWarrior IDE
- **Using MPW ToolServer Overview**—using Apple’s ToolServer with the CodeWarrior IDE

About the CodeWarrior IDE

The CodeWarrior IDE is an application that provides a simple, versatile graphical user interface and tools for developing computer software 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 wide variety of computer systems.

The CodeWarrior IDE permits a software developer to quickly assemble source code files (for example, a file written in the C++ computer language), resource files, library files, other project files, and configuration settings into a project, without writing a complicated build script (or “make file”) for the project. Source code files may be added or deleted from a project using simple mouse and keyboard operations instead of tediously editing a build script.

More than one configuration of files and settings may be added to a project. Such a configuration is called a target. Targets in a project may share files in the project while using their own settings. For example, a project may contain a debugging target which contains source code files, extra diagnostic source code files, and settings to specify the generation of debugger information. Another target in the same project may use the some of the same files as the debugging target, but uses settings that specify the use of compiler optimizations and no debugging information. After debugging a program, generating a final version is as simple as changing a project’s target and using a single **Make** command.

Tools such as a debugger, compilers, linkers, a source code browser and editor are included with the CodeWarrior IDE. These tools allow you to edit your code, navigate and examine your code, com-
Where to Go From Here

If you are an experienced CodeWarrior IDE user, review “What’s New in This Release” on page 24 for an overview of the new features.

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

When you are ready to debug, be sure to read the CodeWarrior Debugger User Guide.

If you are trying to get started quickly with a new platform or if you are new to CodeWarrior, see “QuickStart and Tutorials” on page 22.

To get started quickly with a new target operating system, see “Targeting Documentation” on page 23.

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

- QuickStart and Tutorials
- Targeting Documentation

QuickStart and Tutorials

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 CodeWarrior and pointers to the references available to you.
The *QuickStart Guide* is on your CD, so you can use it regardless of whether you purchased any printed documentation.

- “An Introduction to the CodeWarrior IDE” on page 32 provides a quick overview of the CodeWarrior IDE user interface.
- The CW Core Tutorials 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.

**Targeting Documentation**

This manual describes how to use the CodeWarrior IDE; it doesn’t show you how to use CodeWarrior to develop software for a particular “target.”

A target is an operating system or processor that you develop software for using the CodeWarrior IDE. This means that you choose a target operating system or processor on which you want your finished code to run. Many different targets are supported by this and other CodeWarrior products. All targets supported by CodeWarrior are developed using the CodeWarrior IDE.

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

**NOTE:** The CodeWarrior product you’re using may not generate software for all the targets listed in Table 1.2.

<table>
<thead>
<tr>
<th>Target</th>
<th>Targeting Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeOS</td>
<td>Targeting BeOS</td>
</tr>
<tr>
<td>Java Virtual Machine</td>
<td>Targeting the Java VM</td>
</tr>
</tbody>
</table>
Introduction
What’s New in This Release

What’s New in This Release

The Metrowerks IDE is now at version 2.2. Among its new features, the following are the most notable:

- Integrated Debugger
- File Compare and Merge Tool
- Project Manager Improvements
- Editor Improvements
- Version Control Support (Windows)
- Online Help Support (Windows)
- Native Plug-ins Support (Windows)
Introduction
What’s New in This Release

Integrated Debugger

The CodeWarrior debugger is now integrated into the development environment. This release supports all desktop targets, so you can debug C/C++, Java, and Pascal programs on Windows and Mac OS. Support for all CodeWarrior targets will be added in future releases.

For more information, see “Project Debugging” on page 35, “Debugger Preferences” on page 227, “Global Settings Panel” on page 229.

File Compare and Merge Tool

File comparison has been extended to support comparing folder contents and to allow for the easy comparison of open editor windows.

For more information, see “Comparing and Merging Files & Folders” on page 113

Project Manager Improvements

Modifications to the opening of project files has decreased the amount of time required to open a project file.

Editor Improvements

Windows Key bindings now fully support the use of the Alt and Windows keys. In addition, Intellimouse support has been added for scrolling in editor windows.

For more information, see “Key Bindings Panel” on page 250

Version Control Support (Windows)

The CodeWarrior IDE now has integrated support for the Microsoft Visual SourceSafe version control software. This support allows get, check-out, check-in, and other common VCS actions on single or multiple files.
For more information, see “Configuring Version Control Software” on page 331.

**Online Help Support (Windows)**

Online help is available for error messages and many CodeWarrior IDE window components. Just Right-click on an element in a window to have a pop-up help item appear. Not all IDE windows are fully supported.

For more information, see “WinHelp (Windows)” on page 149

**Native Plug-ins Support (Windows)**

The creation of native Window plug-ins is now fully supported.

For more information, see the Plugin API Manual.
Getting Started

This chapter helps you get started using the CodeWarrior IDE (Integrated Development Environment). In it you’ll find the system requirements and information about installing IDE software and an introduction to the IDE’s user interface and capabilities.

Getting Started Overview

This chapter shows you how to get started using the CodeWarrior IDE.

The sections in this chapter are:

- System Requirements
- CodeWarrior IDE Installation
- Programming Concepts
- An Introduction to the CodeWarrior IDE

NOTE: This manual describes how to use the CodeWarrior IDE; it doesn’t show you how to use CodeWarrior to develop software for a particular processor or operating system “target.” For information on developing software for a target, see “Targeting Documentation” on page 23.

TIP: For a quick look at the IDE’s features, see “An Introduction to the CodeWarrior IDE” on page 32. The tour gives you your first glimpse of the CodeWarrior IDE.
System Requirements

The system requirements to operate the CodeWarrior IDE are specified by platform. These platforms include:

- Windows Requirements
- Mac OS Requirements

Windows Requirements

The Windows-hosted version of CodeWarrior requires a 80486DX or Pentium™-class processor, 24 megabytes of RAM, Microsoft Windows 95 or Windows NT 4.0 operating system or later versions of these operating systems, and a CD-ROM drive to install the software.

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

Mac OS Requirements

The Mac OS-hosted version of the CodeWarrior IDE requires a Motorola MC68040 or greater processor, or PowerPC 601 or greater processor, 24 megabytes of free RAM, Color QuickDraw, Mac OS System 7.5 or later, and a CD-ROM drive to install the software. When running on 68k machines, CFM 68K Enabler 4.0 is required (provided by the installer).

For the best performance, we recommend that you use a Mac OS computer equipped PowerPC microprocessor, with at least 32 megabytes of free RAM.

CodeWarrior IDE Installation

To learn how to install the CodeWarrior IDE, read the QuickStart guide. The CodeWarrior installation software places the CodeWarrior IDE, compilers, linkers, tools, and debuggers on your hard drive.
Programming Concepts

If you are new to programming computers, read this section to learn about the terms used in this manual.

To develop a piece of software you’ll do these tasks:

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

First, you create files that contain statements using a computer language such as C, C++, Java, Pascal, or assembly language. You might also create files that contain resources, which are descriptions of user interface objects such as windows, dialog boxes, and menus.

Next, you apply tools to help turn your “source” materials into an output file that executes on your target computer. These tools include compilers, linkers, other tools, and maybe even assembly-language assemblers. You’ll use these tools to create many different types of software, depending on the tools used and the target computer. Some of the types of software you might create include applications (or executables), dynamic (shared) libraries, and static libraries.

Once you have created a piece of software, you can use a debugger to examine the software as it runs to ensure it behaves correctly. If it doesn’t, you return to the first step to 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 and use to make a program are source code files, resource files, interface or header files, library files, and project files.
**Source code file**

A source code file is a text file containing program statements written in a language such as C, C++, Java, Pascal, or assembly language.

**Resource file**

A resource file contains descriptions of user interface items, such as window definitions, dialog box layouts, and text strings. A resource file may be a binary file linked into your software product, or it may be a text file translated by a special resource compiler before being linked. Placing resources in separate files makes it convenient to tweak and customize these items without having to recompile other parts of a program.

**Interface or header file**

An interface file, also called a header or include file, is a text file referred to by source code files. Typically, these files give access to objects, variables, data structures, routines, and other items in libraries or source code files.

**Library file**

A library file contains objects, variables, routines, and other items that have already been compiled. There are two kinds of libraries: static and dynamic.

The IDE builds a static library right into a program. A static library cannot change (hence the term “static”) unless you rebuild the program. Static libraries can’t be shared with other programs.

Although a program might refer to a dynamic library, the IDE doesn’t build it into a program. Instead, the program hooks up with the library dynamically, that is, while the program is actually running. Often, more than one program shares the same dynamic library. Also, a dynamic library may be replaced with a newer library without affecting the programs that refer to it as long as the newer library provides the same items as its predecessor.
Most processor and operating system targets support static libraries, but not all targets support dynamic libraries. For information on creating and using libraries for a specific target, see “Targeting Documentation” on page 23.

**Project file**

A project file is a file that contains one or more targets. Each target contains a list of source code, resource, interface, library files, and even other projects files that are used to create your software. Each target also contains its own settings that instruct the IDE how your software should be created.

**Generating the Software**

The types of tools the IDE uses to build your software include compilers, assemblers, and linkers. Using information from files and settings in a project’s target, the IDE automatically determines which tools to use to create your software.

**Compilers**

A compiler translates a source code file, such as a C, C++, Pascal, or Java file into binary machine code (also called object code) that will be used by linkers in a later build stage. A compiler is one of the first tools that the IDE invokes to build your program.

**Assemblers**

An assembler translates an assembly language source code file into object code. An assembler is really just compiler, only it translates assembly language source code rather than high-level language source code.

**Linkers**

A linker combines the object code in your project’s target produced by the compilers and assemblers into a piece of software.
Output file

The output file is the piece of software that the linker generates. There are many different types of software, such as applications and libraries. For in-depth information about the kinds of software to develop for a target, see the CodeWarrior Targeting manual appropriate for your platform. For information on which targeting manual to read, see “Targeting Guides for Various CodeWarrior Targets” on page 23.

Debugging and Refining

A debugger is a tool that controls and monitors the execution of your program. With it, you can stop at points in your program’s execution and see what the contents of variables are. You can also execute one line of code at a time.

All you need to do to enable the debugger is produce special data that the debugger uses to control your program. The IDE creates debugging information or symbolics files, which contain the information the debugger needs to display and control statements, variables, objects, and data types in your source code.

The IDE includes an integrated debugger that makes the debugging process even more transparent. In addition, support has been added for third party debuggers on some platforms. See the Targeting manual associated with your platform for additional information.

For information on using the CodeWarrior Debugger, see the CodeWarrior Debugger User Guide.

An Introduction to the CodeWarrior IDE

This section describes the tasks and operations you do with the CodeWarrior IDE. The topics in this section are:

- Projects and Targets
- Source Code Editing and Browsing
- Compiling and Linking
Projects and Targets

The IDE uses projects and targets to organize the files and settings used to create a program. A project is a file that contains one or more targets. A target is a collection of source code files, resource files, libraries, even other projects, and settings that describe how to create a piece of software for a particular processor or operating system. Targets within a project may share the same files, but each target has its own settings.

The CodeWarrior IDE also has preconfigured stationery projects. Creating a new project is as easy as deciding on a target platform to develop for and a programming language to use, then choosing the corresponding stationery project.

Through a target’s settings dialog box, you may set options to choose the target processor or operating system to generate software for, customize compiler optimizations and other object code details, customize source code translation, specify the kinds of files that may be added to a target, and configure other settings.

For more information on working with projects and targets, see “Working with Projects Overview” on page 39 and “Configuring Target Options Overview” on page 271.

For information on developing software for a specific processor or operating system, see “Targeting Documentation” on page 23.
Source Code Editing and Browsing

The IDE has a powerful, flexible text editor for editing source code and text files. Besides regular text editing features, the IDE also provides drag and drop editing, the ability to open, edit, and save text files in Mac OS, Unix, and MS-DOS formats, and setting and jumping to markers in a text file.

The editor also has many advanced features for programmers. Some of its programming aids are: auto-indenting, syntax highlighting, routine and interface pop-up menus, and automatic balancing for braces, brackets, and parentheses. The editor works closely with the class browser to make editing, viewing, and navigating among routines, data structures, variables, and objects intuitive and quick.

The Find dialog box and the Search Menu have commands and features to search and replace text in one file or in a group of files. Text to search for and replace may be normal text or regular expressions.

A file comparison and merging command displays two text files side-by-side and lets you easily apply differences between the two files. In addition, you can compare the files in two folder to look for differences in the files.

For more information on working with text and source code files, see “Working with Files Overview” on page 95, “Source Code Editor Overview” on page 121, “Searching and Replacing Text Overview” on page 153, “Browser Overview” on page 189.

Compiling and Linking

The IDE has commands to preprocess, precompile, compile, update, link, run, run from a debugger, disassemble, and check syntax. The IDE automatically chooses the appropriate compilers and linkers to use and automatically determines which files to operate on when you issue a compile or link command. The IDE uses the settings in a project’s target to instruct its compilers and linkers how to process a target’s files and data. All you have to do is use one of the commands in the Project Menu.
For more information on setting options, compiling, linking, and other software generation operations, see “Configuring Target Options Overview” on page 271 and “Compiling and Linking Overview” on page 297.

Project Debugging

The IDE now has the Metrowerks Debugger integrated into itself to provide a seamless interaction between the programming and debugging of your source code. Some of the benefits that you will experience include:

- Reduced memory requirements. With only one application running, memory demands are significantly reduced.
- Increased productivity. Since you don’t have to switch back and forth between the IDE and Debugger to step through your code, set breakpoints, etc., a more efficient use of time is achieved, increasing your productivity.
- The integrated debugger fully supports x86, PowerPC, 68K, and Java debugging. No longer are separate debuggers required to debug each platform, the integrated debugger handles them all.

Once you enable the debugger, you simply Run the project to activate and use the integrated debugger. You can pause the program at any time to set breakpoints, view variables or memory, step into or out of routines, as well as perform many other debugging tasks.

NOTE: Some versions of the CodeWarrior IDE do not ship with the integrated debugger. In those cases, debugging support is provided by the external MW Debug application or other third party debuggers. See your platform’s Targeting manual for additional information on debugging a specific target.

Version Control

The IDE can be configured to work with Version Control Systems (VCS). The IDE will log onto a file server, retrieve files, store files,
and do other revision control tasks with a VCS software package that supports the CodeWarrior IDE.

For information on using the IDE with VCS software, see “VCS Pop-up Menu” on page 127 and “Version Control System Overview” on page 331.

**Customizing the IDE**

The IDE has many user-configurable options to adapt the IDE to the way you work. Use the IDE’s Preferences dialog box to customize features like set the colors and fonts used to view and edit source code and assign keyboard shortcuts for commands. The IDE also has conveniently placed, easily customizable toolbars that give quick access to commands and information by simply clicking a button.

For more information on customizing the IDE, see “Configuring IDE Overview” on page 223.
Mac OS External Tool Support

The IDE works with other tools like 3rd party text editors, debuggers, and Apple’s ToolServer on the Mac OS.

For information on using CodeWarrior with other software development tools, see “IDE Extras Panel” on page 246, “Debugging a Project” on page 305, “Version Control System Overview” on page 331, and “Using MPW ToolServer Overview” on page 487.

Mac OS Scripting

The IDE supports Apple Events and AppleScripting on the Mac OS. Repetitive, time consuming, or complex tasks can be automated through AppleScript scripts. Use the Scripts Menu (Mac OS) to list and run scripts for the IDE.

For more information on scripting the IDE, see “CodeWarrior Apple Events Overview” on page 405 and “Scripts Menu (Mac OS)” on page 387.
Getting Started

An Introduction to the CodeWarrior IDE
Working with Projects

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

Working with Projects Overview

A project contains one or more targets. Each target in a project contains a collection of files that the IDE uses to build an output file. Targets within a project may share some or all of their files. Some examples of an output file include an application, static library, or dynamic library.

Each target within a project has its own options that customize how the IDE builds the target’s output file. There are a wide variety of options that control code optimization, the browser, compiler warnings, and much more.

Finally, targets within a project can be configured to depend on other targets in the project. This feature makes it possible to build software that, for example, combines the output files for different target processors into a single output file.

This chapter discusses many of the basic tasks involving projects, such as creating, opening, adding files, and saving projects. It also describes advanced 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:

- Guided Tour of the Project Window
- Creating a New Project
- Working with Project Stationery
- Opening an Existing Project
- Saving a Project
- Closing a Project
- Choosing a Default Project
- Managing Files in a Project
- Working with Complex Projects
- Moving a Project
- Controlling Debugging in a Project

Guided Tour of the Project Window

The Project window shows information about the files and targets in a project file in three different views: the File view, Link Order view (sometimes called Segments), and Targets view. To choose a view, click its tab at the top of the Project window, as shown in Figure 3.1.

**Figure 3.1** The view tabs at the top of the Project window

The File view shows a list of all the files in a project. Items in this view may be organized into hierarchical groups that you create and arrange.

The Link Order view (Segment) shows information about how the IDE will compile or link the final output file for the project’s current target.
The Targets view shows information about the active target, target dependencies, and which targets to link with.

The topics that explain the Project window in detail are:

- Navigating the Project Window
- Project Window Toolbar
- File View
- Link Order View
- Targets View

To learn more about debugging information, see “Controlling Debugging in a Project” on page 91.

Navigating the Project Window

To navigate the Project window, use the vertical 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 66.

Project Window Toolbar

The toolbar in the Project window has buttons and other items to provide shortcuts to commands and information about the project. You may configure which items are on the toolbar, and the order in which they are displayed. You can even choose to hide or display the toolbar. To learn more about toolbars in the CodeWarrior IDE and how to configure them, refer to “Customizing Toolbars” on page 262.
File View

The Project window’s File view shows the files for all targets in the project. The files may be arranged hierarchically into groups without affecting the way the IDE builds a target. This view also shows information about file access paths, data size, code size, debugging, target, modification status, and other information.

The topics describe the parts of the File view. These topics are:

- File column
- Code column
- Data column
- Debug column
- Target column
- Touch column
- Interface pop-up
- File Control pop-up
- Checkout Status column
- Project Checkout Status icon

File column

Lists files in the project in a user-configurable hierarchical view. The file column lists the files and groups in a project. A group may contain files and other groups.

Double-clicking a file’s name in the File column will open it if the file can be opened. For information on opening files from the File Column, refer to “Opening Files from the Project Window” on page 98.

Use the expand and collapse controls to view and hide the contents of groups in the File column.

Mac OS To see the path along which the IDE accesses a file, Control-click the file’s name in the project window, then select File Path from the pop-up menu.
Figure 3.2 The Project window

Code column

Shows the size, in bytes or kilobytes, of the compiled executable object code for files and groups. If zero is displayed, it means that your file has not yet been compiled. If “n/a” is displayed in a file’s Code column, the file has no executable code that was compiled by the IDE.

The values in this column do not reflect the amount of object code that will be added to the final output file. The linker may not use all of a file’s object code when creating the final output. Instead, the linker may ignore data and code that are not referenced by other files in the project (“dead stripping”).

For more information on how the linker works, see “Compiling and Linking a Project” on page 299.
Data column

Shows the size, in bytes or kilobytes, of non-executable data in the object code for files in the project. A zero means that your file has not yet been compiled, or that the file does not contain a data section in its object code. If “n/a” is displayed in a file’s Data column, the file has no object code data.

As in the Code column, the data values do not reflect the amount of data that will be added to the final output file.

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 299.

Debug column

Indicates whether the IDE should instruct its compilers and linkers to generate debugging information for files in the project. A black marker in this column means that the IDE will generate debugging information for the corresponding item. A gray marker indicates that only some of the files in that group have generate debug info enabled.

To generate debugging info for a:

- **File**—click in the Debug column next to the file.
- **Group**—click in the Debug column next to the group.
- **Project**—Shift/Option click in the Debug column.

For more information about debugging information for a file, see “Activating Debugging for a File” on page 91.

Target column

Indicates whether or not an item is in the project’s current target. The IDE displays this column if a project has more than one target. A dark marker in this column means that the corresponding item is in the current target. A gray marker indicates that only some of the files in that group are in the current target.
To assign or unassign a current target for a:

- **File**—click in the Target column next to the file.
- **Group**—click in the Target column next to the group.
- **Project**—Shift/Option click in the Target column.

For information on adding or removing a file to or from a target using the Target column, see “Assigning Files to Targets” on page 86.

**Touch column**

Indicates whether a file needs to be compiled the next time a target is built. A marker in this column means that the corresponding item will be re-compiled for the next Bring Up To Date, Make, Run, or Debug command. A gray marker indicates that only some of the files in that group are set for re-compiling.

To indicate re-compiling or disable re-compiling for a:

- **File**—click in the Touch column next to the file.
- **Group**—click in the Touch column next to the group.
- **Project**—Shift/Option click in the Touch column.

To learn more about this feature, see “Synchronizing modification dates” on page 78.

**Interface pop-up**

Lists and opens interface and header files for your project source files. The Interface File Pop-up also allows you to touch or untouch its corresponding item and set other options that appear depending on the current target.

For groups, this pop-up menu lists the files within the group. Choosing a file in the popup will open the file.

For more information about opening interface and header files, see “Interfaces pop-up menu” on page 99.
Working with Projects
Guided Tour of the Project Window

File Control pop-up

Use Right-click/Control-click on an item’s icon in the Project window to view the File Control pop-up menu, as shown in Figure 3.3. From this menu, choose a command to operate on the item.

Figure 3.3 File Control pop-up menu in the project window

Checkout Status column

Check files in or out of a version control system (VCS). Use this column to track changes to your code, particularly when more than one person is working on your software project. The Checkout Status Column, shown in Figure 3.2 on page 43, only appears when you have configured your CodeWarrior project to use a source code revision control system, such as Metrowerks Visual SourceSafe for Macintosh (formerly Metrowerks CodeManager) or Microsoft Visual SourceSafe (for Microsoft Windows).

To learn more about using any other revision control systems, refer to the documentation that came with it.

Project Checkout Status icon

Shows whether the project is writable or not, or what file access permissions are set for the project. A revision control system, such as Metrowerks Visual SourceSafe for Macintosh or Microsoft Visual SourceSafe (for Microsoft Windows), assigns these permissions when you check a project file in or out of the revision control system.

To learn more about what the Project Checkout Status icon means, and how the icon relates to the access permissions assigned to your source code files, refer to “Determining Version Control Status of a File” on page 338.
To learn more about using your revision control system, refer to the documentation that came with it.

**Link Order View**

By default, the CodeWarrior IDE compiles files in the order that appears in the File view, but will fail if one file depends on information from a second file that has not been compiled yet. By putting the files in a correct order in the Link Order view the user can avoid this problem.

Items in the Link Order 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.

For more information, see “Setting Link Order” on page 301.

For more discussion about groups, refer to “Managing Files in a Project” on page 63.

**Mac OS** The Link Order view may be named Segments for some targets. Refer to the targeting documentation for more information (“Targeting Documentation” on page 23).

**Targets View**

The Targets view shows a list of the targets in the project. This view also shows the objects that the targets depend on to create a final output file. shows an example Targets view.
Creating a New Project

There are a few short steps involved in creating 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**.

The CodeWarrior IDE displays the **New Project** dialog box, shown in **Figure 3.5**, from which you choose your project stationery.
TIP: You can create a new empty project not based on project stationary using the Control-Alt/Option-Shift-Command N. This option lets you create a project from scratch. This is not recommended, particularly for beginners, because of the complexities of including the correct libraries and files and choosing the correct settings.

Use the navigation controls to select the kind of project that you are interested in. To create an empty project that contains no libraries or other support files, you may choose the Empty Project option from the list.

To create a new folder containing all the new project files, make sure the Create Folder checkbox is selected.
Naming Your New Project

After choosing New Project from the File Menu and selecting an item in the New Project window, click the OK button and a dialog box appears asking for a project name, as shown in Figure 3.6. 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 file name extension. This makes your project easier to visually identify on your hard disk. In addition, the IDE uses this extension to quickly identify the project file when the project is moved across platforms.

Figure 3.6 Naming a new project (Windows)
When you click Save, the CodeWarrior IDE automatically sets up the project, including:

- Creating a project folder with the same name as your project (if you selected the Create Folder checkbox as discussed in “Choosing the New Project Stationery File” on page 48), without the file name 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.
Working with Project Stationery

Modifying Your New Project

Most new projects created from stationery contain source files that are basically empty placeholders. You probably want to delete these files and replace them with source files of your own. See the section “Managing Files in a Project” on page 63 for more information about manipulating files in a 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. Table 1.2 on page 23 lists all the targeting manuals for the CodeWarrior.

Building Your New Project

After you have created your project and added your own files 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 “Compiling and Linking Overview” on page 297.

Working with Project Stationery

This section discusses how to create and use project stationery. It includes an explanation of project stationery that you can use to customize the project creation process described in “Creating a New Project” on page 48.

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

The topics in this section include:

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

A project stationery file is typically a minimal, preconfigured “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 folder for the project. It then copies all the files related to the stationery project to the new folder.

A stationery project includes:

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

When 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.

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.

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

- Normally saved projects
- Project stationery files
- (Mac OS) Aliases to projects
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.

In essence, any CodeWarrior project you have can become a stationery project. To qualify as a stationery project it must: (1) be located in the (Stationery) folder, and (2) have its associated source files stored with it. When your project stationery is chosen in the New Project dialog, CodeWarrior duplicates the project and source files associated with the stationery project using the new project name.

To create your own custom stationery from scratch, start by creating a new project from an already available set of stationery or just create an empty project. Save the new project to the (Stationery) folder. Then, modify the stationery project’s settings to best suit its requirements. Be sure to add and remove files as necessary to create the exact base project you want. Also, make sure a copy of all its source files are present in the new stationery project’s folder, otherwise, the source files will not be copied to any new projects created with that project stationery. Make sure to save all your changes.

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

The next time you want to create a new project, choose your custom project stationery. All of its project and source files are copied to the new project’s location, ready for modification.

To learn how to configure Preferences settings, refer to “Choosing Preferences” on page 226. 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 275.

For information about adding or changing files in the project, see “Managing Files in a Project” on page 63.
See “Backing up files” on page 106 for more information about saving a copy of the project under a different name.

**Figure 3.8  Save A Copy As dialog box (Windows)**

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 ways to 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. See “Choosing a Default Project” on page 63 for more information on default projects.

The current project window you have open can be made the active window using the Window Menu. 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

To open a Project File, choose the Open command from the File Menu. The IDE displays an Open file dialog box, as shown in Figure 4.1.

Windows If not already set, use the Files of Type pop-up menu to select Project Files. The list of files changes to show only project files that are eligible for you to open.

Mac OS The dialog displays a list of the available project and text files in the current folder.

Select the project file you would like to open, then click the Open button. The CodeWarrior IDE then opens the project and displays it in a Project window.

If the project was created with an older version of CodeWarrior, you will be prompted to update the older project. If you decide to update, a backup of the project is created before the updating takes place.

For more information about working with CodeWarrior project files, see “Working with Projects Overview” on page 39.

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 and files 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.

To learn about setting the number of files that the IDE remembers in this menu, see “IDE Extras Panel” on page 246.
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 “Working with Complex Projects” on page 79.

Opening Project Files Created on Other Host Platforms

Project files are cross-platform compatible. That means, for example, that a project created on a Mac OS computer may be opened and used on a Windows computer.

To use a project created on another host platform, copy only its project file, not its associated Data folder, from the other host platform to your computer. After copying the project, open it in the IDE and recompile its files. Although a project’s format is cross-platform compatible, its compiled object code isn’t.

NOTE: Before copying a project make sure the project has a “.mcp” file name extension (without the quotes). The CodeWarrior IDE uses this file name extension to recognize project files. If the three letter extension is not present, the IDE will be unable to identify the project file.

TIP: When creating any project, always add the “.mcp” extension to the project file’s name. This makes it easy to move project files to a different platform.

See also “Options Pop-Up Menu” on page 126 for information on editing source code files created on other platforms, and “Host Flags” on page 285 for information on setting up access paths for a target’s host platform.
Opening Project Files from Earlier IDE Versions

The CodeWarrior IDE version 3.0 cannot use a project file from any version of CodeWarrior earlier than 1.7. You must recreate the project file from scratch using the new IDE, or convert the project file to work with the new IDE.

This section discusses how you can convert projects. For single projects, it is automatic. However, you can convert projects singly or in groups, manually or automatically.

Converting a single 1.7 project

The 3.0 CodeWarrior IDE has the ability to convert project files to the new 3.0 format from the format used by 1.7 versions of the IDE. It uses a conversion utility called Project Converter.

You can convert a single project in either of two ways.

- You can simply open the 1.7 project from the 3.0 IDE. The 3.0 IDE and later versions will automatically run Project Converter to convert the project to the new format.
- Or, you can drag and drop the 1.7.x project file onto Project Converter itself.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>Click Start, click Programs, click Metrowerks CodeWarrior, click Project Converter.</td>
</tr>
<tr>
<td>Mac OS</td>
<td>Located in the Metrowerks:Metrowerks Tools subfolder on the hard drive where CodeWarrior was installed.</td>
</tr>
</tbody>
</table>

Whichever way you choose to convert your project, the name of the converted project file will be based on the name of the original file.
Working with Projects
Opening an Existing Project

NOTE: 1.7.x projects must be converted to the 3.0 format if you want to work with them in the 3.0 IDE, and the conversion is permanent. The 3.0 IDE cannot save a 3.0 project in 1.7.x format.

Converting multiple 1.7 projects

Project Converter has the ability to convert multiple 1.7 project files into a series of separate 3.0 projects, or into one multi-target 3.0 project file.

To take advantage of this, drag and drop a group of 1.7 project files onto Project Converter. You’ll be asked if you want to keep them as separate projects, or merge them into a multi-target 3.0 IDE project. Choose the option you wish.

Project Converter will prompt you to select a folder and name for the project or projects.

If you encounter problems, contact support@metrowerks.com.

Metrowerks recommends reading the Project Converter release notes file before converting your projects. You’ll find this file in the Metrowerks:CWPro Release Notes: IDE Notes folder after CodeWarrior is installed. The release note will give you information on any late-breaking topics.

Opening project files from versions prior to 1.7 (Mac OS)

Project Converter only converts project files from the 1.7 IDE releases with CodeWarrior 10 and CodeWarrior 11. If you have an earlier project, you must convert it to CW10 or CW11 format before using Project Converter.
To update a project to the 1.7 format, first make certain that the 3.0 IDE is not running before you launch the 1.7 IDE. Then launch the 1.7 IDE and open the project.

The 1.7 IDE will update your project. You can then run the updated project through Project Converter to update to 3.0.

If you don’t have version 1.7 of the IDE, you can find it on the CWPro Tools CD. Look for the "CW11 IDE and Prefs.sit" file in the Other Metrowerks Tools folder.

**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 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
Working with Projects
Closing a Project

- 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

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

**WARNING!** Do not attempt to make a copy an open project from the desktop. Always close the project before copying the project file to prevent it from being corrupted.

Closing a Project

After you have been working with your project, you may want to close it to work on another project, or to quit the CodeWarrior IDE application to work on something else.

To close a project, ensure the Project window is frontmost, then choose Close from the File Menu.

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 61.

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.
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.

However, source code files can be in more than one open project. Set which project is the default project for builds using the Set Default Project command on the Project Menu. In any ambiguous case when a source code file belongs 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.

Managing Files in a Project

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

- Expanding and Collapsing Groups
- Selecting Files and Groups
- Adding Files
- Moving Files and Groups
- Creating Groups
- Removing Files and Groups
- Renaming Groups
Expanding and Collapsing Groups

Groups display files in collapsible lists. Click the tree control or disclosure triangle depending upon your platform to display or hide the contents of the group. The Group Display Controls table describes the interface item for displaying hierarchical lists on the various host platforms.

To expand a group and view its files, click the control in the top left edge of the desired group. To close a group and view only its name, click the control again.

Figure 3.10 Expanding groups and subgroups

Use Alt/Option click to expand a group and all its subgroups in the project window as shown in Figure 3.10. It does not expand groups that are at the same level as the chosen group. Alt/Option click again to collapse the group.
Use Control/Command click to expand all groups at the same level in the project window as shown in Figure 3.11. Control/Command click again to collapse the sibling groups.

**Selecting Files and Groups**

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

When a group is selected, all of the files within the group are selected, regardless of whether or not one of its files are included in the selection.

**Selection by mouse-clicking**

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

To select a consecutive list of files or groups, select the first file or group in the list by clicking its name, then Shift+click the last file or group to be selected.
Working with Projects
Managing Files in a Project

Everything between and including the first file or group you clicked on and the last file or group you clicked on will be selected, whether they’re groups or filenames. You cannot select a consecutive list containing only files using this method. You can select a consecutive list containing only groups using this method, but only if all groups are completely collapsed.

Another way to select a consecutive list of items is to drag-select them, like you would select items on the desktop.

Use Control/Command click to select or deselect any non-consecutive file or group in the project window as shown in Figure 3.12.

Figure 3.12  Non-consecutive file selection

Selection by keyboard

With the project window frontmost, you can type the first few characters of the name of the item 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/Delete key if you make a typo.

Use the Enter/Return key to open a 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 IDE project.

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

When adding a file to a project, the 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.

Normally, the IDE forces each file in a project to have a unique name. To remove this restriction, turn on the Save Project Entries Using Relative Paths option in the Target Settings panel.

For more information about Access Paths see “Access Paths” on page 280.

For more information about the Message Window, see “Guided Tour of the Message Window” on page 316.

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 65.
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.

This command opens a dialog box to add files to your project from a directory (Figure 3.13). 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 299.

Windows Add Files

The Files of Type pop-up is automatically set to view all files types.

Select the file in the Files List that you wish to add to your project, then click the Add button.

To select multiple files, as shown in Figure 3.13, 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.

When all the files you want to add are selected, click the Add button.
Managing Files in a Project

Figure 3.13  Adding files to a project (Windows)

Mac OS Add Files

This dialog box is split into two lists (Figure 3.14).

Use the pop-up menu at the top of the Files List to navigate through folders and drives to find files to add. The Select Files To Add List shows the files you have added to your project.

Select the file in the Files List that you wish to add to your project, then click the Add button. The selected file is added to the Select Files To Add list. Alternatively, you can double-click the selected file to move it to the Select Files To Add list.

To add all the files displayed in the file list to the Select Files To Add list (all available files in the folder), click the Add All button. This does not add files stored in any subfolders shown in the file list.

To remove a file from the Select Files To Add list, select the file to be removed. If you erroneously select a file, clicking the file name a
second time will deselect it. Once you’ve selected the appropriate file to be removed, click the **Remove** button.

**Figure 3.14  Adding files to a project (Mac OS)**

To clear the Select Files To Add list, click the **Remove All** button. The **Cancel** button closes this dialog box without adding any files to the Project window.

When you’ve finished selecting all the files you want to add from the currently-selected directory, click **Done**. The files shown in the Select Files To Add list are added to your project.

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

**Using drag and drop**

When you drag and drop files or folders onto an IDE 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 IDE won’t accept the drag.

Use the focus bar (an underline) that appears in the Project window to select the location where files will be inserted into the project list.

Releasing the mouse button (dropping the files) adds the dragged items to the project, inserting them below the position specified by the focus bar. If there are multiple targets in the project file, a dialog box ([Figure 3.15](#)) opens that enables you to identify which targets should receive the files.
To create a new group and add files to it, drop the files when the cursor is over the blank space after the last group.

The CodeWarrior IDE does not allow the dragging of entire volumes (such as your hard disk) onto the project window.

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 on all platforms, some do not allow you to remove files by dragging them 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 editor window to the project. You typically use this command when you’ve created a new file and decided that you would like to add it to the active project.

To use the **Add Window** command, select a location in the project window. Then, open a source code or text 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 (Figure 3.15).

**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 290). The **Add Window** command is dimmed otherwise.

Moving Files and Groups

To move one or more files or groups within a project’s **File** view, or to arrange targets in a project’s **Targets** view, 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 65.

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

A focus bar (an underline) indicates where the selected files will be moved to when the mouse button 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, allowing you to rearrange groups.

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

When a file has the focus bar underneath itself, and the mouse is released, the selected files are placed in the same group, following this file. When a group has the focus bar underneath, the selected files are placed at the end of the chosen group.

**TIP:** The focus bar has a small arrow at the left end that indicates the level of the insertion. If the arrow is to the left of a group icon, the insertion will be at the same level as the target group. If the arrow appears to the right of the icon, the files are inserted into the target group.

**Creating Groups**

With the Project window frontmost, and the File view visible, choose the Create New Group command from the Project menu. Enter a name for the new group in the Create Group dialog box, then press OK.

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.

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

**Mac OS** The new group may use “Segment” in place of “Group” if the project is targeted for the 68K.
Removing Files and Groups

There are two methods used to remove files from a Project window. You can use menu commands or, drag and drop on supported platforms to remove files from a Project window.

**Using menu commands**

You can remove files from either the File view, or the Link Order view of the project window. 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 Link Order view (also called the Segments view for some targets), the files are removed 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 in the Files view, 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 65.

After selection in the Files view, choose the Remove Selected Items command from the Project Menu or press Alt/Option Delete. All the selected files and groups are removed from the project.

**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.

To remove a group from your project in the Files view without removing the files that are in the group, select the group to remove and press Shift+Control+Enter/Shift+Command+Return. The CodeWarrior IDE moves the files in this group up to the previous group and the group numbers are updated.
When you remove a group in the Links view, 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 are added to the group that is above the group you are deleting.

If there is only one group in your project, whether in the Files or Link view, you won’t be able to delete the group, since the CodeWarrior IDE requires at least one group in a project.

Using drag and drop

The CodeWarrior IDE supports dragging and dropping files into the project window, you cannot drag files out of the project window on some platforms.

Renaming Groups

To rename a group, select the group to be renamed by clicking on it then press the Enter/Return key. You may also use the arrow keys to navigate to the group, then press the Enter/Return key. A dialog box appears (Figure 3.16) containing a name text field you use to enter a new group name.

Figure 3.16 Changing a group name

Type the new name in the Enter Group Name field 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.
Working with Projects
Managing Files in a Project

Note that when you attempt to change the group’s name again, “Group” 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.

Mac OS On Mac OS 68K targets, the dialog box in Figure 3.17 will appear. It contains a series of checkboxes that affect segment loader information. The Mac OS 68K linker gives this name to the code segment when the project is built. See the Targeting MacOS manual for more information on using segments.

Figure 3.17 Changing a segment name

Touching and Untouching Files

Use the Touch column shown in Figure 3.2 on page 43 to mark 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 files get compiled. One way is to click in the Touch column beside the filename in the
project window’s **File** view. A check should appear in the Touch Column next to the file name.

Another way is to select the Touch command from the **Interface pop-up** menu. The Touch command may appear at the top of the **Interface 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 to 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 the Interface pop-up is **Touch**. When you choose Touch, the CodeWarrior IDE marks your file to be compiled 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.

To unmark files so that they won’t be compiled, click again in the **Touch column** left of the file name, or choose **Untouch** from the **Interface 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.

**Synchronizing modification dates**

To update the modification dates stored in your project file, click the icon above the **Touch column**. 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 modi-
Working with Projects

Working with Complex Projects

The CodeWarrior IDE provides flexible facilities for creating project files that use sophisticated build rules. This section discusses how to construct complex project files that may contain 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 your software. By switching between shipping and debug targets, the IDE generates different versions of the software 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 optimizations enabled.

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

NOTE: (Mac OS) Some third-party editors use AppleEvents to let the CodeWarrior IDE know when you modify a file. These editors include BBEdit from BareBones Software, Object Master, and Alpha. You don’t need to use the Synchronize Modification Dates command if you use one of these external editors.
What is a Target?

A target is a set of rules and settings that you configure to produce an output file, such as an application or library.

The CodeWarrior IDE has the capability to build many different kinds of output files, or targets, from one project file, as shown in Figure 3.4 on page 48. 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 82.

What is a Subproject?

A subproject is a project file that is nested within a project, as is the project named Java Life.mcp shown in Figure 3.18. Subprojects are useful if you have a project file that you want to keep separate from the main project file. This allows you to organize the build process into separate project files.
One case where this organization might be useful is for developing applications that use a plug-in architecture. Suppose your program uses many different plug-in 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 86.

To learn how to add files to a project, refer to “Adding Files” on page 67.

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 of the main project with the output of the subproject’s target by selecting the marker for the target in the Link column of the main project’s Targets view. For information on the Link column, see “Targets View” on page 47.
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.

**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 using 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 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 view selected in your project window, as shown in Figure 3.4 on page 48.
After you choose this menu command, you see the dialog box as shown in Figure 3.19. 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 85.

Figure 3.19 New Target dialog box

To learn how to configure settings for a target, refer to “Choosing Target Settings” on page 275.
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.

**Changing a Target Name**

To change the name of a target in the Targets view of the project window double-click the name of the target to open the settings dialog box for that target, as shown in Figure 9.4 on page 278. Select the Target Settings panel from the list of available panels. Change the name of the target using the Target Name editable text field.

You can also open the settings dialog box using the Target Settings command in the Edit menu.

To learn more about the Target Settings panel, refer to “Target Settings” on page 278.

**Changing the Target Settings**

Each target in a project has its own settings. You modify these settings through the settings dialog box.

To open this dialog box for a particular target in your project, just double-click the name of the target in the Target view of the project window. Or, use the Target Settings menu command in the Edit menu.

The left side of this dialog contains a list of all settings panels appropriate for this target. Select a panel to see the options you can set.

To learn how to change these, refer to “Configuring Target Options Overview” on page 271. All of the generic settings panels are described in that section of this manual. Settings panels that are specific to a particular operating system are described in a corresponding Targeting manual, such as Targeting Mac OS or Targeting Win32.
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 view of the Project window, as shown in **Figure 3.4 on page 48**. 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 refers to information in a second target, the second target will be built before building the first target.

To specify that a target depends on a second target, first go to the **Targets** view of the Project window. Drag the second target below and indented to the right of the first target’s entry. The IDE adds an italicized entry within the first target’s group for the second target. This makes the first target dependent upon the presence of target two. Now, when the IDE builds the first target, it ensures that the second target is built before attempting to build the first target.

To specify that the first target should be linked with object code from the second target, click the marker in the Link column of the second target’s italicized entry within the first target’s entry.

If you don’t know how to create new targets, refer to “Creating a New Target” on page 82 to learn how.

Refer to “Setting the Current Build Target” on page 85 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 82.

Assigning Files to Targets

There are two ways to assign files to targets in a project: with the Target column in the File View and with the Project Inspector.

Assigning files to targets with the Target column

The Target column in the Project window’s File view indicates whether a file is in the current target or not. The IDE displays this column if the project has more than one target. A file in the File view has a marker in the Target column if it is in the project’s current target.

To assign a file or group to the active target, click in the item’s Target column to place a marker. To remove a file or group from the active target, click in its Target column to remove its marker.

To assign or remove all the items in the File view to the current target, Alt-click/Option-click in the Target column to display or erase markers.

Assigning files to targets with the Project Inspector

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 65. Then, choose the Project Inspector menu command from the Window Menu. A window appears, as shown in Figure 3.20.
Click the Targets tab to switch the view to that shown in Figure 3.21. This window is showing you which targets your selected 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 Save button.
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 3.22, with a project file added to the list of files in your project. You may also add the file using the methods discussed in “Adding Files” on page 67.
Adding a project to another project makes the subproject file part of the main project file. When you do a \texttt{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 86.

\section*{Examining Project Information}

The CodeWarrior IDE allows you to review and configure information about your project and source code files, using the \texttt{Project Inspector} window, shown in Figure 3.20. To show this window, choose the \texttt{Project Inspector} command from the \texttt{Window Menu}.

There are two tabbed views 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 86, as that section of the manual discusses Targets in detail.

To learn more information about a given file in your project, you can review its Attributes settings. The Attributes include the file name, its path to a location on your hard disk, the name of the
Moving a Project

The CodeWarrior IDE stores all required information about your project in the project file. There are other files, usually stored in a folder having a name similar to your project file, that contain information about window positions, object code, debug info, browser data, and other settings which are not crucial to rebuilding the code of your project. These additional files are not needed by the CodeWarrior IDE to recreate your project.

**WARNING!** (Windows) 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.

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 is 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 280.

See also “Opening Project Files Created on Other Host Platforms” on page 58.

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 its files. 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. If you select the Enable Debugger command from the Project Menu, all the necessary configuration is done for you automatically.

To learn how to do this manually, refer to “Choosing Target Settings” on page 275.

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 3.23 on page 93. When you add files to your project, CodeWarrior automatically sets this marker unless you have deselected the appropriate option in the project’s Linker 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 40. When files are added to the project, their debug state is set by the state of the Enable 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.

To generate Debug Info for all files in the project, use Shift/Option click in the Debug column to enable or disable generating debugger information.

**NOTE:** Marking a source code file for debug inclusion does not mean that a debug file is created during linking. The linker settings panel for the target contains the options that enable 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 3.23), 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 3.23  Debug Info markers

Gray dot indicates that some files in group selected for debug info generation.

Black dot indicates debug info will be generated for this file.
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:

```c
#define GLOBAL_DEBUG
```

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

To do this, you would create a precompiled header and insert this symbol definition into the header. See “Using Precompiled or Pre-processed Headers” on page 308 for more information on how to do this.

To learn more about this topic, refer to the Inside CodeWarrior:C/C++ Tools and Inside CodeWarrior:Pascal Tools. manual.
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, comparing, and printing files in the CodeWarrior environment.

To learn about editing files, refer to “Source Code Editor Overview” on page 121.

To learn about working with files using revision control systems, refer to “Version Control System Overview” on page 331.

The sections 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
- Comparing and Merging Files & Folders

Creating a New File

To create a new untitled window where source code or text may be entered, choose the New command from the File Menu.
After the new window appears, a text insertion point appears on the first line of the window. The CodeWarrior IDE places text that you type at this insertion point.

To learn more about text editing in the window you have just created, see “Source Code Editor Overview” on page 121.

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 with the CodeWarrior Editor, because of their binary format.

Opening Files with the File Menu

You can open two types of files:

- 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. For information on opening CodeWarrior project files, see “Opening an Existing Project” on page 56.
Text file

To open a text or source code file, choose the Open command from the File Menu. The IDE displays an Open dialog box, as shown in Figure 4.1 or Figure 4.2.

Windows  From the Files of Type pop-up menu, select All Files. The list of files changes to show all the files in the current folder, including text files.

Mac OS  The dialog box displays a list of the available project and text files in the current folder. If the file is a stationery text file, the IDE will open a new, untitled Editor window and copy the contents of the stationery file into the window.

Figure 4.1  Open dialog box (Windows)
Opening an Existing File

Select the file you would like to open, and click Open. The CodeWarrior IDE opens the file in an Editor window.

For more information about editing source code, see “Source Code Editor Overview” on page 121.

Opening Files from the Project Window

There are different ways to open files from within the project window, depending on the type of file you wish to see. These different ways are:

- **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

**TIP:** (Mac OS) Files that contains binary data cannot be opened and displayed in a CodeWarrior Editor window. If the file was created using another application, double-clicking the file name in the **File column** will open the file in the application that created it.
Refer to “Disassembling Source Code” on page 315 to learn how to view the contents of a library file.

File column

If the file you wish to see appears in the File column of the project window’s File or Link Order views, double-click the file name to open it. If the file is a text file, CodeWarrior opens it in an Editor window.

Mac OS On the Mac OS, if the file is any other type, the CodeWarrior IDE opens the application that created the file. If the file is a binary file, such as a library file, it will not be opened, because binary files are not viewable using printable characters.

Another way to open a file is to select it, and then press the Enter/Return key. You can select multiple files in the Project window, and open them all by pressing the Enter/Return 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 65.

For more information about the File Column, refer to “File column” on page 42.

Group pop-up menu

Another way to open a source file is to use the Interface pop-up for a group. A similar menu is shown in Figure 4.3. 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 Interface 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 Interface pop-up to see a list of files. Select the file you want to open from this list, as shown in Figure 4.3.
Opening an Existing File

Note that header files inside “<...>” are system header files located within the Metrowerks CodeWarrior folder. Files without these symbols are header files you have created and are stored in the same folder as your project or other Access Paths you have designated. To learn more about Access Paths and how to configure them, refer to “Access Paths” on page 280.

TIP: To switch between a source file and its interface file, use the same name for both files, except for the extension. For example, name your files foo.cpp and foo.h. Then press Ctrl/Command Tab to instantly switch between the two files.

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 Untouch 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.

To learn more about touching files, see “Touching and Untouching Files” on page 77.

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 left of the editor window as shown in Figure 5.2 on page 124. 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 Find and Open File 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 Find and Open File command from the File Menu.

The CodeWarrior IDE then searches for the file and opens the file in an editor window.

Here is another method for opening a file. Let’s use an example. If you are editing a C++ .cpp file you can type the Ctrl/Command Tab keyboard shortcut and the CodeWarrior IDE will open a new window in the Editor displaying the corresponding .h file. For this to work, your source and interface files must have the same name with different file name extensions. For example, if you are editing the file myFile.cpp and you press Ctrl/Command Tab, The CodeWarrior IDE will search the Access Paths for the file called myFile.h to open with the Editor.

This method also works if you are editing the .h file and want to see the .cpp file. Just type Ctrl/Command Tab and the corresponding file will be opened for you.

If you’re editing a source code file and want to open a file without selecting any text, choose the Find and 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 Find and Open File, the CodeWarrior IDE then displays a dialog box, as shown in Figure 4.4. Type the name of the file you wish to search for in the Open editable text field.

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.
Opening an Existing File

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

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

To learn more about Access Paths and how to configure them, refer to “Access Paths” on page 280 for more information.

Opening a Related File

If you are working in a source code file and wish to open the corresponding interface file, or working with a interface file and wish to open the corresponding source file, use the keyboard shortcut Ctrl/Command Tab. You can easily switch back and forth between the two files.
Saving a File

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

- Saving one file
- Saving all files
- Saving files automatically
- Renaming and saving a file
- Backing up files
- Saving as a MS-DOS, Mac OS, or UNIX text file

**NOTE:** (Mac OS) When saving a file, you will often have the option of saving it 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. See "Opening Files with the File Menu" on page 96 for more information.

**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.

**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 104. Choose a name and location for your new file with this dialog box.
Projects are saved when they are closed, when you quit or exit the CodeWarrior IDE, or when the Save A Copy As command is selected. You don't need to explicitly save projects.

**Saving all files**

To save your changes to all the files currently open, press the keyboard shortcut Shift-Ctrl-S/Option-Command-S. The CodeWarrior Editor saves all the modified files to disk.

**Saving files automatically**

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

Use the Save All Before “Update” 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 234.

**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 4.5 or Figure 4.6.
Mac OS  Choose the Text or Stationary button to save the file as either a text file or a stationery file.

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 106.
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 in 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 4.5. Specify the file’s new location and choose a unique name for the file.

Mac OS You can also choose whether to save the file as text stationery or a text file.

Now 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.
Figure 4.7  Saving a copy of a project window (Windows)

![Image of saving a copy of a project window (Windows)]

Figure 4.8  Saving a copy of a project window (Mac OS)

![Image of saving a copy of a project window (Mac OS)]

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 4.7. Saving the project as a text file creates a text file that contains the names of all the files in the project.

**Saving as a MS-DOS, Mac OS, or UNIX text file**

When you open a text file originally created in a Windows, Mac OS, or UNIX text editor, CodeWarrior internally converts it to be compatible with the platform upon which it is currently operating, fixing any inconsistent line endings it may find, so that you may edit the file. When you save the file, CodeWarrior saves it in its original format.

To learn about saving a text file under a different text format, see “Options Pop-Up Menu” on page 126.

---

**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 4.9. If you choose to close the file without saving your changes, all changes are lost.
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 4.9. For more on saving project files, consult “Saving a Project” on page 61.

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 234 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.

Mac OS The Close command saves the window position information in a format that is compatible with MPW. If you open a saved CodeWarrior Editor file with MPW, the same window parameters are respected. For more information about MPW, refer to the documentation on the CodeWarrior Reference CD.

Closing All Files

To close all open windows, use the Close All keyboard shortcut. If you modified any file during the edit session, the Editor prompts

Figure 4.9 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 4.9. For more on saving project files, consult “Saving a Project” on page 61.

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 234 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.

Mac OS The Close command saves the window position information in a format that is compatible with MPW. If you open a saved CodeWarrior Editor file with MPW, the same window parameters are respected. For more information about MPW, refer to the documentation on the CodeWarrior Reference CD.

Closing All Files

To close all open windows, use the Close All keyboard shortcut. If you modified any file during the edit session, the Editor prompts
you for save information before closing each window that contains changes.

**Close All** doesn’t close all the CodeWarrior IDE windows, just editor and debugger windows. The Find dialog box and any Project windows remain open when using this command.

**TIP:** To close all Editor windows at once, press Alt/Option 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 the printer setup appropriate for your platform. CodeWarrior displays the appropriate printer dialog box.

**Windows** Choose Print Setup (Windows) from the **File Menu**.

**Mac OS** Choose Page Setup (Mac OS) from the **File Menu**.

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.

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

To print a window, make the window active and choose the Print command from the File Menu. This menu command allows you to print some or all of the active window.

When you choose this command, the CodeWarrior IDE displays the print dialog box for your printer. There are two additional CodeWarrior-specific options available for configuration in this dialog box. Depending on your printer and printer software, these options may be displayed in various places in different print dialog box.

The CodeWarrior-specific options are:
- **Print Selection Only**
- **Print using Syntax Highlighting**

Print Selection Only

If there is selected text in the Editor window you are printing, the Print Selection Only option appears. When this option is on, the CodeWarrior IDE prints only the selected text in the window, not the entire file. When this option is off, the CodeWarrior IDE prints the entire file.

Print using Syntax Highlighting

When the Print using Syntax Highlighting option is on, the CodeWarrior IDE prints the file with syntax coloring. On a black and white printer, the colors come out as shades of gray. When the Print using Syntax Highlighting option is off, the CodeWarrior IDE prints the file in black and white without syntax coloring.

**TIP:** (Mac OS) To print color syntax-highlighted text in bold and comments in italics, choose Black & White printing from the Print dialog box and turn on Print using Syntax Highlighting.
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 4.10 appears.

![Figure 4.10 Revert to a Previous File](image)

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 click Cancel, the file you’re working with is not changed or saved to disk, and you can continue editing it.
Comparing and Merging Files & Folders

The CodeWarrior IDE has a facility to compare two text files, mark the differences between the files, and apply changes between the files. In addition, you can also compare the contents of two folders.

The topics in this section show you how to use the IDE’s file comparison features:
- File Comparison and Merge Overview
- Choosing Files To Compare
- Examining and Applying Differences
- Choosing Folders To Compare

File Comparison and Merge Overview

The IDE’s file comparison window displays two text files and the differences—insertions and deletions—between them. The window has controls to examine, add, and remove the differences between the files. The currently selected difference is shown with a darker color and outlined in black to contrast it from the other differences visible in the window.

The file comparison window has these parts:

Source file

Displays the source text file that IDE uses as a basis for its comparison with the destination file. This pane appears on the left side of the file comparison window.

Destination file

Displays the source destination file that’s compared with the source file. This pane appears on the right side of the file comparison window. Differences between the source file and the destination may be added to or removed from the destination file.
Working with Files
Comparing and Merging Files & Folders

Figure 4.11 The file comparison window

Source file
Comparison column
Destination file

Difference list
Toolbar

Comparison column
Shows a graphical representation of where text was added or removed between the source and destination files. This column appears between the source and destination panes in the comparison window.
Difference list

Lists the insertions, deletions, and lines of mismatching text between the two files. Selecting an item in the list displays the difference in the source and destination panes. The comparison column also shows how and where the difference occurs between the two files. Text will appear in *italics* when a difference is applied.

Toolbar

Has buttons to apply or remove changes between the two files to the destination file. The toolbar also has buttons to undo and redo changes to the source and destination files. For information on configuring toolbars, see “Customizing Toolbars” on page 262.

The controls include:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Apply difference]</td>
<td>Apply difference</td>
</tr>
<tr>
<td>![Unapply difference]</td>
<td>Unapply difference</td>
</tr>
<tr>
<td>![Undo]</td>
<td>Undo</td>
</tr>
<tr>
<td>![Redo]</td>
<td>Redo</td>
</tr>
</tbody>
</table>

Choosing Files To Compare

To open a file comparison window, choose Compare Files from the Search Menu to show a dialog box that prompts you for two files, source and destination files, to compare. To use a file dialog box to browse for the source and destination files, click their respective Choose buttons. You may also drag and drop text files into their respective boxes.
Working with Files
Comparing and Merging Files & Folders

Figure 4.12  Compare Files Setup dialog box

Text Compare Options

Select the Case Sensitive checkbox to consider the case of letters as part of the comparison operation. To ignore the case of letter, deselect this checkbox.

To take space and tab characters into account while comparing, select the Ignore Extra Space checkbox. To ignore extra space, and tab characters, deselect this checkbox.

See “Folder Compare Options” on page 117 for information on the folder comparison options.

To show the comparison window after choosing files and setting options, click the Compare button.

Examining and Applying Differences

Use the comparison window’s toolbar and difference list to choose among the differences between the source and destination files and apply changes to the destination file.

To view a difference between the two files, click its entry in the difference list. To apply the difference, click the Apply button in the
toolbar or choose **Apply Difference** from the **Search Menu**. To reverse a difference you’ve already applied, click the Unapply button or choose **Unapply Difference** from the **Search Menu**.

**WARNING!** Currently, using the **Apply** and **Unapply** difference buttons erase all actions from the Undo stack. In other words, when you exit the Difference window after apply or unapplying, all undo and redo actions will have been cleared from the Undo stack.

### To Compare Editor Files

To compare two files that are already open in Editor windows, click on the **Editor Files** pop-up menu next to the source and destination paths. A list of open editor windows appear. Choose a file name from the menu to make it the source or destination file. Click **Compare** once the source and destination files are chosen to display the comparison window.

### Choosing Folders To Compare

To open a folder comparison window, choose **Compare Files** from the **Search Menu** to show a dialog box that prompts you for two folders, the source and destination folders, to compare. To set these folders, drag and drop the folders into their respective boxes (Figure 4.13).

**Folder Compare Options**

Select **Only Show Different Files** to only display files that are different in both folders in the **File in Both Folders** list of the Compare Folders window. By default, this option is disabled, so all files in the source and destination folders are displayed.

Comparisons between files in the source and destination folders are normally based upon the file modification dates and file sizes. This is usually good enough to determine if there are differences between the two files.
Select **Compare Text File Contents** to perform a more accurate compare of the files in the two folders. In essence, this performs a Compare Files on every file in the source and destination folders and does not check the modification dates nor file sizes. This option is a lot slower since it has to open every file, but the comparison information is more accurate.

See "Text Compare Options” on page 116 for information on the file comparison options.

The **Files in Both Folders** list displays all files in both the source and destination folders unless the **Only Show Different Files** option is enabled. Files that are different in the two folders have a small bullet positioned to the right of their name.

Click on a file in the list to view the Selected Items specific information at the bottom of the window.

Double-click on a file in the **Files in Both Folders** list to open a Compare Files window for resolving the differences between the two differing files.
The **Files Only in Source** list display all the files that appear only in the source folder while the **Files Only in Destination** list displays only files that appear within it.

You can click on a zoom box for any of the three lists to expand them to fill the window. Click again to collapse back to their original size.

Use the Toolbar area at the top of the window to add Toolbar commands to the Compare Folders window. By default, no toolbar commands appear in the Toolbar area of the window. See “Modifying a Toolbar” on page 267 for information on adding toolbar commands.
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 among your routines quickly.
- Syntax highlighting that formats source code for easy identification of comments and keywords in your source files.
- Convenient on-line reference material for routines available instantly. You just point and click.

The topics in this chapter are:

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

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 234.
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.

Figure 5.1 The Editor window

To see an editor window, create a new text file using the New command on the File Menu.
**TIP:** (Mac OS) Command-click support in the Editor for Internet Config (an internet extension freely available on the World Wide Web) is supported. Command-clicking on a URL (Uniform Resource Locator) will launch the configured helper application.

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
- VCS Pop-up Menu
- File Path Caption
- Dirty File Marker
- Pane Splitter Controls
- Line Number Button
- Toolbar Disclosure Button
- Mac OS Path Pop-Up Menu

**Text Editing Area**

The Text Editing Area of the editor window is where your text is entered and edited.

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 138.
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.

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 96.

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

To learn more about touching files, see “Touching and Untouching Files” on page 77.

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**

![Routine pop-up menu](image)

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

**NOTE:** 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 Ctrl/Option and click on the routine icon.

If you’d like to change the default display order of the routines to alphabetical, enable the Sort Function Popup option. See “Editor Settings” on page 234 for more information about editor options.

**TIP:** If you’re editing a Pascal file, the Routine Pop-up Menu displays function names in italics, procedure 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 line of text, to remember where you left off, and for other identification purposes.

Figure 5.4 The Marker pop-up menu

For more information on using markers, see “Adding, Removing, and Selecting a Marker” on page 142.

Options Pop-Up Menu

Use the Options Pop-up Menu, shown in Figure 5.5, to turn color syntax highlighting on or off for the current file, and also to set the format for how to save the file.

To turn on or turn off syntax highlighting, choose Syntax Highlighting from the Options popup menu.

A checkmark next to the Macintosh, DOS, or UNIX options indicates the type of file currently open in the Editor window. The next time you save the file, the CodeWarrior IDE saves it in the format you select.
The Options pop-up menu

For more information on the Syntax Coloring option shown in this menu, see “Syntax Coloring” on page 240.

VCS Pop-up Menu

The VCS Pop-up Menu indicates the read/write revision control database status of the current file. If the pop-up icon box shows an Unlocked icon or the Read/Write icon, 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 337.

Using this pop-up menu, you can get a new copy of your file, check-out the file for modification, make it writable so you can make changes without doing a checkout, and other operations.

For more information about revision control system software, see “Using Source Code Control with Files” on page 337.

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 122.

Mac OS To learn about another method of determining the path of a file, refer to “Mac OS Path Pop-Up Menu” on page 129.
Dirty File Marker

The Dirty File Marker tells you if the file displayed in a window has been modified since it was last saved or opened. The states of the Dirty File marker include:

- ![Diamond symbol] Unchanged file
- ![Red checkmark] Modified and unsaved file

Pane Splitter Controls

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

You use these controls to adjust the sizes of the panes after you’ve created them. Figure 5.8 on page 131 shows an editor window with multiple panes.

For more information on this topic, see “Splitting the Window into Panes” on page 131.

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 146.

Toolbar Disclosure Button

The Toolbar Disclosure Button hides or displays the Editor window’s toolbar along the top of the window. If the toolbar is hidden, a row of smaller controls appears at the bottom of the Editor window (see Figure 5.7 on page 130).
For more information on using the Toolbar Disclosure Button, refer to “Seeing Window Controls” on page 130.

**Mac OS Path Pop-Up Menu**

To see the directory path of the file in the active Editor window, press the Command key and click on the name of the file in the title bar of the window, as shown in Figure 5.6. You can also directly open a folder by choosing it from the menu.

![Figure 5.6 File Path Pop-up in the window titlebar](image)

**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 Editor Window Settings

To learn about configuring the Editor window’s toolbar, see “Customizing Toolbars” on page 262.
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, see “Font and Tabs” on page 239.

Seeing Window Controls

The row of pop-up menus and controls that appears along the top of the editor window is the Editor window’s toolbar. The Toolbar Disclosure Button is shown in Figure 5.1 on page 122.

However, if you hide the toolbar, the default pop-up menu controls appear along the bottom of the editor window, as shown in Figure 5.7. Note that the File Path Caption is no longer visible.

NOTE: If you hide a customized Editor window toolbar, custom items do not appear at the bottom of the window. The default items always appear when the toolbar is hidden. When you show the toolbar again, it retains its custom configuration. For information on toolbars in general, and customizing them, see “Customizing Toolbars” on page 262.
To show the toolbar along the top of the editor window again, click the Toolbar Disclosure Button once more.

You can make your choice about whether to display the toolbar the default choice for editor windows. To do so, see “Toolbar Disclosure Button” on page 128 for more information. See also “Showing and Hiding a Toolbar” on page 265.

### Splitting the Window into Panes

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

**Figure 5.8  Multiple panes in a window**
Creating a new pane

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

As you drag a Splitter Bar, a gray focus line tracks your progress and indicates where the new pane will go. When you release the mouse button, the editor creates a new pane.

You can also double-click 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, a gray focus lines indicates 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, a gray focus line indicates 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.

You can also double-click on a Resize Box to remove a split.

Saving Editor Window Settings

The current settings of an editor window are automatically saved whenever the window is closed, or when the toolbar is hidden or shown. The Save Default Window menu command in the Window Menu is disabled when an editor window is frontmost.
The settings saved are the size and location of the window, and the setting of the Toolbar Disclosure Button. Any new editor 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.

**WARNING!** (Mac OS) If you are using a revision control system for your source code files that uses native file locking, you must have your file checked out in order for the settings to be saved. If you are using c.k.i.d resources to lock the files, the information is saved for you without checking out the file.

To learn more about configuring editor window settings, refer to “Font and Tabs” on page 239.

To learn about saving CodeWarrior Browser windows, see “Saving a Default Browser” on page 221.

**Basic Text Editing**

The CodeWarrior IDE gives you lots of help in editing source files, all of it described in the topics 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. Review this section again after you become more familiar with the CodeWarrior editor’s features.

Scrollbar navigation

Like any other text editor, you may adjust the text to view in an Editor window by using the scroll bars.

The CodeWarrior IDE lets you configure how the scroll bars affect the window view when you drag the scroll bar thumb 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 236.

Keyboard navigation

Table 5.1 describes how to move the insertion point around in a file with function keys.

<table>
<thead>
<tr>
<th>To move insertion point to</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>Control-Left Arrow</td>
<td>Option-Left Arrow</td>
</tr>
<tr>
<td>Next word</td>
<td>Control-Right Arrow</td>
<td>Option-Right Arrow</td>
</tr>
<tr>
<td>Beginning of the line</td>
<td>Home</td>
<td>Command-Left Arrow</td>
</tr>
<tr>
<td>End of the line</td>
<td>End</td>
<td>Command-Right Arrow</td>
</tr>
<tr>
<td>Beginning of the file</td>
<td>Control-Home</td>
<td>Option-Up Arrow</td>
</tr>
<tr>
<td>End of the file</td>
<td>Control-</td>
<td>Option-Down Arrow</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>To move insertion point to</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous page</td>
<td>Page Up</td>
<td>Page Up</td>
</tr>
<tr>
<td>Next page</td>
<td>Page Down</td>
<td>Page Down</td>
</tr>
<tr>
<td>Beginning of the file</td>
<td>Control-Home</td>
<td>Home</td>
</tr>
<tr>
<td>End of the file</td>
<td>Control-End</td>
<td>End</td>
</tr>
<tr>
<td>Insertion point</td>
<td>Left Arrow or Right Arrow</td>
<td>Left Arrow or Right Arrow</td>
</tr>
<tr>
<td>Previous line</td>
<td>Control-Up Arrow</td>
<td>Control-Up Arrow</td>
</tr>
<tr>
<td>Next line</td>
<td>Control-Down Arrow</td>
<td>Control-Down Arrow</td>
</tr>
</tbody>
</table>

### 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 134.

### Deleting Text

There are several different methods for deleting text.

To delete text that you just typed, press the Backspace/Delete key.
To delete text that is in front of the text insertion point, use the Delete/Del (⌫) key.

To delete text from the text insertion point to the end of the file, use the Control/Command-Delete keyboard shortcut.

To delete more than one contiguous character at a time, select the text you want to delete and press the Backspace/Delete keyboard shortcut.

If you don’t know how to select text, learn how by reading “Selecting Text” on page 136.

### Selecting Text

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

Select text by holding down the Shift key while pressing any of the shortcuts listed in Table 5.1. You may also select a word, a line, or a range of text.

To select a word:
- double-click on the word

To select a line:
- triple-click anywhere in the line
- move the mouse pointer to the left edge of the editor window so that the mouse pointer points left and press the mouse button

This selection method is available when the Left Margin Click Selects Line option is on in the Editor Settings preference panel.

To select a range of text:
- click and drag the mouse in a portion of your window where there is text
- set your text insertion point to mark the beginning of your selection then press the Shift key while clicking the place in your text where you want the selection to end
• move the mouse pointer to the left edge of the editor window so that the mouse pointer points left, then click and drag the mouse pointer to select lines of text

This selection method is available when the Left Margin Click Selects Line option is on in the Editor Settings preference panel.

Mac OS 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 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.

Table 5.3 describes how to select text using the keyboard, starting at the current 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>
<thead>
<tr>
<th>To select text to</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>Shift-Control-Left Arrow</td>
<td>Shift-Option-Left Arrow</td>
</tr>
<tr>
<td>Next word</td>
<td>Shift-Control-Right Arrow</td>
<td>Shift-Option-Right Arrow</td>
</tr>
<tr>
<td>Beginning of the line</td>
<td>Shift-Home</td>
<td>Shift-Command-Left Arrow</td>
</tr>
<tr>
<td>End of the line</td>
<td>Shift-End</td>
<td>Shift-Command-Right Arrow</td>
</tr>
<tr>
<td>Beginning of page</td>
<td>Shift-Page Up</td>
<td>Shift-Option-Up Arrow</td>
</tr>
</tbody>
</table>
Editing Source Code
Basic Text Editing

<table>
<thead>
<tr>
<th>To select text to</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of page</td>
<td>Shift-Page Down</td>
<td>Shift-Option-Down Arrow</td>
</tr>
<tr>
<td>Beginning of the file</td>
<td>Shift-Control-Home</td>
<td>Shift-Command-Up Arrow</td>
</tr>
<tr>
<td>End of the file</td>
<td>Shift-Control-End</td>
<td>Shift-Command-Down Arrow</td>
</tr>
</tbody>
</table>

For more information about using drag and drop with text, see “Moving Text (Drag and Drop)” on page 138.

You can also select blocks of code quickly using the Balance command. To learn how to do this, refer to “Balancing Punctuation” on page 139.

Moving Text (Drag and Drop)

If you have some text in your editor window that you would like to move to a new location, 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 234.

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.

**TIP:** (Mac OS) If you press the Option key and drag selected text, the text will be copied to your new location instead of moved.

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**.
Editing Source Code
Basic Text Editing

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 356.

Balancing Punctuation

When you’re editing source code, you often have to 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 check. Then, choose Balance from the Edit Menu. Alternatively, double-click on a parenthesis, bracket, or brace character that you want to check for a matching character.

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.

**TIP:** Use the Balance command to select blocks of code quickly.

Using automatic balancing

You can have the 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 236.
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 136. 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 “Font and Tabs” on page 239.

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 your last action. For example, if you just typed in some text, the command changes to Undo Typing. Choose Undo Typing to remove the text you just typed.

Undoing and redoing multiple edits

When the Use Multiple Undo option is on, 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 237.

Note that when the **Use Multiple Undo** option is turned off, the Undo keyboard shortcut switches between **Undo** and **Redo, Multiple Undo, and Multiple Redo** in the **Edit Menu**.

---

**WARNING!** Undo actions are saved in a stack, therefore it’s possible to lose actions when performing several undo and redo actions. Each undo action adds an item to the stack, while each redo repositions a pointer to the next undo action. For example, if there were five undo actions on the stack (**ABCDE**), and you redo two of them, the stack appears to the pointer like this: **ABC**, then perform a new action (**ABCF**), the undo events (**DE**) are no longer available.

---

**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 112.

---

**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 240.

You can also highlight custom keywords, which are in list of words you designate. See “**Syntax Coloring**” on page 240 for instructions on configuring the Editor to do this for you.
Navigating in Text

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

This section covers these methods:

- Finding a Routine
- Adding, Removing, and Selecting a Marker
- Opening a Related File
- Going to a Particular Line
- Using Go Back and Go Forward
- Configuring Editor Commands
- Opening a File’s Directory (Mac OS)

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 189.

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 “Editor bindings” on page 257.

Finding a Routine

Click the Routine icon to display the Routine pop-up menu, discussed in “The Routine pop-up menu” on page 125, 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 like book-
marks. They 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**

To add a marker, move the text insertion point to the location in the text you want to mark, then choose Add marker from the Marker Pop-Up Menu. A dialog box named Add Marker appears, shown in Figure 5.9.

**Figure 5.9 Add Marker dialog box**

![Add Marker dialog box](image)

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.

When you are through adding your text note, click Add and your marker will be visible in the Marker Pop-Up Menu, as shown in Figure 5.10.

**TIP:** If you select some text in a source file, then choose Add Marker, the selected text will appear as the new marker name in the Add Marker dialog. This is handy for quickly adding specific routines or lines as markers.
Figure 5.10  Example text File with a marker added

Adding Markers with #Pragmas

There is another method for marking files on a more permanent basis. For C/C++ language programs, use

```
#pragma mark myMarker
```

to leave markers in a file. For Pascal, use

```
{$PRAGMAC MARK myMarker}
```

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.

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 5.11 is shown, and you may select the marker you wish to delete. After you select the marker, click Remove to remove it permanently from the marker list. When finished, click Done to close the Remove Markers dialog box.
Editing Source Code
Navigating in Text

Figure 5.11 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 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 5.2 on page 124 to open interface or header files referenced by the current file. You can also use the Touch and Untouch commands from this pop-up.

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, type Control/Command D after selecting the file name in the active window. To learn more about this method for opening files, refer to “Opening an Existing File” on page 96.

**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 5.12. Then enter the number of the line you want to go to and select OK.

**Figure 5.12  Go to Line Number dialog box**

![Go to Line Number dialog box](image)

**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 215 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 189.
Configuring Editor Commands

The CodeWarrior IDE will allow you to customize key bindings for the editor to suit your working style. Refer to “Editor bindings” on page 257 to learn about the commands you can customize to your liking.

Opening a File’s Directory (Mac OS)

If you want to know the full directory path for an Editor window file, the Path Pop-Up Menu will show it to you.

You can open the directory that contains the file on display in the active Editor window. Just click the Mac OS Path Pop-Up Menu and choose any directory you want to open from the pop-up, as shown in Figure 5.6 on page 129.

TIP: You can press the Command key and click on the file name in the window’s title bar to view the file’s complete path.

Online References

As you’re working on a program, you often need to look up the documentation or definition of a particular routine, variable, or type. Most of the time in the CodeWarrior IDE, the documentation or definition of a symbol is readily available.

This section describes how to get started looking up documentation online and how to set up the online reference databases.

The topics in this section are:

- Finding Symbol Definitions
- WinHelp (Windows)
- QuickView (Mac OS)
- Inserting Routine (Reference) Templates (Mac OS)
- THINK Reference (Mac OS)
Finding Symbol Definitions

This method lets you find the definition of a symbol in your project source files. If the symbol isn’t defined in your project, the CodeWarrior IDE finds the symbol’s documentation in WinHelp (Windows), QuickView (Mac OS), or THINK Reference (Mac OS).

TIP: The browser also looks up definitions. See “Using the Browser” on page 213.

To look up the definition of a symbol in your code, first select the symbol name in your code. Then, choose Find Reference from the Search Menu. If you don’t want to use the Search menu you can instead Alt/Option double-click on the symbol’s name.

The CodeWarrior IDE searches all the files in your project for the definition of the symbol. If it finds the definition, it opens a message window with a list pane and a source pane.

If the definition is not included in your project files, it then searches the external on-line documentation for your host platform. If a match is found there, the application that can show the data is launched and the information displayed.

If no match is found, a beep sounds.

Multiple symbol definitions

To choose a particular definition in the list view of the Message window, double-click on it. The CodeWarrior IDE finds the definition and displays the source in the source view.

TIP: Alt/Option double-click or hold down the Alt/Option key and choose Find Definition to search only your project’s internal information. This prevents consulting an external reference application for your search.
Editing Source Code
Online References

If the CodeWarrior IDE doesn’t find the definition in your project, it searches for the symbol in WinHelp, QuickView or THINK Reference.

When the CodeWarrior IDE goes to reference information to find a definition of a symbol, the result is displayed in the Browser rather than in the Message window.

To learn more about the CodeWarrior Browser, refer to “Browser Overview” on page 189.

TIP: To return to your original location after viewing a symbol’s definition, press Shift-Control-B/Shift-Command-' (apostrophe).

Finding symbol documentation

This method looks up a symbol’s documentation in WinHelp, QuickView or THINK Reference. Use this method if you want the documentation for a symbol that is both defined in your project and has references in the on-line documentation.

First, select the symbol’s name. Next:

- (Windows) Choose Find Definition from the Search Menu.
- (Mac OS) Hold down the Option key and choose Find Definition from the Search Menu.

If you don’t want to use the Search menu you can Control/Command double-click on the symbol’s name instead.

The CodeWarrior IDE searches for the symbol in WinHelp, QuickView or THINK Reference, launching the appropriate application if it isn’t already running. If it finds the symbol, it opens up the appropriate page in the on-line documentation. If it doesn’t find the definition, it beeps.

WinHelp (Windows)

WinHelp comes with the CodeWarrior IDE.
You can use WinHelp to look up documentation on Win32, MFC, the ANSI libraries, and the CodeWarrior IDE. In addition, you can Right-click on many components in the IDE to view pop-up help information.

**QuickView (Mac OS)**

QuickView comes with the CodeWarrior IDE and is also part of the *Macintosh Programmer's Toolbox Assistant* (MPTA) published by Addison-Wesley for Apple Computer.

You can use QuickView to look up documentation on PowerPlant, the ANSI libraries, and the CodeWarrior IDE.

If you have the *Macintosh Programmer's Toolbox Assistant*, you can also look up documentation on Mac OS Toolbox routines.

To use QuickView with the CodeWarrior IDE, it is necessary to first setup all the QuickView files properly.

1. **Position your files.**
   Place all your QuickView files in the same folder as the QuickView application that comes with the CodeWarrior IDE. If you already have the Macintosh Programmer’s Toolbox Assistant (MPTA), copy all its QuickView files into the same folder that your CodeWarrior QuickView files are in.

2. **Remove multiple copies of QuickView application.**
   Delete any other QuickView applications on your computer system. Keep the QuickView application that came with the CodeWarrior IDE and delete all others.

3. **Select QuickView for on-line reference.**
   Refer to “IDE Extras Panel” on page 246 to learn how to enable the QuickView application as the on-line reference database for your project.

   Refer to “Inserting Routine (Reference) Templates (Mac OS)” on page 151 or “Finding Symbol Definitions” on page 148 to begin looking up documentation on-line.
THINK Reference (Mac OS)

To use THINK Reference with the CodeWarrior IDE, refer to “IDE Extras Panel” on page 246 to enable the THINK Reference application as the on-line reference database for your project.

You must have the THINK Reference database product properly installed on your computer system. THINK Reference is not part of the CodeWarrior product. THINK Reference is available from Developer Depot on the MacTech CD-ROM and includes several databases on Mac OS Toolbox routines.

Refer to “Inserting Routine (Reference) Templates (Mac OS)” on page 151 or “Finding Symbol Definitions” on page 148 to begin looking up documentation on-line.

Inserting Routine (Reference) Templates (Mac OS)

If you’re looking up a routine (such as an operating system call) in the on-line reference database, you can have the template for the call pasted into your Editor window at the text insertion point. This is useful when you know the name of the call you want to add to your source code, but you don’t know what it’s parameters are supposed to be.

A routine template looks like this:

```
    SetRect(r, left, top, right, bottom);
```

To insert a reference template into your code, type the routine name that you want to insert, then select the name you just typed. Finally, choose Insert Reference Template (Mac OS) from the Edit Menu.

The CodeWarrior IDE searches for the routine in either QuickView (Mac OS) or THINK Reference (Mac OS), starting the required application if it isn’t already running. If the routine is found, the template is copied into your Editor window and replaces the text you selected with the template. If the definition is not found, you will hear a beep.
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, like as those used in UNIX’s `grep` command.

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 161, is a versatile feature of the CodeWarrior IDE. To show the Find dialog box, choose the Find command in the Search Menu. With this dialog box you do text searching through a single file or multiple files in your project. You can search and replace text strings and text sub-strings (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:

- **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 checkbox**
- **Ignore Case checkbox**
- **Entire Word checkbox**
- **Regexp checkbox**

**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 356.
Also, the discussion “Enter ‘Find’ String” on page 362 tells how to enter text into the Find Text Box without using the Find dialog box.

**Figure 6.1** The Find Dialog search and replace section

![Find Dialog](image)

**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 356.

Also, the discussion “Enter ‘Replace’ String” on page 362 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 Find text box and the Replace text box. Selecting an item in one of these popups enters it in the corresponding text box.

**Figure 6.2 Recent Strings pop-up menu**

**Find button**

The Find button shown in Figure 6.1 on page 155, 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 164.

**Replace button**

The Replace button is one of the buttons in the Find dialog box, shown in Figure 6.1 on page 155.

When you enter text in the Replace text box and click the Find button, 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.
Searching and Replacing Text

Guided Tour of the Find Dialog Box

To learn more about searching and replacing text, see “Replacing Found Text” on page 170.

Replace & Find button

The Replace & Find button is shown in Figure 6.1 on page 155. 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 166 and “Searching and Replacing Text in Multiple Files” on page 173.

Replace All button

The Replace All button is shown in Figure 6.1 on page 155. 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 170.

Batch checkbox

The Batch checkbox is shown in Figure 6.1 on page 155. Selecting this checkbox causes the results of the Find command to appear in a Search Results message window (Figure 6.7 on page 173).

To learn more about the role of the Batch checkbox in searching, see “Using Batch Searches” on page 172.

Wrap checkbox

The Wrap checkbox is shown in Figure 6.1 on page 155. This checkbox causes a 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 168.
Searching and Replacing Text
Guided Tour of the Find Dialog Box

Ignore Case checkbox

The **Ignore Case** checkbox is shown in Figure 6.1 on page 155. This checkbox 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 169.

Entire Word checkbox

The **Entire Word** checkbox is shown in Figure 6.1 on page 155. This checkbox 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 169.

Regexp checkbox

The **Regexp** checkbox is shown in Figure 6.1 on page 155. This checkbox 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 182.

Multi-File Search Disclosure triangle

The Multi-File Search Disclosure triangle is shown in Figure 6.1 on page 155. Click this triangle to expose the **Multi-File Search Section** of the **Find** window, so that the window looks as shown in Figure 6.4 on page 161, or Figure 6.3 on page 159.

To learn more about multi-file searching using the **Find** window, see “Searching and Replacing Text in Multiple Files” on page 173.
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 156, 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 174.

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 list
- Project pop-up menu
- Stop at End of File checkbox
- Sources checkbox
- System Headers checkbox
- Project Headers checkbox
- Others button

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. Use this pop-up menu to select, add, and remove saved sets of files to search and replace.

You can build up sets of files 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 175.

File list

The File 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 enabling the Sources, System Headers, Project Headers, and Others controls. You can also drag and drop groups or files from the Project window into the list.

For more information about adding files and removing files in file sets, see “Choosing Files to be Searched” on page 175.
Figure 6.4 The Find Dialog Box for a multiple file search

Project pop-up menu

The Project pop-up menu, shown in Figure 6.6, allows you to choose the project file with 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.

Stop at End of File checkbox

If you turn off the Stop at End of File checkbox, all the files in the File 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 checkbox. When the CodeWarrior IDE reaches the end of a file, it stops searching and 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 checkbox, see “Controlling Search Range” on page 168.

Figure 6.5 File Sets pop-up menu

Sources checkbox

The Sources checkbox is shown in Figure 6.4 on page 161. This checkbox adds all the source files from the current project to the File list.
For more information about source files in file sets, and their role in Multi-file searches, see “Adding project source files” on page 175.

**Figure 6.6  Project Pop-up Menu**

![Project Pop-up Menu](image)

**System Headers checkbox**

The System Headers checkbox is shown in Figure 6.4 on page 161. This checkbox adds all the system header or interfaces files from the current project to the File list.

For more information about system headers in file sets, and their role in multi-file searches, see “Adding system header files” on page 176.
Searching and Replacing Text

Searching for Selected Text

Project Headers checkbox

The Project Headers checkbox is shown in Figure 6.4 on page 161. This checkbox adds all the project header or interface files from the current project to the File list.

For more information about project headers in file sets, and their role in Multi-file searches, see “Adding project header files” on page 176.

Others button

The Others button is shown in Figure 6.4 on page 161. This button and its checkbox allows you to add one or many additional files to the File list.

For more information about adding file to file sets, see “Adding and removing arbitrary files” on page 176.

Searching for Selected Text

The IDE 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 136.

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.
Searching and Replacing Text
Searching for Selected Text

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. You can also press the keyboard shortcut Ctrl/Command F for Find and F3/Command-G for Find Next.

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

**Mac OS** To search toward the beginning of the file for the previous occurrence of the text string, hold down the Shift key and 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.

**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.
Searching and Replacing Text in a Single File

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. You can also press the keyboard shortcut Ctrl/Command F for Find and F3/Command-G for Find Next.

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

Mac OS To search toward the beginning of the file for the previous occurrence of the Find string, hold down the Shift key and 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 96.

If you haven’t yet created a file, see “Creating a New File” on page 95.

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 6.2 on page 156.

**NOTE:** To learn how to enter search text containing a Return or Tab character, refer to “Searching with Special Characters” on page 169.

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 168.

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 169.

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 174.

**Windows** 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.

**Mac OS** 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 by pressing the Shift key and choosing the Find Next menu command if you want to search backwards from the text insertion point.
Searching and Replacing Text
Searching and Replacing Text in a Single File

point in the file. The CodeWarrior IDE 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.

Windows  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.

Mac OS  To continue searching toward the beginning of the file for the previous occurrence of the Find text box string, hold down the Shift key and 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 170.

Controlling Search Range

The Wrap checkbox option in the Find dialog box controls what happens when you reach the beginning or end of a file in a search.

For example, if the text insertion point is somewhere in the middle of a source file in the active editor window, and the Wrap checkbox option is enabled. When you choose Find Previous on the Search Menu, the CodeWarrior IDE searches from the the insertion point to the start of the file, then continues the search from the end of the file to the insertion point. 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 checkbox 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 checkbox 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 checkbox**

The Ignore Case checkbox is shown in Figure 6.1 on page 155. This checkbox 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 checkbox**

The Entire Word checkbox is shown in Figure 6.1 on page 155. This checkbox 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 with Special Characters**

To enter a Tab or Return character in the Find or Replace fields, use one of the following methods:

- Cut and paste your selected text with the Tab or Return characters into the Find or Replace field, or
Searching and Replacing Text

Searching and Replacing Text in a Single File

- If drag and drop is supported, drag the text from the Editor window directly into a field, or
- Enable the Regexp option and enter \t for Tab or \r for Return into the field.

Mac OS  To directly enter a Tab character, press Option-Tab. To enter a Return character, press Option-Return.

WARNING!  Using Regexp will alter the manner in which CodeWarrior locates a string match. See “Using Regular Expressions (grep)” on page 182 for more information on using Regexp.

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 167.

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.
TIP: If you are going to perform a Replace All operation on a single source file, make sure to save the source file before executing the replace operation. In the event that you should change your mind, and before you save any changes, use Revert to replace the modified file in memory with the saved version on disk. This technique will not work across multiple files.

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 167.

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 6.2 on page 156) contains the last five strings you have used.

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.

**Mac OS** To replace the Find text box string and find the previous occurrence, hold down the Shift key as you choose Replace & Find Previous (Mac OS) from the Search Menu, or press the Shift key as you click the Replace & Find button in the Find dialog box. The Editor replaces the selected text with the Find text box string and searches for a previous occurrence of the Find text box string. If the CodeWarrior IDE 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 6.7.

The Search Results window shown in Figure 6.7 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 316.
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 215.
Searching and Replacing Text
Searching and Replacing Text in Multiple Files

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 6.8 on page 175, 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 6.8.

To learn how to configure the Multi-File Search Section of the Find dialog box, refer to “Choosing Files to be Searched” on page 175, “Saving a File Set” on page 179, “Removing a File Set” on page 180, and “Controlling Search Range” on page 181.
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 checkbox. When you turn off the Sources checkbox, the CodeWarrior IDE removes all associated files from the file list.

To include only some of the files, turn on the Sources checkbox and delete the files you don’t want by selecting them and pressing Backspace/Delete after clicking on the file name.
Searching and Replacing Text
Searching and Replacing Text in Multiple Files

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 303.

Adding project header files

To add all the project header or interfaces files from the current project, turn on the Project Headers checkbox. When you turn off the Project Headers checkbox, the CodeWarrior IDE removes all associated files from the file list.

To include only some of the files, turn on the Project Headers checkbox and delete the files you don’t want by selecting them and pressing Backspace/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 303.

Adding system header files

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

To include only some of the files, turn on the System Headers checkbox 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 303.

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 6.9. 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.
**Searching and Replacing Text**

*Searching and Replacing Text in Multiple Files*

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

**Mac OS**  Alternatively, just drag files from the Finder to the File dialog box. Just drag individual or groups of files or complete folders to the Multi-file Search list.

**Figure 6.9  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.

**Windows**  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.

**Mac OS**  To add a file to the search list, select it from the file list and click the Add button. The file is removed from the top file list and appears in the Select Files To Search list.
To add all files in the top file list, click the Add All button. The files are removed from the top file list and reappear in the Select Files To Search list.

To remove files from the Select Files To Search list, select them and click the Remove button.

To remove all the files from the Select Files To Search list, click the Remove All button.

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

When you’re finished choosing files, click Done. All files in the Select Files To Search list appear in the file set. If you change your mind, click Cancel. 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 6.5 on page 162. The files then appear in the File Sets list.

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 IDE displays the Save File Set dialog box shown in Figure 6.11.

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

![Figure 6.11 The Save File Set dialog box](image)

To choose which projects can use this file set, click either the Global or Specific radio buttons in the dialog box. If you plan to use this file set only with the current project, click Specific to this project. The CodeWarrior IDE stores the file set in the project.
Searching and Replacing Text
Searching and Replacing Text in Multiple Files

If you think you’ll use this file set with other projects, click 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 OK 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 dialog box. The CodeWarrior IDE displays the dialog box shown in Figure 6.12.

Select the file set you want to remove, then click the Remove button. The CodeWarrior IDE removes the file set so that future searches do not use the deleted file set. When you are finished removing file sets, click the Done button to return to the Find dialog box. If you change your mind about removing the file set, click Cancel instead.

Figure 6.12   Remove File Sets 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.

To treat all the files in the file set as one large file, turn off the Stop at End of File checkbox. 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 checkbox. 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.

**Windows** 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.

**Mac OS** You must choose Find in Next File or Find in Previous File (Mac OS) 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.

**TIP:** (Mac OS) After you have started searching through one file of a multi-file search, you can search in the previously-searched file in the file list by holding down the Shift key as you choose Find in Previous File from the Search menu. This effectively allows you to go backwards in your search into previously-searched files.
To learn more about text searching, see “Searching for Selected Text” on page 164, or “Searching and Replacing Text in Multiple Files” on page 173.

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.

### Special Operators

The following characters have special meanings depending upon their placement in the regular expression.

<table>
<thead>
<tr>
<th>MetaCharacters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;.&quot;</td>
<td>The <strong>match-any-character operator</strong> matches any single printing or non-printing character except newline and null.</td>
</tr>
<tr>
<td>&quot;*&quot;</td>
<td>The <strong>match-zero-or-more operator</strong> repeats the smallest preceding regular expression as many times as necessary (including zero) to match the pattern.</td>
</tr>
</tbody>
</table>
### MetaCharacters

<table>
<thead>
<tr>
<th>MetaCharacter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&quot;+&quot;</code></td>
<td>The <strong>match-one-or-more</strong> operator repeats the preceding regular expression at least once and then as many times as necessary to match the pattern.</td>
</tr>
<tr>
<td><code>&quot;\+&quot;</code></td>
<td></td>
</tr>
<tr>
<td><code>&quot;?&quot;</code></td>
<td>The <strong>match-zero-or-one</strong> operator repeats the preceding regular expression at least once or not at all.</td>
</tr>
<tr>
<td><code>&quot;\?&quot;</code></td>
<td></td>
</tr>
<tr>
<td><code>&quot;{}&quot;</code></td>
<td><strong>Interval</strong> operators repeat the smallest possible preceding regular expression a specified number of times.</td>
</tr>
<tr>
<td><code>&quot;\n&quot;</code></td>
<td>The <strong>back-reference</strong> operator matches a specified preceding group. The digit ( n ) must range between 1 and 9. Groups are identified by counting the number of open parenthesis &quot;(&quot;.</td>
</tr>
<tr>
<td><code>&quot;\s&quot;</code></td>
<td>The <strong>alternation</strong> operator matches one of a choice of regular expressions. If you place the alternation operator between any two regular expressions, the result matches the largest union of strings that it can match.</td>
</tr>
<tr>
<td><code>&quot;^&quot;</code></td>
<td>The <strong>match-beginning-of-line</strong> operator matches the string from the beginning of the string or after a newline character. When it appears within brackets the &quot;^&quot; represents a not action.</td>
</tr>
<tr>
<td><code>&quot;$&quot;</code></td>
<td>The <strong>match-end-of-line</strong> operator matches the string either at the end of the string or before a newline character in the string.</td>
</tr>
<tr>
<td><code>[...]</code></td>
<td><strong>List</strong> operators enable you to define a set of items to use as a match. The list items must be enclosed within square brackets. Note, that you cannot define an empty list.</td>
</tr>
<tr>
<td><code>[^...]</code></td>
<td></td>
</tr>
</tbody>
</table>
Searching and Replacing Text
Using Regular Expressions (grep)

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

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>.*</td>
<td>.*</td>
<td>dog</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>ABCDEFG</td>
</tr>
</tbody>
</table>

Matching any character

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

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>.art</td>
<td>dart</td>
<td>art</td>
</tr>
<tr>
<td>cart</td>
<td>hurt</td>
<td>tart</td>
</tr>
<tr>
<td>tart</td>
<td>dark</td>
<td></td>
</tr>
</tbody>
</table>
Searching and Replacing Text

Using Regular Expressions (grep)

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:

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>a+b</td>
<td>ab</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>aaab</td>
<td>baa</td>
</tr>
<tr>
<td>a*b</td>
<td>b</td>
<td>baa</td>
</tr>
<tr>
<td></td>
<td>ab</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aaab</td>
<td></td>
</tr>
<tr>
<td>.*cat</td>
<td>cat</td>
<td>dog</td>
</tr>
<tr>
<td></td>
<td>9393cat</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the old cat</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c7sb@#puiercat</td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ab)*c</td>
<td>abc</td>
<td>aabbc</td>
</tr>
<tr>
<td></td>
<td>ababababc</td>
<td>abaac</td>
</tr>
<tr>
<td>(.a)+b</td>
<td>xab</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>ra5afab</td>
<td>gaab</td>
</tr>
</tbody>
</table>
Choosing one character from many

A string of characters enclosed in square brackets ([ ]) matches any one character in that string. To match any character not enclosed within brackets, precede the enclosed expression with a caret (^) like this [ ^abc ].

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ abc ]</td>
<td>a</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>z</td>
</tr>
<tr>
<td>[ ^abc ]</td>
<td>x</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>z</td>
<td>c</td>
</tr>
</tbody>
</table>

Placing a minus sign (-) within square brackets indicates a range of consecutive ASCII characters. For example, [0–9] is the same as [0123456789]. If the minus sign is the first or last character within the enclosed string, it loses its special meaning and is treated as an ordinary character.

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ a–c ]</td>
<td>a</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>z</td>
</tr>
<tr>
<td>[ ^a–c ]</td>
<td>x</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>z</td>
<td>c</td>
</tr>
<tr>
<td>[-ab][ ^ab–]</td>
<td>az</td>
<td>xa</td>
</tr>
<tr>
<td></td>
<td>by</td>
<td>yb</td>
</tr>
<tr>
<td></td>
<td>–x</td>
<td>z–</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[aeiou][0-9]</td>
<td>a6</td>
<td>ex</td>
</tr>
<tr>
<td></td>
<td>i3</td>
<td>9a</td>
</tr>
<tr>
<td></td>
<td>u2</td>
<td>$6</td>
</tr>
<tr>
<td>[^cfl]og</td>
<td>dog</td>
<td>cog</td>
</tr>
<tr>
<td></td>
<td>bog</td>
<td>fog</td>
</tr>
<tr>
<td>END[ . ]</td>
<td>END.</td>
<td>END; END DO</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>^(the cat).+</td>
<td>the cat runs</td>
<td>see the cat run</td>
</tr>
<tr>
<td>.+(the cat)$</td>
<td>watch the cat</td>
<td>the cat eats</td>
</tr>
</tbody>
</table>

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-
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 within 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:

<table>
<thead>
<tr>
<th>This expression...</th>
<th>matches this...</th>
<th>but not this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ab)\1</td>
<td>abab</td>
<td>abc</td>
</tr>
<tr>
<td>(ab.)\1</td>
<td>abcabc</td>
<td>abcab1</td>
</tr>
<tr>
<td></td>
<td>ab1ab1</td>
<td>abab</td>
</tr>
</tbody>
</table>

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.

**References**

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

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:

- “Configuring IDE Overview” on page 223 to learn how to display the dialog box containing the option that activates the browser
- “Choosing Target Settings” on page 275 for an overview of setting target-specific IDE preferences, including browser activation, in the IDE
- “Activate Browser” on page 288 for details about the target-specific preferences setting that activates the browser.

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 213.

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 195 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 211 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

<table>
<thead>
<tr>
<th>viewing style</th>
<th>wide focus</th>
<th>narrow focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>comprehensive</td>
<td>catalog</td>
<td>not applicable</td>
</tr>
<tr>
<td>class-based</td>
<td>multi-class browser</td>
<td>single-class browser</td>
</tr>
<tr>
<td>inheritance-based</td>
<td>multi-class hierarchy</td>
<td>single-class hierarchy</td>
</tr>
</tbody>
</table>

The browser-related menu commands in the Window Menu—Browser Catalog Window, Class 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 196 for details on the catalog window interface.

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 197 and “Single-Class Browser Window” on page 203 for details on the browser view interface.

Figure 7.2  A browser view (multi-class)
The class browser view has a toolbar. To learn how to use and customize the toolbar, see “Customizing Toolbars” on page 262.

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.

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 Browser 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**

Figure 7.4  A catalog window

Category pop-up menu

The Category pop-up menu in the Catalog window controls the current type of information on display in the Symbols pane. The currently-selected item is marked with a bullet.
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 (Figure 7.2) 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.

The Multi-Class Browser window also has a feature for zooming in on one of the panes in the window. Simply click the zoom box in the pane you are interested in to resize it to the entire window, hiding the other panes. Clicking the zoom box again restores the window to its previous state.

**TIP:** The class browser view has a toolbar. To learn how to use and customize the toolbar, see “Customizing Toolbars” on page 262.

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.

**Windows**  Use the mouse to select a different pane.

**Mac OS**  Use the mouse or press Option-Tab to move to a different pane.

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.

**Mac OS**  Use the Control-Tab key combination to cycle through list items alphabetically.

The items in this window include:

- **Browser Toolbar**
- **Pane Zoom Box**
- **Resize bar**
- **Classes pane**
- **List button**
- **Member Functions pane**
- **Data Members pane**
- **Source pane**
- **Orientation button**
- **VCS Pop-up**
- **Open File button**
- **Identifier icon**
Figure 7.5 The multi-class browser in vertical orientation

Browser Toolbar

The toolbar provides easy single-click access to many CodeWarrior commands. The buttons that appear in the Browser window’s toolbar include:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄️</td>
<td>Go Back</td>
</tr>
<tr>
<td>🔄️</td>
<td>Go Forward</td>
</tr>
<tr>
<td>📦️</td>
<td>Show Catalog window</td>
</tr>
</tbody>
</table>
**Browsing Source Code**

*Guided Tour of the Browser*

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Show Hierarchy window</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>New Class Browser</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Save Default window</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Save file</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Switch to MW Debugger</td>
</tr>
</tbody>
</table>

**Pane Zoom Box**

Use the Pane Zoom box to enlarge a pane to fill the Browser window, or shrink it to its original size.

**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.

**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/Option click a disclosure triangle to open all subclasses at all levels. This is a “deep” disclosure. Ctrl/Command click to
open a single level of subclass in a class and all of its siblings at the same level. This is a “wide” disclosure.

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.

**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.

![Click this button to switch to an alphabetical list.](image)

![Click this button to switch to a hierarchical list.](image)

**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 superclass, but alphabetically within each class.
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.

TIP: If you Alt/Option click an item in the Member Functions pane or the Data Members pane, that item is entered into the Source pane text at the current insertion point. This is a neat way to enter routine calls or variable names into the Source pane.

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.

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 7.2, or vertically as in Figure 7.5.

Click this button to switch to a horizontal orientation.

Click this button to switch to a vertical orientation.

VCS Pop-up

This pop-up allows you to use revision control with the source file you are viewing. To learn more about how to use this feature, refer to the discussion in “Using Source Code Control with Files” on page 337.
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. The Open File button uses the CodeWarrior Editor to open the file.

Identifier icon

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
<th>The member is…</th>
</tr>
</thead>
<tbody>
<tr>
<td>🅺</td>
<td>static</td>
<td>a static member</td>
</tr>
<tr>
<td>🅽</td>
<td>virtual</td>
<td>a virtual function that you can override, or an override of an inherited function</td>
</tr>
<tr>
<td>🅽</td>
<td>pure virtual</td>
<td>a member function that you must override in a subclass if you want to create instances of that subclass</td>
</tr>
</tbody>
</table>

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 7.6 shows the window.

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

- Use the Context Pop-Up Menu in the Catalog View when classes or functions are displayed
- Double-click a class name in a Multi-Class Hierarchy Window
- Double-click a class name in a Single-Class Hierarchy Window
Browsing Source Code
Guided Tour of the Browser

- Use the Context Pop-Up Menu in the Multi-Class Browser Window

Figure 7.6 The single-class browser window

TIP: If you select a class in the Classes pane in the Multi-Class Browser Window, you can use the Enter/Return key to open a browser window for the selected class.

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 197. 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 in a box, and is connected to other related classes by lines. Choose Browser Catalog Window from the Window Menu to display the Multi-Class Hierarchy window. Figure 7.7 shows the 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/Return 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
Figure 7.7  The Multi-Class Hierarchy Window

**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.
NOTE: Alt/Option click a disclosure triangle to open all subclasses at all levels. This is a “deep” disclosure. Ctrl/Command click to display subclasses for a class and all of its siblings. This is a “wide” disclosure. Ctrl-Alt-click/Command-Option-click to open both wide and deep. You can use Ctrl-Alt-click/Command-Option-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 7.8 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 7.8  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 197.

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 7.9 shows this window.
You open a Symbol window by clicking and holding on a symbol name in any browser or editor window for which there is information in the browser database. When you do, a Context Pop-Up Menu appears. If the item 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/Option-double-click or Ctrl-click/Command-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 197. 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. The Symbols pane lists all versions of a symbol 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 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 7.10.

You can even insert a routine template at the location if you wish.
Other menus for other kinds of symbols have items of a similar nature.

**TIP:** To find and enter a browser item for a piece of text you’ve selected or just entered, click the text and wait for the Context pop-up menu to appear. The menu will offer a list of matching items. Choosing one of these items will enter it for you. See “Completing Symbols” on page 217 for other ways to type browser items.

**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
- Identifying Symbols in the Browser Database
- Navigating Code in the Browser
- Browsing Across Subprojects
- Completing Symbols
- Opening a Source File
- Seeing a Declaration
- Seeing a Routine Definition
- Editing Code in the Browser
- Analyzing Inheritance
- Finding Functions That Are Overrides
- Saving a Default Browser
- Windows MFC Class Viewing
- 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 190 for information on how to turn the browser on.

When the browser is on, browser-related menu items are enabled. These are the Browser Catalog Window, Class Hierarchy Window, and New Class Browser items in the Window Menu.
Browsing Source Code

Using the Browser

**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.

To tell the IDE to include items from a project’s subprojects in its browser windows, see “Cache Subprojects” on page 289.

To learn how to modify these settings, see “**Browser Coloring**” on page 233 or “**Context Pop-up Delay**” on page 238.

Identifying Symbols in the Browser Database

There is an easy way to know whether or not a symbol is in the browser database without clicking, holding and waiting to see if a Context Pop-Up Menu appears: use browser coloring. If the browser is activated, symbols that are in the browser database will be displayed in the colors you select in editor and browser windows. See “**Browser Coloring**” on page 233 for more information.

**TIP:** The factory default color setting is identical for all of the eight types of browser-database symbols. You can choose a different color for each symbol type if you like. However, if you have also enabled syntax coloring for your code, you may find it easier to identify symbols in the browser database using only one or two colors.

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 211.

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 toolbar, 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 a CodeWarrior toolbar, pressing and holding the mouse button down will show a pop-up menu of the actions you have performed (Figure 7.11 on page 216).
Browsing Source Code
Using the Browser

Figure 7.11  Go Back and Go Forward toolbar buttons

Windows  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.

Mac OS  The underlined item is your current position in the queue. Choose any item from the menu to go to that action. If you choose an action out of sequence (for example, choose LEdit-Field) CodeWarrior will go to that action without going through any previous action. The CodeWarrior IDE can track up to 100 actions.

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.

Browsing Across Subprojects

The IDE normally only displays browser items for the current target. To include browser information from the current target’s subprojects in the browser’s views, turn on subproject caching. For more information, see “Cache Subprojects” on page 289.
Completing Symbols

The IDE has commands to complete your typing for you when enter the name of an item that the class browser is aware of. Use the **Find symbols with prefix**, **Find symbols with substring**, **Get next symbol**, and **Get previous symbol** keyboard commands to find and choose browser items that match the text you’ve selected or just entered in a source code file. These commands are only available from the keyboard. They aren’t available in the IDE’s menus.

To enter the name of a browser item that has the same characters as the text you’ve selected or just typed, use the **Find symbols with substring** keyboard command. To enter the name of a browser item that only has the same first characters, use the **Find symbols with prefix** keyboard command.

After using the **Find symbols with substring** and **Find symbols with prefix** commands, use the **Get next symbol** and **Get previous symbol** to search among the browser symbols that match the text you’ve selected or just entered.

After you’ve found the browser item you want to enter, press the right arrow key to place the insertion point next to the item and continue typing.

**TIP:** Another way to find and enter a browser item is to click and hold down the mouse button on the incomplete text and wait for the Context pop-up menu to appear. The menu will offer a list of matching items. Choosing one of these items will enter it for you. See “Context Pop-Up Menu” on page 211 for more information.

Opening a Source File

There are some quick methods you can use to open a 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.

**Mac OS** If you are looking at a source file, and want to see the interface, or vice versa, simply type Command-Tab to see the related file.

### 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.
Press Alt/Option or Ctrl/Option, then double-click a function name in any source view to open the Symbol Window with all implementations of the function.

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 121.

**Analyzing Inheritance**

Use the Multi-Class Hierarchy Window or the Single-Class Hierarchy Window. Look for the small disclosure 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.

**Windows MFC Class Viewing**

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.pch.

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.

**Mac OS PowerPlant Class Viewing**

If you want the browser to display PowerPlant classes, you have to include PowerPlant 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 PowerPlant 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 names like `PP_ClassHeaders.pch++` or `PP_DebugHeaders.pch++`.

This creates a project-specific precompiled header. By including the source file, CodeWarrior builds the precompiled header from within your project, exposing PowerPlant 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.

To learn about saving editor windows, see “Saving Editor Window Settings” on page 132.
Configuring IDE Options

This chapter discusses the many options available in the CodeWarrior IDE’s Preferences dialog box. In addition, this chapter discusses the toolbars that appear in various IDE windows, and how to configure them.

Configuring IDE Overview

You control many features of the CodeWarrior IDE. The CodeWarrior IDE has a Preferences dialog box for handling global preferences that affect how the IDE works in all projects.

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.

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.

The CodeWarrior IDE’s toolbars are powerful and flexible feature you can use to customize CodeWarrior to your own working style, and to make your work more efficient.

The topics in this chapter include:

- Preferences Guided Tour
- Choosing Preferences
- Customizing Toolbars
Preferences Guided Tour

To open the Preferences dialog box, choose the Preferences command from the Edit Menu.

The topics in this section are:

- Preference Panels
- Dialog Box Buttons

Preference Panels

The Preferences dialog box has a hierarchical list of available panels on its left side. The panel selected from the list appears on its right side. The actual panels available to you may vary, depending upon the CodeWarrior product you are using.

The Preferences panels affect the IDE as a whole, and apply to all projects. To see a panel, select it in the list. You can use the arrow keys or click the name of the panel. Figure 8.1 shows a selected panel in the Preferences dialog box.

Each panel consists of a series of related options that you set. Once you've made your changes, you can save them, discard them, restore them, or reset them. See “Dialog Box Buttons” on page 225 for more information.

Figure 8.1 Selecting a preference panel
Dialog Box Buttons

There are several buttons in the dialog box to control how a panel’s settings 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 8.2 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 8.2 Preferences Confirmation Dialog Box

![Preferences Confirmation Dialog Box]

Factory Settings Button

The Factory Settings button causes the panel to revert to the settings that the CodeWarrior IDE uses as the defaults. If you click this but-
Configuring IDE Options
Choosing Preferences

...ton, you reset the panel to a known state. Settings in other panels are not affected by this button. Only the settings for the current panel are 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. 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 panels. To close the Preferences or Settings dialog box, click the close box in the upper left corner of the dialog box.

Choosing Preferences

This section discusses setting preferences for the IDE as a whole. 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 learn how to open and select a particular preference panel, see “Preferences Guided Tour” on page 224.

In this section we discuss each particular preference panel, and what features of the IDE are controlled by it. The panels are:

- Debugger Preferences
- Editor Preferences
- General Preferences
Debugger Preferences

In this section we discuss Debugger related preference panels, and what features of each are controlled by them. The Debugger panels include:

- Display Settings Panel
- Global Settings Panel
- MetroNub Settings Panel

Display Settings Panel

You can configure the internal CodeWarrior Debugger to conform to the way you work. The Display preferences panel is shown in Figure 8.3.

For more information on the commands contained in the Display preference panel, see Debugger User Guide, “Display” on page 119.

Figure 8.3  Debugger Display preferences panel

<table>
<thead>
<tr>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ In variable panes, show variable types by default</td>
</tr>
<tr>
<td>□ Sort functions by method name in browser</td>
</tr>
<tr>
<td>□ Attempt to use dynamic type of C++, Object Pascal and SOM objects</td>
</tr>
<tr>
<td>□ Show tasks in separate windows</td>
</tr>
<tr>
<td>□ Show variable values in source code</td>
</tr>
</tbody>
</table>

Watchpoint Hilite: [ ]
Variable Change Hilite: [ ] Default size for unbounded arrays: 100

In variable panes, show variable types by default

Shows variable types when a new variable window is opened.
Configuring IDE Options

Debugger Preferences

Sort functions by method name in browser

Changes the way C++, Object Pascal, and Java functions are sorted in the browser window’s function pane.

Attempt to use dynamic type of C++, Object Pascal and SOM objects

Displays the runtime type of C++ or Object Pascal objects; deselecting this preference displays an object’s static type only.

Show tasks in separate windows

Allows you to toggle how to display tasks.

Show variable values in source code

Enable this option to automatically display variable values in your source code. When disabled, the variable values do not appear.

Watchpoint hilite

Allows you to set the color that the debugger uses to identify a watchpoint.

Variable change hilite

Allows you to set the color that the debugger uses to identify a changed variable.

Default size for unbounded arrays

Specifies the array size to use when no size information is available.
Global Settings Panel

You can configure the CodeWarrior Debugger to conform to the way you work. This section tells how to configure the Debugger’s behavior to make debugging easier. The Debugger Settings preferences panel is shown in Figure 8.4.

For more information on the commands contained in the Settings preference panel, see Debugger User Guide, “Settings” on page 118.

![Figure 8.4 Debugger Settings preferences panel](image)

**Cache Edited Files Between Debug Sessions**

Enable this option to maintain a cache of edited files between debugging sessions. When enabled, you can specify the number of days to store the cache in the Maintain files in cache for text box.
Configuring IDE Options
Debugger Preferences

You can also reset the file caches using the **Purge Cache** control. When disabled, no file caches are maintained.

**Confirm invalid file modification dates when debugging**

For more information on this command, see the *Debugger User Guide, “Ignore file modification dates” on page 123.*

**Debug all Java class files in directory hierarchy**

This option applies when you want to debug a Java program but don’t have a project file open.

When enabled during Java debugging, this option causes the debugger to search for additional symbolic files to open in the same folder as the class file opened and all its subfolders. When disabled, only the class file opened will be debugged.

**Automatically launch applications when SYM file opened**

Automatically launches a target program when its symbolic file is opened, setting an implicit breakpoint at the program’s main entry point.

**Confirm “Kill Process” when closing or quitting**

Prompts for confirmation before aborting a process when a target program is killed.

**Stop at beginning of ‘main’ when launching applications**

When you begin debugging an application, the debugger stops at the first line of `main()`.

**Select stack crawl window when task is stopped**

Automatically brings Stack Crawl window to the front when a task is stopped.
Don’t step into runtime support code

Executes constructor code for C++ static objects normally, without displaying it in the program window.

**MetroNub Settings Panel**

The MetroNub panel for Windows is shown in Figure 8.5 and for Mac OS in Figure 8.6.

**Figure 8.5 MetroNub preference panel (Windows)**

![MetroNub Settings Panel](image)

- **Automatically Target Applications**
  
  When enabled, this option causes MetroNub to attempt to debug a previously crashed application that was not being debugged.

- **Stop for Debugger and DebugStr traps**
  
  Use this option to halt program execution whenever it encounters a Debugger or DebugStr trap. When disabled, both of these traps are ignored.
Configuring IDE Options

Debugger Preferences

Catch PowerPC Debugger and DebugStr Traps
Use this option to halt program execution whenever it encounters any PowerPC Debugger or DebugStr traps. When disabled, both of these traps are ignored.

Catch 68K Debugger and DebugStr Traps
Use this option to halt program execution whenever it encounters any 68k Debugger or DebugStr traps. When disabled, both of these traps are ignored.

Download Executable to Remote Machine (Windows)
Use this option to automatically download your executable file to the remote machine using the Port ID for serial connections and Remote IP Address for network connections. This makes debugging easier as the new code is automatically sent to the target machine for debugging. When disabled, the executable must be manually downloaded to the remote machine in order to debug.

Figure 8.6 MetroNub preference panel (MacOS)

Keep Program in Background while Stepping (Mac OS)
Use this option to keep the program being debugged behind all debugger windows. This makes it easier to observe variable changes and step through the source code without constantly switching between views. When disabled, the program operates in the fore-
ground until a breakpoint or watchpoint is encountered, then the debugger will move in front of the program.

Editor Preferences

In this section we discuss Editor related preference panels, and what features of each are controlled by them. The Editor panels include:

- **Browser Coloring**
- **Editor Settings**
- **Font and Tabs**
- **Syntax Coloring**

Browser Coloring

The Browser Coloring preferences are shown in Figure 8.7. The Browser can export its lists of symbols and their types to the editor. This enables the editor to use different colors for displaying various types of symbols. Choose **Activate Browser 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 change the color.

**Figure 8.7** Browser Coloring options
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 preferences panel is shown in Figure 8.8.

Figure 8.8 Editor Settings preferences panel

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

The following options control the color settings for the Editor.

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 using the operating system’s standard color selection window.

Background Color

Click the color area to change the background color of the Editor and Browser windows using the operating system’s standard color selection window.

Remember

These options control which settings the Editor remembers between programming sessions.

Font Preferences

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

Window Position and Size

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

To learn more about writable files, refer to “Using Source Code Control with Files” on page 337.

Selection Position

This option tells the CodeWarrior IDE to remember what text was scrolled into view, and the location of the insertion point or selection. Turn this option off if you always want the editor to go to the top of the file when it opens a window. This feature requires that your files be writable.
To learn more about writable files, refer to “Using Source Code
Control with Files” on page 337.

Other Settings

These options control other behaviors that the Editor remembers between programming sessions.

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 scroll box and jumping to the new location when you release the thumb.

MacOS Hold down the Option key while dragging the thumb to temporarily disable this option.

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 238.

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

Enable this option if you use K&R style coding conventions in your source code to better allow the CodeWarrior IDE to recognize and display function names on the Routine Pop-up menu. Disable this option if you use non-standard macros that can interfere with K&R styled code.

**NOTE:** Some macro functions will not be recognized when this option is enabled. If you encounter problems with routine names not displaying, disable this option and try again.

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 357 for more information.

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 138.

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 124.
Configuring IDE Options
Editor Preferences

Left Margin Click Selects Line

When this option is enabled, moving the mouse pointer to the left edge of an editor window turns the mouse pointer to a left-pointing arrow. Clicking the window when the mouse pointer faces left selects the line at the mouse pointer. Clicking and dragging the mouse when the mouse pointer faces left selects more than one line.

When this option is off, the mouse pointer acts normally.

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 139.

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

Context Pop-up Delay

Context Pop-up 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. Each interval represents 1/60th of a second (16.67 milliseconds).

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

To learn more information about this feature of the browser, refer to “Context Pop-Up Menu” on page 211 and “Using the Context Pop-Up Menu” on page 214.

Default Text File Format

Use the Default Text File Format pop-up to set the end-of-line conventions to use when creating new files. You can choose from three platform text formats including: DOS, Macintosh, and UNIX.
To learn about saving text files under a different text format, see “Options Pop-Up Menu” on page 126.

Font and Tabs

To change the settings for Font and Tabs you use the preference panel shown in Figure 8.9. This preference panel sets the font and tab information for the active Editor window. If you change the font with this panel for each file in your project when the file is the active Editor window, you can have the font remembered each time the file is opened. If no editor window is open, this preference applies to the CodeWarrior IDE defaults.

To change the font and tab settings for a file, make it the active Editor window, then open this preferences panel and make your changes.

Figure 8.9  Font and Tabs preferences panel

As long as you have write permissions on the file (the file is not Read-Only), your changes will be remembered. To learn more about
Configuring IDE Options
Editor Preferences

writable files, refer to “Using Source Code Control with Files” on page 337.

Tab Size is the number of spaces the CodeWarrior IDE inserts to make up a ‘tab’ character using the Tab key.

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.

Syntax Coloring

The Syntax Coloring preferences 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.

Figure 8.10 Syntax Coloring preferences panel

![Syntax Coloring preferences panel]

To enable Syntax Coloring, enable the Use Color Syntax checkbox in the top left corner of the preferences panel, as shown in Figure 8.10.

Table 8.1 describes each element of text that the CodeWarrior Editor displays in color.
### Table 8.1 Syntax coloring highlights

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Text</td>
<td>Anything that’s not a comment, keyword, or custom keyword, such as literal values, variable names, routine names, and type names.</td>
</tr>
<tr>
<td>Comments</td>
<td>Code comments. In Java, C or C++, a comment is text enclosed by /* and <em>/ or text from // to the end of the line. In Pascal, a comment is text enclosed by { and } or (</em> and *).</td>
</tr>
<tr>
<td>Keywords</td>
<td>The language’s keywords. It does not include any macros, types, or variables that you or the system interface files define.</td>
</tr>
<tr>
<td>Custom Keywords</td>
<td>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.</td>
</tr>
</tbody>
</table>

### Changing syntax highlighting colors

The CodeWarrior IDE can use 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 you use to select a color. The next time you view a text file, the CodeWarrior IDE uses the new color.

### Controlling syntax highlighting within a window

Use the Syntax Coloring item in the Options Pop-Up Menu to turn syntax coloring on or off as you view a particular file. For more information, see “Options Pop-Up Menu” on page 126.

### Using color for custom keywords

Use the Custom Keywords dialog box 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.

Click the Edit button to the right of the Custom Keyword Set you want to modify. The CodeWarrior IDE displays the Custom Keywords dialog box, shown in Figure 8.11.

**Figure 8.11 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.
Configuring IDE Options
Editor Preferences

When you’re done, click Done. The dialog box disappears. When you next view a source file, all the custom keywords you entered are colored.

**TIP:** You can also set target-specific colors for custom keywords. To learn more about how to do this, see “Custom Keywords” on page 292.

### 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 8.10 on page 240).

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 Done. The dialog box disappears. When you next view a source file, all the custom keywords you entered will be highlighted as you designated.
In this section we discuss General preference panels, and what features of each are controlled by them. The General panels include:

- **Build Settings Panel**
- **IDE Extras Panel**
- **Key Bindings Panel**

**Build Settings Panel**

The Build Settings panel, shown in Figure 8.12, provide project build customizations.

![Build Settings preferences panel (Windows)](image)

**Other Settings**

These options provide addition customization settings.

**Build Before Running**

Choose from the **Build Before Running** pop-up menu to set how the IDE should handle builds of a project before running it. You can choose from Always, Never, or Ask.
Include File Cache (Mac OS)

Use the Include File Cache option shown in Figure 8.13, to specify the upper limit on how much memory the IDE should use for caching include files and precompiled headers. If you have a lot of memory and want to use it to speed up builds, increase the number in this field.

Compiler Thread Stack

Use the Compiler Thread Stack option shown in Figure 8.12 and Figure 8.13, to specify the upper limit of stack size allocated by the IDE for compiling and linking thread support.

All builds in CodeWarrior are threaded, with all compilation and linking occurring on a thread separate from the main application thread. This setting allows you to control the size of the stack allocated for this thread.

Normally, you should not change this setting. However, if you have a large or very complex project, you can increase this setting to avoid compiler crashes.

Figure 8.13   Build Settings preferences panel (Mac OS)
Configuring IDE Options
General Preferences

Play Sound After ‘Bring Up To Date’ & ‘Make’ (Mac OS)

Use the Play Sound After option shown in Figure 8.13 to allow you to configure the sound that is played when the CodeWarrior IDE finishes a build of your project. You may select different sounds for both successful and unsuccessful build results, using the Success and Failure pop-up menus provided.

To add new sounds to this list, you must install additional sounds in your System Folder on your Mac OS computer. You use the Sound Control Panel on your Mac OS computer to change the volume the sounds will be played at.

IDE Extras Panel

The IDE Extras panel, shown in Figure 8.14, has options for remembering previously opened projects and text files.

Figure 8.14 Extras preferences panel (Windows)

Mac OS The Extras preferences panel also allows you to configure options for things such as external editors, window zooming, the Scripts Menu (Mac OS), multiprocessor compilation, and on-line reference databases.

Open Recent Menu

These options control the Recent Projects and Recent Documents settings.
Recent Projects

Enter the maximum number of projects that are shown in the Open Recent command on the File Menu into the Recent Projects text box.

Recent Documents

Enter the maximum number of files that are shown in the Open Recent command on the File Menu into the Recent Documents text box.

Use Multiple Document Interface (Windows)

Use this option to control which Windows interface style is used by the CodeWarrior IDE. Enable to use MDI (Multiple Document Interface), or disable to use FDI (Floating Document Interface).

Figure 8.15  IDE Extras preferences panel (Mac OS)

Zoom Windows To Full Screen (Mac OS)

This option allows you to configure the behavior of the zoom box in the upper right-hand corner of all Editor windows. If this option is enabled and the zoom box of an Editor window is clicked, the window will be resized to fill the entire screen. If this option is not
checked, the zoom box will cause the window to resize to a standard size.

**Use External Editor (Mac OS)**

If this option is enabled, the CodeWarrior IDE will send open document ‘odoc’ AppleEvents to AppleScript-compatible third-party text editors. To use this feature, create a folder named `Helper Apps` in the folder the CodeWarrior IDE application is in (if it doesn’t already exist), and create an alias named `External Editor` inside this folder which points to your third-party editor application.

If this option is not checked, the CodeWarrior Editor will be the default editor for your text files.

Note that the third-party editor will not be used for text editing unless the file you are working with has been added to the currently-open Project window.

To learn how to add files to the Project window, refer to “Adding Files” on page 67.

**Enable Automatic Toolbar Help (Mac OS)**

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.

**Use Script Menu (Mac OS)**

When this checkbox is clicked, the Scripts Menu (Mac OS) will be shown in the CodeWarrior IDE menu bar.

To learn more information about the Scripts Menu (Mac OS), refer to the discussion “Scripts Menu (Mac OS)” on page 387. Additional information about AppleScripts can be found in the section of this book entitled “CodeWarrior Apple Events Overview” on page 405.
Use ToolServer Menu (Mac OS)

This check box causes the Tools menu (Figure 8.17) to appear in the menu bar. For more information about using the menu, refer to “Tools Menu (Mac OS)” on page 384.

Figure 8.16 ToolServer Menu

Use BBEdit™ Extensions (Mac OS)

When this option is selected, an additional menu bar icon appears in the CodeWarrior IDE menu bar, as shown in Figure 8.17.

Figure 8.17 BBEdit Extensions Menu Icon

To learn more about the Editor Extensions menu, you can refer to “Editor Extensions Menu (Mac OS)” on page 389.

Find Reference Using (Mac OS)

This feature selects which on-line database should be used to look up references and definitions. Either the Symantec THINK Refer-
Configuring IDE Options
General Preferences

ence database (not part of the CodeWarrior product), or Quick-
View-based documents such as the Macintosh Programmer’s Toolbox Assistant (MPTA), are supported. CodeWarrior documentation in QuickView format can be found on the CodeWarrior Reference CD.

To learn more about on-line reference databases and how to use them, refer to the discussion “Online References” on page 147.

Key Bindings Panel

You can customize the keyboard shortcuts used for menu, keyboard, and editor commands in the CodeWarrior IDE. You can attach or “bind” almost any key to any command. This feature allows you to change the keys that activate many commands in the CodeWarrior IDE to those that are convenient for your working style.

For example, you may wish to change the key equivalent for Move line up from the Up-Arrow to Shift-Up-Arrow. You could change this using the key bindings feature.

You control keyboard shortcuts or key bindings in the Key Bindings preference panel shown in Figure 8.18. You can set the key bindings for menu commands, source code editor actions, and other miscellaneous actions. You can also specify special prefix keys.

The topics discussed in this section are:

- Restrictions for choosing key bindings
- What is a prefix key?
- What is the Quote Key prefix?
- Setting the Prefix Key Timeout
- Don’t Strip Command Key (Mac OS)
- Modifying key bindings
- Exporting key bindings
- Importing key bindings
- Miscellaneous bindings
Restrictions for choosing key bindings

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

- The Escape and Space keys are always invalid for key bindings.
- Function keys and the Clear key are valid for creating key bindings.

Windows Only Restrictions

- 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.

Mac OS Only Restrictions

- The Return and Tab keys require at least one of the following: Control, Command, or Shift. This restriction does not apply for the second key of a two-key sequence.
- The Command+period (Command+. ) key combination is invalid for any binding.

What is a prefix key?

Prefix keys allow you to create multiple-keystroke command keys, such as those used in 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 command key for the Save menu command to Control-X Control-S.

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 253.
Configuring IDE Options
General Preferences

What is the Quote Key prefix?

The Quote Key is a special prefix key with a very simple function. It lets you use a simple printing character as a key equivalent (without any modifier key), and still retain the ability to use that character in regular type in the editor window.

In typical use, a key equivalent involves two keys: a modifier key (such as the Control key) combined with an actual printing key. However, you are not required to have a modifier key in CodeWarrior. For example, you can assign the key for the number 1 (with no modifier) to a command.

However, if you do, you can no longer simply type a 1 into your code in the editor. It is interpreted as a command. The Quote Key prefix is the way around this conflict.

In the Key Bindings preference panel, in the prefix section, you can assign any key to be recognized as the Quote Key prefix. Despite the name, it doesn’t have to be the key that creates quote symbols in text.

If you have assigned a Quote Key prefix and you then type the Quote Key prefix, CodeWarrior will interpret the next keypress as a keystroke, not as a command. It’s that simple.

Returning to the earlier example, assume you have assigned the 1 key to a command. Assume also that you have assigned the tilde key (~) to be your Quote Key prefix. To issue the command, you press the 1 key. To type a 1 into the editor, you would type the tilde key first, then type the 1 key. (To type a tilde, you would press the tilde key twice.)

WARNING! The Quote Key only affects the next key or combination of keys that you type. You must use the Quote Key once for each bound key or combination of keys for which you want to type the equivalent character on-screen.
Configuring IDE Options
General Preferences

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 pressed. Larger values mean that the IDE will wait longer for the second key to be pressed.

This value is in “ticks” which are the same as \( \frac{1}{60} \)th of a second (16.67 milliseconds). Legal values are in the range of 1 to 999. The default is 120.

Don’t Strip Command Key (Mac OS)

Enable this option to allow key bindings to accept the Command key as a valid entry. When disabled, the Command key is stripped from the key entries.

Figure 8.18 Key Bindings preference panel (Windows)
Modifying key bindings

When you select the Key Bindings item from the list of IDE preference panels, you see the Key Bindings panel shown in Figure 8.18.

To change the key 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.

**Mac OS** If you are using a Dvorak keyboard and experiencing problems with the Command key being recognized, enable the Dvorak KCHR Support option.

![Figure 8.19 Key Bindings preference panel (Mac OS)]
To change a key binding, double-click the command you wish to change, or select it and press the Return/Enter key.

When you do, the dialog box shown in Figure 8.20 appears.

Once you see this dialog, press the key combination that you want to use for the command. For example, if you want to make the command key Control-8, you would press the Control key and the 8 key at the same time.

You may also specify an alternate key that can be used for the command. To create an additional command key, click on the Alternate rectangle so that it is selected. Then, hold down the keys that you want to use for the command. If you specify an alternate command key, either key can be used to execute the command.

The only exception to the alternate key feature is prefix keys. You cannot specify an alternate key for any prefix key.

If you wish to clear the Primary or Alternate command keys, click the Clear button to the right of the appropriate field.

**Figure 8.20  Modify Key Binding Dialog Box**

Use the Allow Auto Repeat option to make a command key repeat if you continue to press the key combination. A good key binding to use here as an example is the Find Next command. If you select 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 press them again every time.

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.

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, a standard file dialog box appears. Navigate to the place on your hard disk where you want to save the key bindings file, and click Save.

Importing key bindings

If you wish to import saved key bindings from a previously-exported file, you use the Import button. Use the standard file dialog to locate and open the key bindings file.

Miscellaneous bindings

Table 8.2 lists the default key equivalents for some miscellaneous IDE commands, and describes what each command does.

<table>
<thead>
<tr>
<th>Command</th>
<th>Default Key Binding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to header/source file</td>
<td>Ctrl/Command-Tab</td>
<td>Switches between a source and header file of the same name. See “Opening a Related File” on page 145.</td>
</tr>
<tr>
<td>Toggle balloon help</td>
<td>Shift-Ctrl/Shift-Command /</td>
<td>Toggles balloon help on and off.</td>
</tr>
</tbody>
</table>
## Configuring IDE Options

### General Preferences

Table 8.3 lists the default key equivalents for each command in the editor. To learn more about text navigation features of the Editor, refer to “Navigating in Text” on page 142.

### Editor bindings

Table 8.3 Editor key bindings

<table>
<thead>
<tr>
<th>Command</th>
<th>Default Key Binding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to previous error message</td>
<td>Alt-Ctrl/Option-Command Up Arrow</td>
<td>Moves to the previous error message in the Message window. See “Stepping Through Messages” on page 321.</td>
</tr>
<tr>
<td>Go to next error message</td>
<td>Alt-Ctrl/Option-Command Down Arrow</td>
<td>Moves to the next error message in the Message window. See “Stepping Through Messages” on page 321.</td>
</tr>
<tr>
<td>Move character left</td>
<td>Left Arrow</td>
<td>Moves the text insertion point one character to the left.</td>
</tr>
<tr>
<td>Move character right</td>
<td>Right Arrow</td>
<td>Moves the text insertion point one character to the right.</td>
</tr>
<tr>
<td>Move word left</td>
<td>Alt/Option Left Arrow</td>
<td>Moves the text insertion point one word to the left.</td>
</tr>
<tr>
<td>Move word right</td>
<td>Alt/Option Right Arrow</td>
<td>Moves the text insertion point one word to the right.</td>
</tr>
<tr>
<td>Move sub-word left</td>
<td>Control-Left Arrow</td>
<td>Moves the text insertion point to the left of the next capital letter or start of a word on the left.</td>
</tr>
<tr>
<td>Move sub-word right</td>
<td>Control-Right Arrow</td>
<td>Moves the text insertion point to the left of the next capital letter or start of a word on the right.</td>
</tr>
</tbody>
</table>
## Configuring IDE Options

### General Preferences

<table>
<thead>
<tr>
<th>Command</th>
<th>Default Key Binding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move to start of line</td>
<td>Ctrl/Command-Left Arrow</td>
<td>Moves the text insertion point to the start of the current line.</td>
</tr>
<tr>
<td>Move to end of line</td>
<td>Ctrl/Command-Right Arrow</td>
<td>Moves the text insertion point to the end of the current line.</td>
</tr>
<tr>
<td>Move line up</td>
<td>Up Arrow</td>
<td>Moves the text insertion point one line up.</td>
</tr>
<tr>
<td>Move line down</td>
<td>Down Arrow</td>
<td>Moves the text insertion point one line down.</td>
</tr>
<tr>
<td>Move to top of page</td>
<td>Alt/Option Up Arrow</td>
<td>Moves the text insertion point to the top of the current view.</td>
</tr>
<tr>
<td>Move to bottom of page</td>
<td>Alt/Option Down Arrow</td>
<td>Moves the text insertion point to the bottom of the current view.</td>
</tr>
<tr>
<td>Move to top of file</td>
<td>Command-Up Arrow</td>
<td>Moves the text insertion point to the top of the current file.</td>
</tr>
<tr>
<td>Move to bottom of file</td>
<td>Command-Down Arrow</td>
<td>Moves the text insertion point to the bottom of the current file.</td>
</tr>
<tr>
<td>Delete character left</td>
<td>Backspace/Delete</td>
<td>Erases the last character typed.</td>
</tr>
<tr>
<td>Delete character right</td>
<td>Delete/Del (Delete key)</td>
<td>Deletes the character immediately to the right to the text insertion point.</td>
</tr>
<tr>
<td>Delete to end of file</td>
<td>Ctrl-Backspace/Command-Delete</td>
<td>Deletes the characters from the text insertion point to the end of the file.</td>
</tr>
<tr>
<td>Character select left</td>
<td>Shift-Left Arrow</td>
<td>Adds the character to the left of the text insertion point to the current selection.</td>
</tr>
<tr>
<td>Character select right</td>
<td>Shift-Right Arrow</td>
<td>Adds the character to the right of the text insertion point to the current selection.</td>
</tr>
<tr>
<td>Select word left</td>
<td>Alt-Shift/Option-Shift Left Arrow</td>
<td>Selects the full word to the left of the text insertion point.</td>
</tr>
<tr>
<td>Command</td>
<td>Default Key Binding</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Select word right</td>
<td>Alt-Shift/ Option-Shift Right Arrow</td>
<td>Selects the full word to the right of the text insertion point.</td>
</tr>
<tr>
<td>Select sub-word left</td>
<td>Control-Shift-Left Arrow</td>
<td>Selects the partial word to the left of the text insertion point, up to the first capital letter or white space, whichever comes first.</td>
</tr>
<tr>
<td>Select sub-word right</td>
<td>Control-Shift-Right Arrow</td>
<td>Selects the partial word to the right of the text insertion point, up to the first capital letter or white space, whichever comes first.</td>
</tr>
<tr>
<td>Select line up</td>
<td>Shift-Up Arrow</td>
<td>Selects all text from the current location of the text insertion point to a position one line directly above.</td>
</tr>
<tr>
<td>Select line down</td>
<td>Shift-Down Arrow</td>
<td>Selects all text from the current location of the text insertion point to a position one line directly below.</td>
</tr>
<tr>
<td>Select to start of line</td>
<td>Shift-Ctrl/Shift-Command Left Arrow</td>
<td>Selects all text left of the text insertion point through the beginning of the current line.</td>
</tr>
<tr>
<td>Select to end of line</td>
<td>Shift-Ctrl/Shift-Command Right Arrow</td>
<td>Selects all text right of the text insertion point through the end of the current line.</td>
</tr>
<tr>
<td>Select to start of page</td>
<td>Alt-Shift/ Option-Shift-Up Arrow (Mac OS)</td>
<td>Selects all text left of the text insertion point up to the top of the current view of the file.</td>
</tr>
<tr>
<td>Select to end of page</td>
<td>Alt-Shift/ Option-Shift-Down Arrow</td>
<td>Selects all text right of the text insertion point down to the bottom of the current view of the file.</td>
</tr>
<tr>
<td>Select to start of file</td>
<td>Shift-Ctrl/Shift-Command Up Arrow</td>
<td>Selects all text left of the text insertion point up to the top of the file.</td>
</tr>
</tbody>
</table>
## Configuring IDE Options

### General Preferences

<table>
<thead>
<tr>
<th>Command</th>
<th>Default Key Binding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select to end of file</td>
<td>Shift-Ctrl/Shift-Command</td>
<td>Selects all text right of the text insertion point up to the top of the file.</td>
</tr>
<tr>
<td></td>
<td>Down Arrow</td>
<td></td>
</tr>
<tr>
<td>Scroll line up</td>
<td>Control-Up Arrow</td>
<td>Scrolls the view up one line.</td>
</tr>
<tr>
<td>Scroll line down</td>
<td>Control-Down Arrow</td>
<td>Scrolls the view down one line.</td>
</tr>
<tr>
<td>Scroll page up</td>
<td>PgUp (Page Up Key)</td>
<td>Scrolls the view up one page.</td>
</tr>
<tr>
<td>Scroll page down</td>
<td>PgDn (Page Down Key)</td>
<td>Scrolls the view down one page.</td>
</tr>
<tr>
<td>Scroll to top of file</td>
<td>Home</td>
<td>Scrolls the view up to the top of the file.</td>
</tr>
<tr>
<td>Scroll to end of file</td>
<td>End</td>
<td>Scrolls the view down to the bottom of the file.</td>
</tr>
<tr>
<td>Find symbols with prefix</td>
<td>Control-’ (Mac OS)</td>
<td>Enters the first browser item that is matched by first part of the selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or just-entered text.</td>
</tr>
<tr>
<td>Find symbols with substring</td>
<td>Control-Command-’ (Mac OS)</td>
<td>Enters the first browser item that is matched by any part of the selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or just-entered text.</td>
</tr>
<tr>
<td>Get next symbol</td>
<td>Control-. (Mac OS)</td>
<td>Enters the next matching browser item.</td>
</tr>
<tr>
<td>Get previous symbol</td>
<td>Control-. (Mac OS)</td>
<td>Enters the previous matching browser item.</td>
</tr>
</tbody>
</table>
Customizing Toolbars

A toolbar contains a series of elements, represented by icons, that act as buttons. Each element typically represents a corresponding menu command. When you click the element, that command is executed. There are other kinds of actions besides menu commands that can appear on a toolbar as well. Figure 8.21 shows the toolbar from the CodeWarrior Project window.

Figure 8.21   The Project window toolbar

TIP:  (Mac OS) To display a balloon help window that describes a toolbar icon, place the cursor over the toolbar element and press the Control key.

This section discusses toolbars in detail. The topics are:

- “Kinds of Toolbars” on page 263
- “Toolbar Elements” on page 264
- “Showing and Hiding a Toolbar” on page 265
- “Modifying a Toolbar” on page 267
- “Adding a Toolbar Element” on page 267
- “Rearranging Toolbar Elements” on page 268
- “Removing a Toolbar Element” on page 268
- “Removing All Toolbar Elements” on page 268
- “Restoring a Toolbar to Default Settings” on page 269
- “Anchoring the Floating Toolbar (Mac OS)” on page 269
Kinds of Toolbars

There are four toolbars in CodeWarrior:

- the floating toolbar (sometimes also referred to as the global toolbar)
- the Project window toolbar, which appears in any project window
- the Editor window toolbar, which appears not only in the editor window, but the editing pane of some other windows
- the Class Browser window toolbar, which appears in the browser’s single- and multi-class browser views

Each kind of window toolbar applies to all windows of the type. For example, there is one Editor window toolbar that appears in all editor windows. If you modify the toolbar in one window, the change affects all editor windows. For more information, see “Modifying a Toolbar” on page 267.

Each of the toolbars has a factory-default configuration of elements that you can restore at any time. For more information, see “Restoring a Toolbar to Default Settings” on page 269.

There are really two groups of toolbars: the floating toolbar that can be used at all times, and toolbars that appear in particular windows.

This distinction is important, because you show, hide, clear, and reset the toolbars using menu commands in the Toolbar Submenu, which appears in the Window menu in the CodeWarrior IDE. These commands distinguish between the floating toolbar and window toolbars.

When you choose a menu command related to a window toolbar, the toolbar in the active window is affected. For more information, see “Toolbar Submenu” on page 383.
Configuring IDE Options
Customizing Toolbars

Toolbar Elements

A toolbar may contain any of four types of different elements. The four types of elements are:

- **Commands**—buttons that execute IDE menu commands when clicked
- **Controls**—the IDE popup menu buttons (Document Settings, Function, Header, Marker, Version Control), plus the Change Current Target button
- **Miscellaneous**—other elements (such as the File Dirty and File Path indicators)
- (Mac OS) **Scripts**—buttons that execute one of the scripts available through the Scripts menu in the IDE.

**NOTE:** (Windows) Scripting functionality has not yet been added to CodeWarrior Pro for Windows, so you cannot add script elements to a toolbar.

You access individual elements through the Toolbar Elements window, shown in Figure 8.22. To display this window, choose the **Toolbar Elements Window** item from the **Toolbar Submenu** of the **Window Menu**.

Each type of element has its own tab in the Toolbar Elements window. To see the list of available elements for any particular type, click the corresponding view.

You use the Toolbar Elements window when adding new items to a toolbar. For more information on adding elements to a toolbar, see “Adding a Toolbar Element” on page 267.
Figure 8.22 The Toolbar Elements window

Showing and Hiding a Toolbar

Use Toolbar Submenu commands to show or hide a toolbar. Hiding a toolbar does not change the toolbar’s configuration of elements.

Floating Toolbar

To hide the floating toolbar, choose Hide Floating Toolbar. To display the hidden floating toolbar, choose Show Floating Toolbar.
NOTE: hiding the floating toolbar (on platforms that support it) does not change whether it is anchored or unanchored. If the floating toolbar is unanchored when you hide it, it will be unanchored the next time you display it. For more information, see “Anchoring the Floating Toolbar (Mac OS)” on page 269.

Window Toolbar

To hide a window’s toolbar, first make sure that window is the active window. Then choose Hide Window Toolbar. The other components of the window shift to fill the space previously occupied by the window’s toolbar. All subsequent windows of that type which you open will have hidden toolbars.

To display a window’s hidden toolbar, first make sure that window is the active window. Then choose Show Window Toolbar. The other components of the window shift to allow room for the window’s toolbar. All subsequent windows of that type which you open will have hidden toolbars.

Figure 8.23 The Editor window with hidden toolbar

When you hide the Editor window toolbar, the default tools appear along the bottom of the editor window, as shown in Figure 8.23.

You can also show or hide the Editor window toolbar with the Toolbar disclosure button. See “Seeing Window Controls” on page 130.
Modifying a Toolbar

You can modify a toolbar by:

- Adding a Toolbar Element
- Rearranging Toolbar Elements
- Removing a Toolbar Element
- Removing All Toolbar Elements

In certain circumstances there are limitations as to which elements you can add or remove from a toolbar. These are fully explained in “Adding a Toolbar Element” on page 267 and “Removing a Toolbar Element” on page 268.

If you modify a toolbar, the changes apply to every instance of that toolbar subsequently created. For example, if you customize the Project window toolbar, those changes will affect every Project window you open, not just the toolbar in the active Project window. Windows that are already open are not affected.

Adding a Toolbar Element

You add an element to a toolbar by dragging and dropping it from the Toolbar Elements window onto a toolbar.

Open the Toolbar Elements window, and locate the element you want to add to a toolbar. Make sure the destination toolbar is visible as well.

Drag the element from the Toolbar Elements window to the toolbar.

As you drag across the toolbar, framing corners will appear in any location within the toolbar that can accept the new element. If for some reason you cannot add this element to this particular toolbar, no framing corners will appear.

There are several reasons why a toolbar may not accept an element:

- the toolbar is full
- the element already exists on the toolbar
• commands can be added to a window’s toolbar only for menu commands that are available when the current window is the active window
• the five popups on the Controls view of the Toolbar Elements window, as well as the File Dirty and File Path indicators on the Miscellaneous view, can only be added to the Editor window toolbar
• the Change Current Target element from the Controls view can only be added to the Project window toolbar

**Rearranging Toolbar Elements**

You can re-position toolbar elements to better suit your working requirements.

To reposition an element in a toolbar, press Ctrl-Right/Control-Command click, then drag the element to a new position. That is, press the shortcut keys, then drag the item to its new location.

As you do, framing corners will appear in any location that can accept the drop.

**Removing a Toolbar Element**

You can remove toolbar elements to better suit your working requirements.

To remove an element from a toolbar, press Ctrl-Right/Control-Command and click on the element. That is, press the keyboard equivalent keys, then drag the item to its new location.

When you do, the element is removed from the toolbar.

**Removing All Toolbar Elements**

You can clear all elements from a toolbar using commands in the Toolbar Submenu of the Window menu. Clearing a toolbar is handy if you want to build your own toolbar from scratch.

To clear the floating toolbar, choose Clear Floating Toolbar.
Configuring IDE Options
Customizing Toolbars

To clear a window toolbar, make that window the active window. Then choose Clear Window Toolbar.

In some cases, certain elements may not be removed by a Clear Window Toolbar command because they are critical to the window’s basic purpose.

**NOTE:** If the floating toolbar is currently hidden, you cannot clear it. You’ll have to display the floating toolbar before clearing it. See “Showing and Hiding a Toolbar” on page 265.

Restoring a Toolbar to Default Settings

You can reset a toolbar to its original factory settings (the program defaults) using commands in the Toolbar Submenu of the Window menu.

To reset the floating toolbar, choose Reset Floating Toolbar.

To reset a window toolbar, make that window the active window. Then choose Reset Window Toolbar.

**NOTE:** If the floating toolbar is currently hidden, you cannot reset it. You’ll have to display the floating toolbar before resetting it. See “Showing and Hiding a Toolbar” on page 265.

Anchoring the Floating Toolbar (Mac OS)

The floating toolbar can be anchored to the top left corner of the screen, just below the menu bar. In its anchored state, the floating toolbar is joined to the IDE menu bar, loses the ability to be closed with a mouse-click, and cannot be moved. Unanchored, the floating toolbar can be positioned anywhere on your screen.

To anchor the floating toolbar when it is currently free-floating, choose the Anchor Floating Toolbar command from the Toolbar Submenu of the Window menu.
To release the floating toolbar when it is currently anchored, choose the Unanchor Floating Toolbar command.
Configuring IDE Options

Customizing Toolbars
Configuring Target Options

This chapter discusses the Target Settings dialog box. Target settings are options you set to specify to the IDE how it should process a target in a project.

Configuring Target Options Overview

The Target Settings dialog box handles settings that affect a particular target in a project. 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 contains settings that specify which folders the IDE should search for files listed in a target’s file view.

After you have set all the settings for a particular target, you can create Project stationery so that your choices will 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 54.

The topics in this chapter include:

- Target Settings Guided Tour
- Choosing Target Settings
Target Settings Guided Tour

To open the Settings dialog box, use the Target Settings command from the Edit Menu. The actual name of this command will include the name of the target.

The topics in this section are:

- Panels
- Dialog Box Buttons

Figure 9.1 Selecting a settings panel
Panels

The Target Settings dialog box has a hierarchical list of available panels on the left side of the dialog box. The panel selected in this list appears on the right side of the dialog box. The actual panels available to you may vary, depending upon the CodeWarrior product you are using, and the current target.

To see a panel, select it in the list. You can use the arrow keys or click the name of the panel. Figure 9.1 shows a selected panel in the Settings dialog box.

Each panel consists of a series of related options that you set. The options you set apply to the currently selected target in the active project. See “Dialog Box Buttons” on page 273 for information on applying or ignoring changes to settings panels.

After you have set all the settings for a particular target, you can create Project stationery so that your choices will 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 54.

Dialog Box Buttons

There are several buttons in the dialog box to control how a panel’s settings 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 in the Target Settings dialog box and attempt to close it, a dialog box similar to that shown in Figure 9.2 may appear. To save your changes and close the dialog box, click Save. To discard your changes and close the dialog box, click the Don’t
Save button. To continue using the dialog box without saving changes, click Cancel.

Figure 9.2 Settings Confirmation Dialog Box

```
“CoolApp” settings not saved.
Do you want to save changes to the settings for target “CoolApp” before closing them?

Don’t Save  Cancel  Save
```

Factory Settings button

The Factory Settings button causes the panel to revert to the settings that the CodeWarrior IDE uses as the defaults. If you click this button, you will reset the panel to a known state. Settings in other panels are not affected by this button. Only the settings for the current panel are 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. 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 panels. If you changed an option that will require that the project be re-compiled, you will see a dialog box similar to that shown in Figure 9.3. Click OK, or Cancel depending on whether you want to keep your changes or not.
Choosing Target Settings

This section discusses setting options for a particular target. You can change many different settings to configure the CodeWarrior IDE to build your target the way you want.

To learn how to open the Settings dialog box and select a particular settings panel, see “Target Settings Guided Tour” on page 272.

There are many settings panels available in CodeWarrior Professional and other Metrowerks products. The particular panels that appear in the Settings dialog box depend upon the particular operating system or chip family you have set as your target, and the programming language available with your Metrowerks product.

For example, if you are working with an x86 project, you will not see any panels for Motorola 68K configuration. If the Pascal language is not available for a particular target, panels related to Pascal will not appear.

In addition, the IDE takes great care to ensure that only the files affected by a preference option are marked for recompilation. For example, changing a preference in the resource compiler only marks
Configuring Target Options
Choosing Target Settings

resource compiler sources as dirty. This reduces the amount of re-compilation required during the next make operation and increases linking speed.

This manual does not discuss panels that are specific to a particular operating system, microprocessor family, or language. Table 9.1 lists the various manuals where you can learn the details of particular OS-, processor-, or language-specific settings panels. This list is not exhaustive, and does not include every target or panel supported by Metrowerks in various CodeWarrior products. This list represents the panels provided with CodeWarrior Professional.

While some panels are specific to a particular language, operating system, or microprocessor family, other panels control options that are pertinent to any target. These panels are discussed in detail in this section. They are:

- Target Configurations
- Editor Configurations
### Table 9.1 Where to learn about specific settings panels

<table>
<thead>
<tr>
<th>For these panels</th>
<th>See this manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>68K Target</td>
<td>Targeting Mac OS</td>
</tr>
<tr>
<td>68K Processor</td>
<td></td>
</tr>
<tr>
<td>68K Linker</td>
<td></td>
</tr>
<tr>
<td>CFM68K Linker</td>
<td></td>
</tr>
<tr>
<td>68K Disassembler</td>
<td></td>
</tr>
<tr>
<td>Rez</td>
<td></td>
</tr>
<tr>
<td>PPC Target</td>
<td>Targeting Mac OS</td>
</tr>
<tr>
<td>PPC Processor</td>
<td></td>
</tr>
<tr>
<td>PPC Linker</td>
<td></td>
</tr>
<tr>
<td>PPC PEF</td>
<td></td>
</tr>
<tr>
<td>PPCAsm</td>
<td></td>
</tr>
<tr>
<td>PPC Disassembler</td>
<td></td>
</tr>
<tr>
<td>x86 Target</td>
<td>Targeting Win32</td>
</tr>
<tr>
<td>x86 CodeGen</td>
<td></td>
</tr>
<tr>
<td>IR Optimizer</td>
<td></td>
</tr>
<tr>
<td>x86 Linker</td>
<td></td>
</tr>
<tr>
<td>WinRC Resource Compiler</td>
<td></td>
</tr>
<tr>
<td>Java Project</td>
<td>Targeting Java</td>
</tr>
<tr>
<td>Java VM</td>
<td></td>
</tr>
<tr>
<td>Java Linker</td>
<td></td>
</tr>
<tr>
<td>C/C++ Compiler</td>
<td>C Compiler Guide</td>
</tr>
<tr>
<td>C/C++ Warnings</td>
<td></td>
</tr>
<tr>
<td>Pascal Compiler</td>
<td>Pascal Compiler Guide</td>
</tr>
<tr>
<td>Pascal Warnings</td>
<td></td>
</tr>
<tr>
<td>Runtime Settings</td>
<td>IDE User Guide</td>
</tr>
<tr>
<td>Settings</td>
<td></td>
</tr>
</tbody>
</table>
Target Configurations

The following panels apply to the CodeWarrior targets:

- Target Settings
- Access Paths
- Build Extras
- File Mappings

Target Settings

The Target Settings panel is the single most critical panel in CodeWarrior. This is the panel where you pick your target operating system and/or microprocessor.

Figure 9.4 Target Settings panel

The Target Settings panel, shown in Figure 9.4, allows you to set the name of your target, as well as which linker and post linker plugins to use for the target. When you select a linker, you are specifying the target operating system and/or chip. The other panels available in the Settings dialog box will change to reflect your choice.
Because the linker choice affects the visibility of other related panels, you must set your target first before you can specify other target-specific options like compiler and linker settings.

It is possible to change targets completely in this panel. When you change the target, you must change the libraries contained in the project file for that target. Choosing a new value for a target does not change these files for you. For this reason, you should be careful when changing the target to remove inappropriate libraries and add those required for the new target.

If you create a new project from stationery appropriate for the new target, the necessary library and support files will be included automatically. You can then add source files from the old project. For more information on creating a project based on stationery, see “Creating a New Project” on page 48.

**Target Name**

Use the Target Name text field to set or change the name of a target. When you use the Targets view in the Project window, you will be able to see the name that you have set.

This is not the name of your final output file, just the name you assign to the target for your personal use. The name of the final output file is typically set in the linker settings panel for the linker you choose for your target.

**Linker**

Choose a linker from the items listed in the Linker pop-up menu.

To learn more about the choices available to you for linker and post linker, refer to the Targeting manual for your selected target. The actual choices available to you will depend upon the plugin linkers you have for your CodeWarrior product.

For example, CodeWarrior Professional provides linkers for Mac OS on 68K and PowerPC, Win32 on x86, and the Java Virtual Machine. CodeWarrior for PlayStation OS provides a linker for the PlayStation OS running on MIPS, as well as a post-linker.
**Configuring Target Options**

**Target Configurations**

---

**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.

**Output Directory**

This is the directory where your final linked output file will be placed. The default location is the directory that contains your project file. Click the **Choose** button to specify another directory.

**Save Project Entries Using Relative Paths**

When this setting is enabled the IDE remembers the location of a project entry as a relative path from one of the access paths. This extra location information lets the IDE distinguish different source files with the same name. The IDE will remember this location even if it needs to re-search for files in the access paths.

If this setting is disabled then the IDE remember project entries by name and re-searching for files could cause the IDE to find the project entry in a different access path.

See “Re-search For Files” on page 369 and “Reset Project Entry Paths” on page 370 for more information.

**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 settings panel, shown in Figure 9.5.

You can use drag and drop to add paths to the Access Paths settings panel. Just drag the folder to the path list.

**Windows**  Remove paths by dragging them to the Recycle Bin on the Desktop.

**Mac OS**  Remove paths by dragging them to the Trash on the Desktop.
If a folder icon appears beside the name of a folder in either the User Paths pane, or the System Paths 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 Paths pane or the System Paths pane, 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 or User 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.
A folder can be made invisible to the search by enclosing its name in parenthesis. For example, the folder (Project Stationery) and all its subfolders are automatically excluded from any file search. To add it to the search list, it must be explicitly added as an access path.

**Windows** Resource.frk files are also automatically excluded from the search list.

**TIP:** You can prevent any folder and all its subfolders in an access path from being searched by renaming it with enclosing parenthesis. For example, changing GameImages to (GameImages) will exclude it from all subsequent searches.
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 Paths (Windows)

Displays the User Paths pane in the Access Paths preference panel.

System Paths (Windows)

Displays the System Paths pane in the Access Paths preference panel.
Interpret DOS and Unix paths (Mac OS)

Determines how the IDE should treat file names for interface files. When this checkbox is deselected, the IDE treats “\” and “/” characters as simply part of the file name in an interface file. If this checkbox is selected, the IDE treats these characters as sub-folder separator characters.

For example, with this checkbox deselected, the IDE treats this directive,

```
#include "sys/socks.h"
```

to mean that a file named “sys/socks.h” should be searched for. If the checkbox is selected, then the IDE looks for a sub-folder named “sys” that has a file named “socks.h”.

User Paths 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 Paths 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 Paths pane or System Paths 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.
Host Flags

Specifies on which host platform an access path may be used. To set the host platform that an access path may be used on, select it in the User Paths pane or System Paths pane, then choose the host platform that the access path will be used on.

To specify that the access path may be used no matter which platform the IDE runs on, choose All from the pop-up menu.

Figure 9.7 Access Path Selection Dialog Box

For example, selecting a path and choosing Mac OS in the Host Flags pop-up menu specifies that the IDE must search the access
path when the IDE is running on a Mac OS computer and ignore the access path when running on any other platform.

**Add**

To add a new access path, first select the System Paths pane or the User Paths pane. Then, click the Add button. The dialog box shown in Figure 9.7 appears.

You can specify how CodeWarrior stores an access path by choosing one of these path type options: 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 operating system’s base folder will be incorporated in the Access Path. You do not need to update system relative access paths if you move a project as long as the hierarchy of the relative path is the same.
Figure 9.8  Access Path Selection Dialog (Mac OS)

Windows  The base folder is the Windows folder under Windows.

Mac OS  The base folder is the System Folder on Mac OS.

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 9.7 on page 285 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 286.
Configuring Target Options
Target Configurations

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 9.9.

![Figure 9.9 Build Extras settings panel (Windows)](image)

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.

Activate Browser

Selecting this option causes the information needed by the CodeWarrior code browser to be generated the next time your
project is built. Without this information, you cannot open Browser windows for your project.

See “Making a Project” on page 303 for more information about rebuilding your project. For more information on browser settings and options, see “Browser Overview” on page 189.

**Cache Subprojects**

Selecting this option improves multiproject updating and linking. Selecting this option also allows the IDE’s class browser to include browser information from a target’s subprojects as well as the target’s own browser information.

Deselecting this option reduces the amount of memory required by the CodeWarrior IDE.

**Dump Internal Browse Information After Compile**

Select this option to view the raw browser information that a plugin compiler or linker provides for the IDE. This option’s only practical use if for developing plugins for the IDE.

**NOTE:** When this option is on, compile only single files, or small files. The information that the IDE displays when this option is on can be huge when compiling an entire project.
Use Third Party Debugger (Windows)

Enable this option to use a third party debugger in place of the CodeWarrior internal Debugger. Enter the path to the debugger application in the field provided, or click the Browser button to locate the debugger application of your choice using the standard open file dialog.

File Mappings

The File Mappings settings panel, shown in Figure 9.11, 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.

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.

To add documentation files to your project, choose the .doc Extension, delete the Filetype, delete the compiler selection, and click Add.

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.
Configuring Target Options
Target Configurations

Figure 9.11 File Mappings settings panel

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.
Configuring Target Options

Editor Configurations

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.

Editor Configurations

The following panels apply to the CodeWarrior Editor:

- Custom Keywords

Custom Keywords

The Custom Keywords settings panel, shown in Figure 9.12, 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.

For information on setting global keyword sets, see “Using color for custom keywords” on page 241. That topic also discusses the details of setting colors, specifying keywords, and importing and exporting keywords.

To change the color for a keyword set, click the color sample. To change the contents of a keyword set, click the Edit button and make the appropriate entries in the dialog box that is presented.
Global Optimizations

The Global Optimizations panel, shown in Figure 9.13, is available when targeting many platforms. Use this panel to instruct the compiler to rearrange its object code to produce smaller and faster-executing object code. Some optimizations remove redundant operations in a program, other optimizations analyze how an item is used in a program to attempt to reduce its effect on a program’s performance.

All optimizations rearrange object code without affecting the object code’s logical sequence of execution. In other words, an unoptimized program and its optimized program produce the same results.
NOTE: Use compiler optimizations only after you’ve debugged your software. Using a debugger on a program that has been optimized may affect the debugger’s source code view.

For information on finding specific information on how these settings apply to a target, see “Targeting Documentation” on page 23.

Optimize Space

Reduces the size of object code that the compiler produces.

With this option on, object code is smaller, but may be slower. Turn on Optimize Speed to reduce the effect that this option has on a program’s performance.

Common Sub-Expressions

Replaces similar redundant expressions with a single expression. For example, if two consecutive statements both use the expression a * b * c + 10, the compiler generates object code that computes the expression only once and applies the resulting value to both statements.

With this option on, object code is smaller and faster.

Copy/Const Propagation

Replaces multiple occurrences of one variable with a single occurrence.

With this option on, object code is smaller and faster.

Reduction in Strength

Replaces multiplication instructions that are inside loops with addition instructions to speed up the loop.

With this option on, object code is larger but executes faster.
**Lifetime Analysis**

Uses the same processor register for different variables in the same routine if the variables aren’t used in the same statement.

With this option on, object code executes faster.

**Optimize Speed**

Improves the execution speed of object code.

With this option on, object code is faster, but may be larger. Turn on **Optimize Space** to reduce the effect that this option has on a program’s size.

**Loop Invariants**

Moves computations that don’t change on the inside of a loop to the outside of a loop to improve the loop’s speed.

With this option on, object code is faster.

**Dead Store Elimination**

Removes assignments to a variable if the variable is not used before being reassigned again.

With this option on, object code is smaller and faster.

**Dead Code Elimination**

Removes statements that, logically, can never be executed or are never referred to by other statements.

With this option on, object code is smaller.
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 “Working with Projects Overview” on page 39, “Working with Files Overview” on page 95, “Source Code Editor Overview” on page 121, and “Configuring IDE Overview” on page 223.

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 “Working with Projects Overview” on page 39.

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 23.

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.
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 Plug-in 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, plug-in compilers are used.

Plug-ins are basically small loadable code modules that allow the IDE to have many different compilers at its disposal. For example, there are plug-in compilers for C, C++, Pascal, Java, and assembly language.

Plug-in compilers usually have default target settings to help the CodeWarrior IDE decide which project files a plug-in handles. During regular compile and link operations, the IDE assigns files to the proper plug-in 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 290.

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. 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 63.

The topics in this section are:

- Compiling Files
- Updating a Project
- Making a Project
- Enabling Debugging
- Running a Project
- Debugging a Project
- Generating a Link Map
- Synchronizing Modification Dates
Compiling and Linking

Compiling and Linking a Project

- Removing Objects
- Advanced Compile Options

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 85.

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.

**Windows Only**

- 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 10.1**).

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.
Compiling and Linking
Compiling and Linking a Project

Figure 10.1  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.

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 65.

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 77.

After touching the file or files that you want to recompile, choose Bring Up To Date or Make from the Project Menu.

Setting Link Order

You can specify the order in which files are compiled in the Links view of the Project window. By re-arranging the order of the files,
its possible to prevent linkage errors caused by file dependencies. That is, if file Aleph depends upon file Zeta already being compiled, you can position file Zeta to compile before file Aleph, preventing an error message.

To Set the Lnk Order:

1. Click the Link view tab in the Project window.
   The Links view appears in the Project window.
2. Drag files into the correct link order.
   Use drag and drop to reposition the files into the desired build order.

   The next time you Bring Up To Date, Make, Run, or Debug the project, the new build order will be used when compiling the project files.

   See “Link Order View” on page 47 for additional information.

   NOTE: (Mac OS) If you are building a 68K project, the Links tab is replaced by the Segments tab.

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 it’s output binary produced. This command only runs the compiler.

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 85.
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 10.1 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 16 to find out which guide to read for your target.

Table 10.1 Example Products for Certain Targets

<table>
<thead>
<tr>
<th>Project Type</th>
<th>Target</th>
<th>Make Creates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Win32</td>
<td>Win32 application</td>
</tr>
<tr>
<td>Library</td>
<td>Win32</td>
<td>Win32 Library (.lib)</td>
</tr>
<tr>
<td>Dynamic link library</td>
<td>Win32</td>
<td>Win32 Dynamic Link Library (.dll)</td>
</tr>
<tr>
<td>Application</td>
<td>68K/PowerPC</td>
<td>Mac OS Application</td>
</tr>
<tr>
<td>Library</td>
<td>68K/PowerPC</td>
<td>Mac OS A4 or A5 Library</td>
</tr>
<tr>
<td>Code resource</td>
<td>68K/PowerPC</td>
<td>Code Resource</td>
</tr>
<tr>
<td>MPW tool</td>
<td>68K/PowerPC</td>
<td>MPW Tool</td>
</tr>
<tr>
<td>Shared library</td>
<td>68K/PowerPC</td>
<td>PowerPC Shared Library</td>
</tr>
</tbody>
</table>

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 85.

**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 91.

To learn more about running your project, refer to “Running a Project” on page 304.

**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 not executable, the Run command is not available. Non-executable projects include libraries, shared libraries, dynamic linked libraries, code resources, or tools.

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 275.

If the current project is designed to operate on another platform, you must connect your host computer to the target computer or device before choosing Run. For additional information, see the appropriate targeting manual for the target platform.
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 304.

The second step is for you to start the Metrowerks 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 304 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 304. 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.

If the current project is designed to operate on another platform, you must connect your host computer to the target computer or device before choosing Run. For additional information, see the appropriate targeting manual for the target platform.
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 (Mac OS) 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.

The setting to control this option are in the target’s Linker panel. After configuring your project, you will need to Make your project. If the compile and link is successful, a link map file named after your project with the .MAP extension is saved in your project folder.

To learn more about a target’s Linker panel, see the discussion in the appropriate targeting manual.

---

**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 78.

---

**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 com-
Compiling and Linking
Compiling and Linking a Project

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 10.2 appears.

Figure 10.2 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 85.

Remove Objects & Compact command

When the Option key is held down, Remove Object Code changes to Remove Object Code & Compact. This command removes all binaries from the project and compacts it to consume the minimum amount of space on your hard disk.
Compiling and Linking

Using Precompiled or Preprocessed Headers

The Remove Object Code & Compact command removes all binaries from the current target (or optionally all targets) of the project and compacts it to consume the minimum amount of space on your hard disk. The procedure is similar to that discussed in “Removing Object Code” on page 307, but in addition the project file is compressed.

Compacting the project removes all object code and debugging information stored in the project file and retains only the information about which files belong to the project and project-specific option settings.

Advanced Compile Options

This section describes two options that either speed up your project build times or alert you when a build is completed.

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 288.

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 288.

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.

**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:

```c
#include <stdio.h>
#include <string.h>
```

To specify the precompiled header filename in source code instead, add

```c
#pragma precompile_target “name”
```

**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. 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 321.
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 the CodeWarrior IDE encounters a .pch or .pch++ file in the project that was modified since it was last precompiled, the CodeWarrior IDE 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 309.

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 filename extension valid for your platform.

Windows Use .pch as the extension for precompiled headers.

MacOS Use .pch or .pch++ as the extension for precompiled headers.

Now add the source file to the open project with the Add Window command in the Project Menu.

Whenever the precompiled header 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 can specify a precompiled header as a prefix file, using the settings for your target. To learn about how to do this, refer to the discussion of the C/C++ Language settings panel in the C Compiler Guide.

**NOTE:** Do not use a precompiled header source file in `#include` directives; use the name of the resulting precompiled header file instead. Although using the precompiled header source 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 make it the prefix file specified in the C/C++ Language settings panel.

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++ Language 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.

If this is a C header, use the precompiled header extension .pch. For example, save this file as “MyPrecomp.pch”.

**Mac OS**  For a C++ header, use .pch++.

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++ Language settings panel.

In the Prefix File field, enter your precompiled file's name, in this example “MyPrecomp.pch”. 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 Language 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
   
   ```pascal
   {$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 Language 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.

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.

To save the contents of the new window, choose one of the save commands in the File Menu.

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."

To save the contents of the ".dump" window, choose one of the save commands in the File Menu.

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, VCS Pop-up Menu, File Path Caption and Line Number Button do, refer to “Guided Tour of the Editor Window” on page 122.

The topics in this section include:

- Error Button
- Warning Button
- Project Information Caption
- Extra Information Button
- Stepping Buttons
- Message List Pane
- Source Code Disclosure Triangle
- Source Code Pane
- Pane Resize Bar
Figure 10.3  The CodeWarrior Message Window

The Error Button in the Message window 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 320.
Warning Button

The Warning Button in the Message window 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 320.

Project Information Caption

The Project Information Caption gives a short description of the view you are looking at in the Message window. Your project name will appear here.

Extra Information Button

Expands a message to show information about the project, target, and file that caused the message.

Stepping Buttons

The Stepping Buttons 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 321.

Message List Pane

The Message List Pane displays your messages.

To learn more about seeing messages in the Message window, refer to “Seeing Errors and Warnings” on page 320.

Source Code Disclosure Triangle

The Source Code Disclosure Triangle 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 320.

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.

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. See “Guided Tour of the Message Window” on page 316 for information on the interface items in this window.

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:

<table>
<thead>
<tr>
<th>Select this …</th>
<th>To display this…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>Either compiler or linker errors. Both types of errors prevent the compiler and linker from creating a final binary.</td>
</tr>
<tr>
<td>Warnings</td>
<td>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.</td>
</tr>
<tr>
<td>Notes</td>
<td>All other types of messages issued in the Message window. For example, results of a batch find are notes messages.</td>
</tr>
</tbody>
</table>

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.

Mac OS Only Messages
• During a Find Reference when more than one definition for the same function is found (multiple definitions).
• During a Find Reference on a C++ function that has been overridden, the message window appears stating that there are two or more instances.

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.

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.

See “Guided Tour of the Message Window” on page 316 for information on the interface items in this 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 122.

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
**Compiling and Linking**

*Using the Message Window*

**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 318 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. Notice that the **Source Code Pane** view now shows the source code that corresponds to the message. You can then edit the offending line directly in the Source Code Pane view.

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 122.

**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.

---

**Figure 10.4** Statement arrow pointing to error.

```
/*
{*
  int i, j, k, m;
  for (i=0; i<BSIZE; i++) {
    for (j=0; j<BSIZE; j++) {
```
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 318. To learn about changing the view of messages in the Message window, refer to “Seeing Errors and Warnings” on page 320.

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:

- 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.

Windows Only Linker Errors

- 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 23. Linker error messages of this type occur when the project is missing a library.

Mac OS Only Linker Errors

- Your project is missing the necessary libraries. Linker error messages of this type occur when the project is missing a library. To learn how to find which libraries you should be us-
Your compiled code generates a reference too large to be handled by the selected code model.

**Finding Which Library to Use (Mac OS)**

Often a linker error occurs when a library is missing from a project. As a result, one or more functions or identifiers cannot be found and the linker reports a problem. The difficulty you face is trying to figure out which library defines the function or identifier.

To find out which libraries or shared libraries should be added to your project, you can refer to the *CodeWarrior Targeting* manual appropriate for your target, as described in Table 1.2 on page 23. In addition, you can use the Find Library tool included on the CodeWarrior Tools CD.

Find Library is a set of files with the internal names of functions and global identifiers from many of the libraries included with the CodeWarrior IDE for Mac OS and ANSI Libraries. Use these files as a file set with the CodeWarrior IDE Find dialog box. Search for the missing identifier in these files, and you can quickly find the right library to include in your project.

Here’s how to use the Find Library tool.

1. **Create a File Set**
   Use the Find window to create a set that includes all the files in the Find Library folder on your CodeWarrior Tools CD. To Learn how to create and use a file set, refer to “Choosing Files to be Searched” on page 175 and “Saving a File Set” on page 179.

2. **Set the search parameters to ignore case.**
   Be sure to select the "Ignore Case" option when doing your search with the Find window. Some identifiers may be all uppercase or lowercase.

3. **Search for the symbol that the linker can’t find.**
   Enter the name of the unlinked symbol, and do a batch search of the Find Library file set. When the search has found a matching identi-
fier, you only need to check the file name of the library to determine
the name of the library to include in your CodeWarrior project.
There is no path information given in the files, so you'll have to lo-
cate the correct library in the appropriate Metrowerks CodeWarrior
subdirectory on your hard disk.

For example, if you do a batch search for the identifier num2dec in
this file set, you'll get a hit in the MathLib file. You now know that
if you want to use the num2dec function, you must include the
MathLib in your project.

**WARNING!** The files in the Find Library folder are just compila-
tions of names taken out of the libraries and put into a file. These
are not the actual library files. Do not include the 'Find Library' files
in your projects.

**NOTE:** Some names may be duplicated in more than one file. If
the name has a leading period, this probably is a declaration in
that file, not a definition. Be sure to use command-G or batch
search to retrieve all files with matching identifiers. Then select the
appropriate library for your particular project.

Some of the 68K and PowerPC shared libraries differ slightly al-
though the library names are the same. To identify the correct li-
brary, we’ve added the suffix -68K or -PPC as appropriate. For ex-
ample, DragLib is is listed both as DragLib -68k and DragLib -PPC.
Ignore the suffix when looking for the library by name, but make
sure you include the correct CPU version of the library for your tar-
get platform.

The CodeWarrior IDE will report the lines on which hits occur, but
you can ignore those line numbers. The line information reported is
irrelevant with respect to the actual library, and refers only to the
line number in the special naming file.

There are several versions of the ANSI Libraries. They all contain
the same identifiers, so only one version is included. You should as-
Compiling and Linking

Using the Message Window

Assume that the appropriate library for your project will contain the same functionality.

The ANSI C++ Library file will be difficult to understand because of name mangling. However, most linker errors will list a function name as well as the identifier. Searching for that function name should give you good results. For example, you might get an error message saying "__aad__9bitstringFRC9bitstring in bitsand.c" is unknown. You could either search for the mangled name or for bits and to get a hit.

Correcting Pascal Circular References

The Make and Run command for the Metrowerks Pascal compiler builds your project by examining every Pascal file in your project file. As this examination is performed, a tree of dependencies is built for the interfaces of your units and for their implementations.

A circular reference occurs when a unit declares something that is used in another unit and that same unit declares something used by the former. To break this loop, the Pascal compiler does not allow such things among the interface parts of units, but it is permitted for implementations.

Listing 10.1 A valid example of circular referencing

```
UNIT A;
INTERFACE
USES C;
TYPE A_type = ...;
IMPLEMENTATION USES B;
....
```

The example in Listing 10.1 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 de-
Compiling and Linking
Using the Message Window

PENDS 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 print or save the contents of the Message window, just follow these steps.

To Print the Message Window:

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 Print command from the File Menu.
   If you choose the Print command on the File Menu to print the Message window, the print dialog box appears. 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.

**To Save the Message Window:**

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 command from the File Menu.**
   The Save A Copy As command will display a standard file dialog box.

3. **Specify the name of the file and the location.**
   A text file will be saved containing all the errors, warnings, and messages listed in the message window.

**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.

**Mac OS Special Library Options**

There are a few special linker options you can use in your project. These options are only accessible from the Project Inspector window as shown in Figure 10.5, available from the Window Menu. These options are only available for library files in the project, and will be disabled for text files in the project.
Figure 10.5   Special Library Options

The topics in this section are:

- **Import Weak**
- **Initialize Before**
- **Merge into Output**
Import Weak

Import weak tells the operating system to ignore unresolved symbols at load time. Use this option for features not always installed on all machines (such as QuickDraw GX or QuickTime). For more information on shared libraries, see the Targeting Mac OS manual.

Initialize Before

Initialize Before indicates which PEF container (shared library or application) gets initialized first. By default, imported PEF containers are initialized before the PEF containers that import them. Use this option for mutually-dependent PEF containers, to specify which gets initialized first.

Merge into Output

Merge shared libraries allows you to put a copy of a shared library into your project output file. For instance, if you have created your own custom shared library, you can merge it into the application's data fork, guaranteeing that the system finds it.

This option may be useful for creating fat libraries as well, where you want to include both CFM68K and PowerPC code.

NOTE: When you merge a shared library into your application, you are copying all the code fragments in that library. If the library contains both 68K and PowerPC code fragments, your application's data file gets them all, which is desirable for fat libraries, but in other cases may increase size unnecessarily.
This chapter explains how to use the CodeWarrior IDE version control integration facilities to control your source code.

Version Control System Overview

The CodeWarrior IDE includes features for integrating your projects with revision control systems, such as Metrowerks Visual SourceSafe for Macintosh (formerly MW CodeManager), Microsoft Visual SourceSafe, and MW SourceServer.

The sections in this chapter are:

- Version Control System (VCS) Setup
- Setting up a Project for Revision Control
- Using Source Code Control with Files

**NOTE:** Neither Metrowerks Visual SourceSafe for Macintosh nor Microsoft Visual SourceSafe is included with the Metrowerks CodeWarrior Professionel product. For more information on either of these products, please contact the respective company.
Version Control System (VCS) Setup

The VCS settings panel, as shown in Figure 11.1, is present if you have installed recognized revision control software for use with the CodeWarrior IDE. Use the VCS Setup panel to specify client options for connecting to a version control system.

**Figure 11.1** The VCS settings panel

**Windows**  
CodeWarrior is capable of connecting to Microsoft Visual SourceSafe on PC’s running Windows.

**Mac OS**  
Metrowerks Visual SourceSafe for Macintosh is a recognized version control system for the Mac OS.
NOTE: Both Microsoft Visual SourceSafe and Metrowerks Visual SourceSafe for Macintosh are available separately from their respective manufacturer. Neither product is included with CodeWarrior Professional.

Once Use Version Control is enabled, a VCS menu appears on the menubar as shown in VCS menus for Windows and Mac OS. You select commands from this VCS menu to Get, Checkin, Checkout, and Undo Checkout items from the active version control system.

Figure 11.2 VCS menus for Windows and Mac OS
Configuring Version Control Software

Setting up a Project for Revision Control

NOTE: The VCS menu differs between the Windows and Mac OS versions of SourceSafe due to the level of support offered by the VCS plug-in.

To learn more information about using a revision control system with the CodeWarrior IDE, refer to “Version Control System Overview” on page 331. For detailed information on configuration of your revision control software, refer to the documentation that came with the product.

TIP: If you use the Method Pop-up menu in this panel to select “None,” the Checkout Status column in the CodeWarrior IDE Project window will display the same states as the CodeWarrior IDE Editor’s VCS Pop-up Menu, checking only file locks and ‘ckid’ resources.

Setting up a Project for Revision Control

Setting up a CodeWarrior project for version control is simple. You just need to enter some information in the Version Control System (VCS) Setup panel while a project is open. You’ll need to input the information separately for each project that uses revision control.

NOTE: For purposes of this exercise we will use SourceSafe as the version control software. SourceSafe-compatible databases are cross-platform and available in Windows, Mac OS, and UNIX versions.

WARNING! If you change the VCS Settings while you’re connected to a SourceSafe database, the CodeWarrior IDE may disconnect you.
SourceSafe must be installed before any of this configuration will work properly.

**NOTE:** See the documentation that comes with the version of SourceSafe you are using for installation instructions.

**TIP:** Before starting this exercise, ensure that you already have access to a SourceSafe-compatible database and a user account available. See your database administrator for information on the database’s location, your username, password, etc. as you will need that information in order to access the project database.

**To Activate Your VCS Settings:**

1. **Choose Version Control Settings from the Edit menu.**
   The VCS Settings window appears.

2. **Enable the Use Version Control option.**
   Click the **Use Version Control** option to activate version control.

3. **Choose a version control system from the Method menu.**
   The CodeWarrior IDE is capable of supporting several different types of version control. For this exercise you should specify that you’re using SourceSafe.

   **NOTE:** This option is called SourceSafe since it works with any SourceSafe-compatible database, including MS Visual SourceSafe and Metrowerks Visual SourceSafe for Macintosh.

4. **Enter your username.**
   Enter your SourceSafe user name in the **Username** field.
5. **Enter your password.**

To avoid the Login dialog, turn on the **Remember Password** option and enter your password in the **Password** field.

If you don’t enter your password, the CodeWarrior IDE will display the Login dialog each time you connect to a SourceSafe database.

6. **Enable the Always Show Login Dialog option.**

If you want to see the Login dialog even though you entered your password, turn on the **Always Show Login Dialog** option. This option is useful if you usually use the same SourceSafe username, but may want to use a different one occasionally.

7. **Enable the Connect on Open option.**

If you turn on the **Connect on Open** option, the CodeWarrior IDE connects you to the SourceSafe database each time you choose a command from the VCS menu. If you leave this option off, you have to connect to the database yourself by choosing the **Connect** command from the VCS menu.

8. **Locate the SourceSafe database folder.**

Click the **Choose** button and locate the database folder from the open file dialog that appears. Your database folder is where the SourceSafe database is located, and it should be the same directory specified by the **Data_Path** variable.

The folder’s pathname appears in the Database Path field.

9. **Choose a local folder.**

Click the **Choose** button to set the local path to your working directory. The local working directory folder is where SourceSafe places files after checking them out of the specified project database and where the CodeWarrior IDE expects to find your files.

The folder’s pathname appears in the Local Path field.

---

**NOTE:** The SourceSafe options can be used with any SourceSafe-compatible database, including Metrowerks Visual SourceSafe for Mac OS.
10. Click Save.

Save your VCS Setting changes using the Save button.

To Set a Database Project

1. Click the SourceSafe panel.

Select the SourceSafe panel from the Version Control group in the VCS Settings Panel list. The SourceSafe panel appears.

2. Enter the project path to use.

Enter the path to a project in the database to use as the default project. See your database administrator for this information if you don’t already have it.

3. Click Save.

Click the Save button to preserve your project setting.

Congratulations. If all your data was correct, you should be connected to the SourceSafe-compatible database set in the Database Path field. If you aren’t, review the above steps, ensuring that the data entered is correct.

Using Source Code Control with Files

A revision control system, such as *Microsoft Visual SourceSafe* or *Metrowerks Visual SourceSafe for Macintosh*, allows you to maintain a database of your source code, and then check files in or out of the database. This makes it easy to track changes to your code, particularly when more than one person is working on your software project.

There are two primary ways to use revision control from the CodeWarrior IDE. You can use the VCS menu (Figure 11.2) that appear on the menubar, or use the VCS Pop-Up menus (Figure 11.3) that appear in all Editor windows.

**Windows** A software plugin for *Microsoft Visual SourceSafe* is included with the CodeWarrior IDE, and revision control operations can be used if you have SourceSafe (available separately) already installed on your system.
NOTE: If you are working on a Macintosh, you can learn more about the VCS menu in the MW Visual SourceSafe Plug-in Guide. It is available as part of the MW Visual SourceSafe for Macintosh product, purchased separately from CodeWarrior.

Mac OS There is a third, more obscure method available on the Macintosh. You can use ToolServer and the command-line MW Visual SourceSafe tools. To learn more about the command-line MW Visual SourceSafe tools, refer to the MW Visual SourceSafe documentation. To learn more about ToolServer, refer to “Using MPW ToolServer Overview” on page 487.

The topics in this section include:

- Determining Version Control Status of a File
- Modifying a Read Only Source Code File
- Other Revision Control Operations
- Viewing a File’s Status in the Project Window
- Mac OS ‘ckid’ Resources
- VCS Window

Determining Version Control Status of a File

The CodeWarrior Editor displays the revision control status of the file, in the VCS Pop-up Menu in the Editor window, as shown in Figure 11.3. If version control is not activated or unavailable the pop-up shown in Figure 11.4 appears.

If a file is not already added to the revision control system database, you will see a menu as shown in Figure 11.4.
Mac OS  The CodeWarrior Editor displays the revision control status of the file, which is controlled by the ‘ckid’ resource, in the VCS Pop-up Menu in the Editor window, as shown in Figure 11.3. Both Projector from Apple and MW Visual SourceSafe use the ‘ckid’ resources to mark files that are in revision control with the permissions of the file.

Figure 11.4  VCS Not Available
To understand the permissions of a file you are viewing in the Editor, look at the VCS Pop-Up Menu and compare the icon in the menu with the icons shown in Table 11.1.

**TIP:** (Mac OS) Some of the icons shown in Table 11.1 can also appear if you are using files that have a 'ckid' resource in them. Many programs can put 'ckid' resources in files to track file permissions. One such program that can do this is the Projector tool in MPW, included on your CodeWarrior CD. For more information about 'ckid' resources, refer to the Building and Managing Programs in MPW document on your CodeWarrior Reference CD.

### Table 11.1 File Permissions Icons

<table>
<thead>
<tr>
<th>If the icon is...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Checked out" /></td>
<td>You can edit the file and add your changes to the revision control database.</td>
</tr>
<tr>
<td><img src="image" alt="Modify Read-Only" /></td>
<td>You can edit the file, but you cannot add your changes to the revision control database because the file was not properly checked-out for modification.</td>
</tr>
<tr>
<td><img src="image" alt="Read-Only" /></td>
<td>You cannot edit the file, and it is part of the revision control database.</td>
</tr>
<tr>
<td><img src="image" alt="Unlocked" /></td>
<td>The file can be edited, and it is not checked into a revision control database.</td>
</tr>
<tr>
<td><img src="image" alt="Locked" /></td>
<td>You cannot edit the file, and it is not part of the revision control database. You may not have the access privileges needed to access the file, or someone may have locked the file.</td>
</tr>
</tbody>
</table>

### Modifying a Read Only Source Code File

You can change the status of a Read Only source code control file so you can modify it, but after modification, you will not be able to check it back into the source code control database. This feature is
provided to allow you to experiment with a file by temporarily making it writable. If the revision control database allowed you to override permissions on files without consequence, it would not be providing any valuable control services for you.

**Windows** Right-mouse click on the file in Explorer, select Properties from the menu that pops up, and see if the Locked attribute is checked. If you uncheck it, the file can then be edited.

**Mac OS** Click the Read Only icon in the VCS pop-up menu in the Editor window so that the menu pops up, as shown in Figure 11.5. Choose the Make Writable command from the menu. The VCS Pop-Up Menu icon changes to Modify-Read Only. You can now modify the file, but you won’t be able to check it back into its source code control database.

![Figure 11.5 Changing Permissions from Read-Only to Modify Read-Only](image)

**Other Revision Control Operations**

When using the VCS Pop-up Menu, there are some other operations you can perform from the menus.

**Windows** Refer to Figure 11.6 to see the various menus that occur depending on the revision control status of the file, and the Explorer attributes (such as locked) for the file.

**Mac OS** Refer to Figure 11.6 to see the various menus that occur depending on the state of the ‘ckid’ resource and the Finder lock for the file.

**TIP:** (Mac OS) If for some reason the CodeWarrior IDE gets confused about the VCS status of a file, hold down the Command key and click the VCS Menu will enable all VCS commands, regard-
Configuring Version Control Software
Using Source Code Control with Files

less of a file’s status. This might be required if you’ve added a file to the project that already contained a ‘ckid’ resource.

Figure 11.6 Other Permissions Pop-Up Menu Commands

Depending on the state of the VCS Pop-Up Menu when you click on it, certain operations may be performed on the file you are editing.

- **Unlock**—change the Finder lock on the file (if possible) so that the file is writable
- **Add**—add the file to the revision control database
- **Make Writable** (Mac OS)—make the file writable for experimentation, though it will not be able to be checked into the revision control database
- **Get**—retrieve a fresh copy of the file from the revision control database
- **Checkout**—check the file out from the revision control database for modifications
- **Undo Checkout**—discard any changes made to the file, and tell the revision control database to cancel the checkout of the file.
- **Checkin**—tell the revision control database to accept the file with the changes that have been made to it.

For more information on these operations, refer to the documentation that came with your revision control system.

**Windows** Note that some items may have an ellipsis character “...” after them. This means that advanced features for this opera-
tion are available. To learn more about these operations, refer to “Advanced VCS Operations” on page 344.

**Viewing a File’s Status in the Project Window**

If you have configured your project file to use a revision control system, and have checked in your files in accordance with the procedures described in its documentation, you will see one of the icons of Table 11.1 in the Checkout Status Column of the Project window for every file under revision control in your project. Refer to Figure 11.7 to see where the Checkout Status Column is located in the Project window.

Clicking the Checkout Status Column icon at the top of the column will perform a Synchronize Status command on the project’s files against the revision control database. The status of all files you have on your hard disk will be compared and synchronized with the status of the files in the revision control database.

**Mac OS** For more information about the Synchronize Status command, refer to the *MW Visual SourceSafe for Macintosh* documentation.

**Mac OS ‘ckid’ Resources**

The CodeWarrior IDE *almost* always respects the presence of ‘ckid’ resources in text files. The ‘ckid’ resources are used by Projector and MW Visual SourceSafe to track whether a file is writable or not, and are stored within the resource fork of your source code files.

The one time that the CodeWarrior IDE may do something different than indicated by a ‘ckid’ resource is when dealing with a project file that contains a ‘ckid’ resource.

To learn more about how to view the checkout status of a Project window, refer to “Project Checkout Status icon” on page 46.
Advanced VCS Operations

To access more advanced VCS features, choose the optional form of the menu command. You'll notice that some of the menu commands have an ellipsis (...) next to them, indicating that more options are available when you select the command.

- Windows VCS Features
- Mac OS VCS Features
NOTE: For purposes of these instructions, we will use the MS Visual SourceSafe as the source-control program. Other source-control programs may have their own unique windows. See their documentation for details.

Windows VCS Features

The features present in your source-control system are entirely dependent upon that particular product.

Windows VCS Commands

Choose commands from the VCS menu that contain the ellipsis (…) character after them, as shown in Figure 11.8.

![VCS Commands (Windows)](image)

Windows Supported Commands

The commands shown in Table 11.2 are fully supported from within the CodeWarrior IDE.
Table 11.2 Supported Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronize Status</td>
<td>Updates the VCS Status column in the Project window by examining each file’s status and updating its display information.</td>
</tr>
<tr>
<td>Project</td>
<td>Contains a submenu of VCS commands that enable you to Get, Checkout, Undo Checkout, Checkin, Status, and Add entire CodeWarrior project files.</td>
</tr>
<tr>
<td>Get</td>
<td>Retrieves a copy of the file without checking it out of the project database.</td>
</tr>
<tr>
<td>Checkout</td>
<td>Checkout files for exclusive modification.</td>
</tr>
<tr>
<td>Undo Checkout</td>
<td>Cancels a checkout, voiding all changes.</td>
</tr>
<tr>
<td>Checkin</td>
<td>Returns a modified file to a project and relinquishes exclusive control.</td>
</tr>
<tr>
<td>Status</td>
<td>Displays a file’s checkout status.</td>
</tr>
<tr>
<td>Add</td>
<td>Adds a file to the current project.</td>
</tr>
<tr>
<td>Connect or Disconnect</td>
<td>Connects or disconnects you from the project database depending upon the current open status of the project database.</td>
</tr>
<tr>
<td>About</td>
<td>Displays copyright and version information.</td>
</tr>
</tbody>
</table>

**Connect**

When the Connect command is selected and Always Show Logon is activated, the window shown in Figure 11.9 appears.
Figure 11.9  VCS Connect window

About

To determine which version of the CodeWarrior VCS you are running, choose About from the VCS menu to view the dialog shown in Figure 11.10.

Figure 11.10  VCS About window

Mac OS VCS Features

Mac OS  To view ellipsis characters on a command menu on the Macintosh, press the Option key while clicking on the VCS Pop-up Menu, as shown in Figure 11.11.
Figure 11.11  Advanced VCS Features (Mac OS)

To learn about each of the operations in these dialogs, refer to the *MW Visual SourceSafe for Macintosh* documentation.

- Checkin Options—shown in Figure 11.12
- Checkout Options—shown in Figure 11.10
- Get Options—shown in Figure 11.9

Figure 11.12  Advanced VCS Checkin Options
VCS Window

The CodeWarrior IDE now shows a variant of the Message Window for revision control messages, as shown in Figure 11.14. To learn how to use the controls in the window, refer to “Using the Message Window” on page 319.

Figure 11.14 VCS Window
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**
- **Debug Menu**
- **Data Menu**
- **Version Control System (VCS) Menu**
- **Window Menu**
- **Help Menu**
- **Toolbar Submenu**
- **Apple Menu (Mac OS)**
- **Tools Menu (Mac OS)**
- **Scripts Menu (Mac OS)**
- **Editor Extensions Menu (Mac OS)**

**Windows** Under Windows, the **File Menu**, **Edit Menu**, **Search Menu**, **Project Menu**, **Window Menu**, and **Help Menu** are visible at all times.

**Mac OS** Under the Mac OS, the **Apple Menu (Mac OS)**, **File Menu**, **Edit Menu**, **Search Menu**, **Project Menu**, **Tools Menu (Mac OS)**, **Window Menu**, and **Help Menu** are visible at all times.
The **Version Control System (VCS) Menu** appears only if you have installed and configured your CodeWarrior product to work with a compatible revision control system that you purchased separately. To learn more about revision control systems, and how to use them with CodeWarrior, refer to the documentation that came with that additional software.

The **Debug Menu** only appears when debugging has been enabled for a project, and the **Data Menu** appears when a window that can use its commands is present on the screen.

**Mac OS Only Menus**

The **Tools Menu (Mac OS)** appears if ToolServer is active. The **Scripts Menu (Mac OS)** and **Editor Extensions Menu (Mac OS)** only appear when you’ve set the correct preferences in the Preferences dialog box. Refer to “IDE Extras Panel” on page 246 and for more information on how to do this.

In addition, you can make the commands that appear in many menus change by holding down the Option or Shift key when clicking on the menu.

The text of some menu commands may change depending on the context of the action, or actions, performed.

Many menu commands can also have button equivalents on the toolbars. To learn more about how to customize the toolbars, refer to “Customizing Toolbars” on page 262.

**File Menu**

The **File Menu** contains commands you use when opening, creating, saving, closing, and printing existing or new source code files and projects. The **File Menu** also provides a few different methods of saving edited files.

**New**

Creates a new editable text file.
To learn more about this command, refer to “Creating a New File” on page 95 for more information.

**New Project**

Creates a new project file.

To learn more about this command, refer to “Working with Project Stationery” on page 52.

**New Empty Project**

Creates a new project file without using stationery.

**Windows** This command is available by pressing the key binding Ctrl+Alt+N.

**Mac OS** This command is available when you press the Option key before clicking on the File Menu.

To learn more about this command, refer to “Working with Project Stationery” on page 52.

**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 96.

**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 96.

**Find and Open File**

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.
See “Opening Files with the File Menu” on page 96 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 108 for more information.

To learn how to close all open Editor windows, refer to “Closing All Files” on page 109.

**Close All**

Closes all open Editor windows.

**Windows**  This command is available by pressing the key binding Ctrl+Shift+W.

**Mac OS**  This command appears in the menu when you press the Option key when selecting the menu.

To learn more about this topic, refer to “Closing All Files” on page 109.

**Switch to MW Debugger (Mac OS)**

Gives control to the Metrowerks external 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 User Guide*.

**Save**

Saves the contents of the active Editor window to disk.

For more information on this topic, refer to “Saving one file” on page 103.
Save All

Saves all Editor files that are currently open.

**Windows**  This command is available by pressing the key binding Ctrl+Shift+S.

**Mac OS**  This command appear in the menu when you press the Option key when selecting the menu.

For more information on this topic, refer to “Saving all files” on page 104.

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 104.

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 106.

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 112.
Page Setup (Mac OS)

Print Setup (Windows)

Sets the options used when printing files from the CodeWarrior IDE.

For more information about this command, see “Setting Print Options” on page 110.

Print

Prints files from the CodeWarrior IDE on your printer.

For more information on printing files, see “Printing a Window” on page 111, or read the documentation that came with your printer.

Quit (Mac OS)

Exit (Windows)

Quits the CodeWarrior IDE 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 Quit command saves all changes to the project file before the environment quits. If an Editor window is open and changes have not been saved, the CodeWarrior IDE asks if you want to save the changes before quitting.

Edit Menu

The Edit menu contains all the customary editing commands and some CodeWarrior additions, including the commands that open the Preferences and Project Settings dialogs.
Undo

The text of this menu command varies depending on the most recent action, and your Editor options settings.

**Undo** reverses the effect of your last action. The name of the **Undo** command varies depending on the type of operation you last 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 140, and “Undoing and redoing multiple edits” on page 140.

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 234.

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 140, and “Undoing and redoing multiple edits” on page 140.

To learn about how to configure multiple undo, refer to “Editor Settings” on page 234.

Cut

Deletes the selected text and puts it in the Clipboard, replacing the contents of the Clipboard.
Copy

Copies the selected text in the active Editor window onto the system Clipboard. If the Message Window is active, the Copy command copies all the text in the Message Window onto the Clipboard.

Paste

Pastes the contents of the Clipboard into the active Editor window.

The Paste command replaces the selected text with the contents of the Clipboard. If no text is selected, the Clipboard contents are placed after the text insertion point.

If the active window is the Message Window, the Paste command is dimmed and cannot be executed.

Clear

Deletes the selected text without placing it in the Clipboard. The Clear command is equivalent to pressing the Delete or Backspace key.

Select All

Selects all the text in the active window. This command is usually used in conjunction with other Edit menu commands such as Cut, Copy, and Clear.

To learn more about selecting text, refer to “Selecting Text” on page 136.

Copy to Expression

Copies the variable selected in the active pane to the expression window. You can also drag an expression to the expression window from source code or from another window or pane.

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 139.

**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 140.

**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 140.

**Insert Reference Template (Mac OS)**

Inserts a routine template corresponding to the selected Mac OS Toolbox call in the active window. The CodeWarrior IDE uses the on-line reference database application selected in the Find Definition Using pop-up menu to search for the routine’s definition.

To learn about configuring the on-line reference database application, refer to “IDE Extras Panel” on page 246.

**Show Breakpoints and Hide Breakpoints**

Displays or hides the breakpoint window. This command toggles between Show Breakpoints and Hide Breakpoints, depending on whether the breakpoint window is currently visible on the screen.

**Clear All Breakpoints**

Clears all breakpoints in all source-code files belonging to the target program.
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 226.

Target Settings

Use this command to display the Settings dialog box where you can 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 the Settings dialog box, refer to “Choosing Target Settings” on page 275. To learn how to change the current target, refer to “Set Current Target” on page 372.

Version Control Settings

This menu command displays the Version Control System options panel. To learn more about this options panel, refer to the section of this manual entitled “Version Control System Overview” on page 331.

If this command is not enabled for you to choose, you do not have a revision control system configured to use with the CodeWarrior IDE.

Search Menu

The Search Menu contains all the necessary commands used to find text, replace text, and compare files. There are also some commands for code navigation.

Mac OS Use the commands in this menu to find the definitions of routines in your source code and in a library reference database like THINK Reference™ or Apple Macintosh Programmer’s Toolbox Assistant™.
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 153.

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 167.

Find Previous

Find Previous operates the same way as Find Next, except that it finds the previous occurrence of the Find text box string.

Mac OS  Hold down the Shift key to change the Find Next command to Find Previous.

To learn more about this feature, refer to “Finding Search Text” on page 167.

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 159, this command is dimmed.

To learn more about this feature, refer to “Searching and Replacing Text in Multiple Files” on page 173.
Find in Previous File (Mac OS)

If you hold down the Shift key, the Find in Next File command is changed to Find in Previous File. This command operates in much the same way as Find In Next File. Beginning at the end of the previous file in the file list, it searches for the next occurrence of the Find text box string.

To learn more about this feature, refer to “Searching and Replacing Text in Multiple Files” on page 173.

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 136.

Enter ‘Replace’ String

This command copies the selected text in the active window into the Replace text box, making it the replacement string. This is an alternative to selecting the string to be found and copying it into the Find window.

Mac OS Hold the Shift key down in the Search Menu to change Enter ‘Find’ String to Enter ‘Replace’ String.

To learn more about replacing text, refer to “Replacing Found Text” on page 170.

Find Selection

Finds the next occurrence of the selected text in the active text editor window.

To learn more about this feature, refer to “Finding Search Text” on page 167.
Find Previous Selection (Mac OS)

If you hold down the Shift key, the Find Selection command becomes Find Previous Selection.

Find Previous Selection finds the previous occurrence of the selected text in the active text editor window.

To learn more about this feature, refer to “Finding Search Text” on page 167.

To learn how to select text, refer to “Selecting Text” on page 136.

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 170.

To learn how to select text, refer to “Selecting Text” on page 136.

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 170.
To learn how to select text, refer to “Selecting Text” on page 136.

Replace & Find Previous (Mac OS)

Press the Shift key to change Replace & Find Next to Replace & Find Previous. This command operates the same way as Replace & Find Next except that it performs a Find Previous.

Replace All

Finds all the occurrences of the Find string and replaces them with the Replace string. If no text is selected in the active editor window and there is no text in the Find string field in the Find dialog box, this command is dimmed.

Find Definition

This command searches for the definition of the routine name selected in the active window. Searching occurs in the source files belonging to the open project. If the definition is found, the CodeWarrior IDE opens the source code file where the routine is defined. The routine name is highlighted.

If the CodeWarrior IDE finds more than one definition, a Message window appears warning you of multiple definitions. For more information on the Message window, consult “Using the Message Window” on page 319.

If no definition is found, a system beep sounds.

Find Reference

This command searches for the definition of the routine name selected in the active editor window. Searching starts within the source code files belonging to the open project.

Windows If the routine definition is not found within the project files, searching continues using WinHelp files.

Mac OS If the routine definition is not found within the project files, searching continues in QuickView™, the THINK Reference™ version 2.x or Toolbox Assistant™.
If no definition is found, a system beep sounds.

To learn more information about on-line databases, refer to “Online References” on page 147.

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 215.

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 215.

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 146.

Compare Files

Opens a dialog box to choose two files or folders to compare and merge. After choosing files to compare, a file comparison window appears, showing differences between the two files. If two folders are compared, the differences between the folders are shown in the Compare Folders window. For more information, see “Comparing and Merging Files & Folders” on page 113.

Apply Difference

Adds, removes, or changes text in the destination file shown in a file comparison window that is different from the text in the comparison window’s source file.
Unapply Difference

Reverses the action of an Apply Difference command in a file comparison window.

Project Menu

The Project menu lets you add and remove files and libraries from your project. It also lets you compile, build, and link your project. All of these commands are covered in this section.

Add Window

This command adds the file in the active Editor window to the open project.

To learn more about this feature, refer to “Using the Add Window command” on page 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 68.

Create New Group

The Create New Group command allows you to create a new group in the current project. This command is present 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

The Create New Target command allows you to create a new target for the current project. This command is present in the Project menu if the Targets category is selected in the current project window.
For more information about creating targets, refer to “Working with Complex Projects” on page 79.

**Create New Segment (Mac OS)**

The Create New Segment command allows you to create a new segment (also referred to as a group of files) in the current Mac OS 68K 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 63.

**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 63.

**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 Esc/Command+Period.

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 321.
**IDE Menu Reference**

*Project Menu*

**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 315.

**Precompile**

This command precompiles the source code file in the active Editor window into a precompiled header file.

To learn more about this topic, refer to “Using Precompiled or Preprocessed Headers” on page 308.

**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 299.

**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 315.

**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 302.

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 303.

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 306.

Remove Object Code & Compact

This command removes all binaries from the project and compacts it. Compacting the project removes all binary and debugging information and retains only the information regarding which files belong to the project and project settings.

Windows  This command is available by pressing the key binding Ctrl+Shift+-.

Mac OS  Hold down the Option key to change the Remove Object Code command to Remove Object Code & Compact

To learn more about this topic, refer to “Removing Objects” on page 306.

Re-search For Files

To speed up builds and other project operations the IDE caches the locations of project files after it has found them in the access paths. Re-search for Files forces the IDE to forget the cached locations of files and re-search for them in the access paths. This command is
useful if you have moved files around on disk and want the IDE to find them in their new locations.

If the `Save Project Entries Using Relative Paths` setting is enabled the IDE does not reset the relative path information stored with each project entry, so re-searching for files will find the source files in the same location (the exception is if the file no longer exists in the old location). In this case the IDE will only re-search for header files. To force the IDE to also re-search for source files, you must first select `Reset Project Entry Paths`.

If the `Save Project Entries Using Relative Paths` is disabled, the IDE will re-search for both header and source files.

**Reset Project Entry Paths**

This command resets the location information stored with each project entry when the `Save Project Entries Using Relative Paths` setting is enabled. The next time the project entries are accessed, the IDE will re-search for the project entries in the access paths. If the `Save Project Entries Using Relative Paths` setting is disabled this command does nothing.

**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 78.

**Enable Debugger**

**Disable Debugger**

Use these commands to changes settings to allow your project to be debugged or not. The Enable Debugger command sets preferences to allow your project to be debugged. The Disable Debugger sets preferences so no debugging can occur.
Mac OS  With this command, the Run command changes to Debug and lets the debugger launch and debug your project. When it is not checked, the Run command runs your project normally.

To learn more about this topic, refer to “Controlling Debugging in a Project” on page 91.

Resume (Mac OS)

This command appears in the Project menu when the application created by the CodeWarrior IDE is running already. You choose this command to switch into the application from the CodeWarrior IDE.

Run

This command compiles, links, creates a stand-alone application, and launches that application.

Windows  If the project type is set as a library or a shared library, then the Run command is dimmed.

Mac OS  If the project type is set as a code resource, library, MPW Tool, or a shared library, then the Run command is dimmed. Press the Option key to change this command to Debug.

To learn more about this topic, refer to “Running a Project” on page 304.

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.

Mac OS  This command appears in the Project menu when you press the Option key to change the Run command to Debug, or when the debugger is enabled.

To learn more about the CodeWarrior Debugger, refer to the CodeWarrior Debugger User Guide.
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 63.

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.

Debug Menu

The Debug menu contains commands that allow you to manage program execution. All of these commands are available in both the internal and external Metrowerks debuggers. This menu only appears when Enabled Debugging is selected in the Project menu. It disappears when Disable Debugging is chosen.

For more information see the Debugger User Guide.

Stop

Temporarily suspends execution of the target program and returns control to the internal Metrowerks debugger.

Kill

Permanently terminates execution of the target program and returns control to the internal Metrowerks debugger.

Step Over

Executes a single statement, stepping over function calls.

Step Into

Executes a single statement, stepping into function calls.
Step Out

Executes the remainder of the current function until it exits to its caller.

Break on C++ Exception

Causes the debugger to break at __throw() every time a C++ exception occurs.

Break on Java Exception

Causes the debugger to break every time a Java exception occurs.

View Memory

Displays the contents of memory as a hexadecimal/ASCII character dump.

View Memory As...

Displays the memory a selected variable occupies or a selected register points to.

Switch to Monitor (Mac OS)

Gives control to the Macintosh ROM Monitor program or any low-level debugger (such as MacsBug) that you may have installed on your computer.

For information on this command, see the Debugger User Guide.

Data Menu

The Data menu lets you control how data values are displayed in the debugger. All of these commands are available in both the internal and external Metrowerks debuggers. This menu is present anytime a window that can employ the Data commands is frontmost on the screen. Otherwise it is hidden from view.

For more information see the Debugger User Guide.
IDE Menu Reference
Data Menu

Show Types

Shows the data types of all local and global variables displayed in the active variable pane or variable window.

Expand

Displays the C members, C++ data members, Pascal fields, or Java fields inside a selected structured variable, or dereferences a selected pointer or handle.

Collapse All

Hides all C members, C++ data members, Pascal fields, Java fields, or pointer or handle dereferences.

New Expression

Creates a new entry in the expression window, prompting you to enter a new expression.

Open Variable Window

Creates a separate window to display a selected variable.

Open Array Window

Creates a separate window to display a selected array.

View As...

Displays a selected variable as a value of a specified data type.

Set Watchpoint

Sets or clears a watchpoint for the selected variable or range of memory.

Clear Current Watchpoint

Clears the watchpoint your program has just hit and stopped at.
**Default**
Displays the selected variable in its default format based on the variable type.

**Signed Decimal**
Displays the selected variable as a signed decimal value.

**Unsigned Decimal**
Displays the selected variable as an unsigned decimal value.

**Hexadecimal**
Displays the selected variable as a hexadecimal value.

**Character**
Displays the selected variable as a character value.

**C String**
Displays the selected variable as a C character string.

**Pascal String**
Displays the selected variable as a Pascal character string.

**Floating Point**
Displays the selected variable as a floating-point value.

**Enumeration**
Displays the selected variable as an enumeration.

**Fixed**
Displays the selected variable as a numerical value of type `Fixed`. 
Fract

Displays the selected variable as a numerical value of type Fract.

Window Menu

The Window menu includes commands that tile open editor windows, switch between windows, and reopen previously opened projects. There is also a submenu for customizing the toolbars.

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 disabled when the active window is the Project window or Message window.

Mac OS ToolServer Worksheets also disable this command.

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 Editor Window Settings” on page 132, or “Saving a Default Browser” on page 221.

Toolbar

This menu item causes the Toolbar submenu to appear. To learn more about this submenu, refer to “Toolbar Submenu” on page 383.

Browser 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 196. To learn how to activate the Browser, refer to “Activating the Browser” on page 190.

Class 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 206.

To learn how to activate the Browser, refer to “Activating the Browser” on page 190.

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 197.

To learn how to activate the Browser, refer to “Activating the Browser” on page 190.

**Build Progress Window**

This menu command brings the progress window for builds, as shown in Figure 10.1 on page 301, 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 316. Also, refer to “Using Batch Searches” on page 172.

**Project Inspector**

This menu command allows you to view information about your project, and also enable debug information generation.

To learn more about this command’s window, refer to “Guided Tour of the Project Window” on page 40.

**ToolServer Worksheet (Mac OS)**

Brings the ToolServer Worksheet window to the front. This window is used in conjunction with ToolServer. This command is disabled provided one of the following conditions have been met:

- ToolServer is not installed on your machine.
- ToolServer is installed on your machine, but it has not been started. To start ToolServer, select the Start ToolServer command in the Tools menu.

To learn more about ToolServer, refer to “Using MPW ToolServer Overview” on page 487.
Processes Window
Displays the Process window. For information on this command, see the Debugger User Guide, “Show/Hide Processes” on page 146.

Expressions Window
Displays the Expression window. For information on this command, see the Debugger User Guide, “Show/Hide Expressions” on page 147.

Global Variables Window
Displays the Global Variables window. Within this window you can view the global variables for the entire project or those contained in a file. Click on a file name in the Files list to display its global variables in the Variables list.

Breakpoints Window
Displays the Breakpoint window. For information on this command, see the Debugger User Guide, “Show/Hide Breakpoints” on page 147.

Watchpoints Window
Displays the Watchpoint window. For information on this command, see the Debugger User Guide, “Show/Hide Watchpoints” on page 147.

Registers Window
Displays the Registers window. For information on this command, see the Debugger User Guide, “Show/Hide Registers” on page 147.

FPU Registers Window
Displays the FPU Registers window. For information on this command, see the Debugger User Guide, “Show/Hide FPU Registers” on page 147.
Close All Variable Windows

Closes all open variable and array windows. For information on this command, see the Debugger User Guide, “Close All Variable Windows” on page 147.

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 windows are shown in this menu and the first nine files (1 through 9) are given key equivalents. The current project is always assigned the number 0 (zero). Press Ctrl/Command and a number to open a specific Editor window. A checkmark is placed beside the active window. A file whose modifications have not been saved is underlined.

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.

Version Control System (VCS) Menu

The Version Control System (VCS) Menu, shown in Figure 12.1, may appear in the menu bar of your CodeWarrior IDE if you have purchased the MW Visual SourceSafe for Macintosh Version Control System (available separately) for use with the CodeWarrior IDE.
To learn how to configure MW Visual SourceSafe to use revision control with your projects, refer to the documentation with your MW Visual SourceSafe product.

Refer to Figure 12.1 to see what the VCS Menu looks like.
Help Menu

On-line help is available from the Help menu. When you are working in the CodeWarrior IDE, select one of the items to get interactive, on-line help.

Windows Help Menu

Contents (Windows)

This menu command displays the CodeWarrior help files.

Keys (Windows)

This menu command displays the topic index for all CodeWarrior on-line help files.

CodeWarrior IDE (Windows)

This menu command displays the topic index for the CodeWarrior IDE on-line help file.

Debugger (Windows)

This menu command displays the topic index for the CodeWarrior Debugger on-line help file.

How to Use Help (Windows)

This menu command displays the on-line help that tells how to use the help facilities.

About Metrowerks (Windows)

Displays the Metrowerks About Box.

Mac OS Help Menu

The Help menu under Mac OS 8 or newer contains a list of Apple Guides that can assist you in understanding and using the

**Toolbar Submenu**

The Window Menu has another submenu under it for the Toolbar command. 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 “Customizing Toolbars” on page 262.

**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.

**Anchor Global Toolbar (Mac OS)**

This menu command causes the IDE Toolbar to be moved so that it is on the left edge of the screen, immediately below the menu bar.

**Apple Menu (Mac OS)**

The Apple menu contains one IDE-related item.

**About Metrowerks...**

Choose this item to see the way-cool About Box.

**Tools Menu (Mac OS)**

The Tools menu contains commands used to do things like start and stop ToolServer. It also contains commands that execute Macintosh
Programmer’s Workbench (MPW) tools and scripts, and reference the MPW 411 database.

To learn more about how to use the Metrowerks command-line tools, refer to *CodeWarrior Command Line Tools* manual for more information.

To learn more about ToolServer, refer to “Using MPW ToolServer Overview” on page 487.

**Start ToolServer**

Initiates ToolServer handshaking. If you do not have ToolServer installed on your computer, this command is dimmed.

**Stop ToolServer**

When ToolServer is launched, the Stop ToolServer command replaces the Start ToolServer command. Stop ToolServer quits ToolServer and removes the extra ToolServer menu from the CodeWarrior IDE’s menu bar.

**Commando**

This command brings up the Commando dialog box, as shown in Figure 12.2. To learn more about Commando, refer to “Using MPW ToolServer Overview” on page 487.
Figure 12.2 Commando Dialog Box

Execute as ToolServer Script

This option could not be documented at press time. Check the release notes for the latest information.

Lookup Symbol

This option could not be documented at press time. Check the release notes for the latest information.

Insert Template

This option could not be documented at press time. Check the release notes for the latest information.

ToolServer Tools

You can use this submenu to run an MPW tool or script that is installed in the Tools folder of your ToolServer application. To learn more about ToolServer, refer to “Using MPW ToolServer Overview” on page 487.
Scripts Menu (Mac OS)

The Script menu, shown in Figure 12.3, contains a list of any AppleScripts in the (Scripts) folder.

This menu will only appear if the appropriate setting is turned on in the appropriate preference panel. Also, this menu will only be shown if the (Scripts) folder is present in your Metrowerks CodeWarrior folder. To learn more about how to configure the appropriate preferences, refer to “IDE Extras Panel” on page 246.

To learn more information about AppleScripts and scripting the CodeWarrior IDE, refer to “CodeWarrior Apple Events Overview” on page 405.

To learn about what the individual scripts do, open them with a text editor or an AppleScript editor application.

Note that the menu is hierarchical, based on the directory structure of the (Scripts) folder. You may place scripts in the (Scripts) folder inside other directories to create a hierarchical menu.
Figure 12.3  The Script menu

Open Scripts Folder

Brings the Finder to the front and opens the (Scripts) folder. This command is always available if the Script menu is active.

Other Script menu items

The other items in this menu are the names of any AppleScripts in the (Scripts) folder. If you do not have any AppleScripts, no items are listed.
Editor Extensions Menu (Mac OS)

The Editor Extensions menu (Figure 12.4) is similar to the Script Menu. If the Use Editor Extensions preference is enabled, a list of BBEdit™ extensions are added to the menu bar. You must have an alias or folder called (Editor Extensions), including the parentheses, in the same folder as the CodeWarrior IDE application.

To learn more about how to configure the appropriate preferences so this menu becomes visible, refer to “IDE Extras Panel” on page 246.

For more information about BBEdit, refer to the documentation that came with the product.

Figure 12.4 Editor Extensions menu
Default CodeWarrior Key Bindings

This chapter describes the default key bindings assigned to commands in the CodeWarrior IDE.

Some commands do not have any key bindings assigned to them by Metrowerks, so their respective cells appear blank. You can assign key bindings to any blank command. For more information on key bindings, see “Key Bindings Panel” on page 250.

The key bindings sections include:

- File Menu
- Edit Menu
- Search Menu
- Project Menu
- Debug
- Data
- Window Menu
- Misc.
- Editor Commands
- Prefix Keys

Mac OS Key Binding Legend

On the Mac OS, the special keys appear as symbols. The legend in Table 12.1 describe the meaning of each character symbol.
Table 12.1  Mac OS key binding legend

<table>
<thead>
<tr>
<th>For character…</th>
<th>Press this key…</th>
</tr>
</thead>
<tbody>
<tr>
<td>⌘</td>
<td>Command key</td>
</tr>
<tr>
<td>⌘</td>
<td>Option key</td>
</tr>
<tr>
<td>⌘</td>
<td>Shift key</td>
</tr>
<tr>
<td>⌘</td>
<td>Control key</td>
</tr>
</tbody>
</table>

File Menu

The key bindings in Table 12.2 contains the key bindings for manipulating projects and files from within the CodeWarrior IDE.

Table 12.2  File key bindings

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Ctrl-N</td>
<td>Command-N</td>
</tr>
<tr>
<td>New Project</td>
<td>Ctrl-Shift-N</td>
<td>Shift-Command-N</td>
</tr>
<tr>
<td>New Empty Project</td>
<td>Ctrl-Alt-N</td>
<td>Option-Shift-Command-N</td>
</tr>
<tr>
<td>Open</td>
<td>Ctrl-O</td>
<td>Command-O</td>
</tr>
<tr>
<td>Open Recent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Find and Open File</td>
<td>Ctrl-D</td>
<td>Option-Command-D</td>
</tr>
<tr>
<td>Close</td>
<td>Ctrl-W</td>
<td>Command-W</td>
</tr>
<tr>
<td>Close All</td>
<td>Ctrl-Shift-W</td>
<td>Option-Command-W</td>
</tr>
</tbody>
</table>
## Default CodeWarrior Key Bindings

### Edit Menu

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch to MW Debugger</td>
<td></td>
<td>Command-J</td>
</tr>
<tr>
<td>Save</td>
<td>Ctrl-S</td>
<td>Command-S</td>
</tr>
<tr>
<td>Save All</td>
<td>Ctrl-Shift-S</td>
<td>Option-Command-S</td>
</tr>
<tr>
<td>Save As</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save a Copy As</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revert</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page Setup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print</td>
<td>Ctrl-P</td>
<td>Command-P</td>
</tr>
<tr>
<td>Exit/Quit</td>
<td></td>
<td>Command-Q</td>
</tr>
</tbody>
</table>

### Edit Menu

The key bindings in Table 12.3 contain the key bindings for ...

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>Ctrl-Z</td>
<td>Command-Z</td>
</tr>
<tr>
<td>Redo</td>
<td>Ctrl-Shift-Z</td>
<td>Shift-Command-Z</td>
</tr>
<tr>
<td>Cut</td>
<td>Ctrl-X</td>
<td>Command-X</td>
</tr>
<tr>
<td>Copy</td>
<td>Ctrl-C</td>
<td>Command-C</td>
</tr>
<tr>
<td>Paste</td>
<td>Ctrl-V</td>
<td>Command-V</td>
</tr>
<tr>
<td>Clear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select All</td>
<td>Ctrl-A</td>
<td>Command-A</td>
</tr>
<tr>
<td>Balance</td>
<td>Ctrl-B</td>
<td>Command-B</td>
</tr>
</tbody>
</table>
Default CodeWarrior Key Bindings

Search Menu

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift Left</td>
<td>Ctrl-[</td>
<td>Command-[</td>
</tr>
<tr>
<td>Shift Right</td>
<td>Ctrl-]</td>
<td>Command-]</td>
</tr>
<tr>
<td>Insert Reference</td>
<td></td>
<td>Command-Y</td>
</tr>
<tr>
<td>Template</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preferences</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target Settings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCS Settings</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Search Menu

The key bindings in Table 12.4 contains the key bindings for ...

**Table 12.4** Search key bindings

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td>Ctrl-F</td>
<td>Command-F</td>
</tr>
<tr>
<td>Find Next</td>
<td>F3</td>
<td>Command-G</td>
</tr>
<tr>
<td>Find Previous</td>
<td>Shift-F3</td>
<td>Shift-Command-G</td>
</tr>
<tr>
<td>Find in Next File</td>
<td>Ctrl-T</td>
<td>Command-T</td>
</tr>
<tr>
<td>Find in Previous File</td>
<td>Ctrl-Shift-T</td>
<td>Shift-Command-T</td>
</tr>
<tr>
<td>Enter Find String</td>
<td>Ctrl-Shift-F3</td>
<td>Command-E</td>
</tr>
<tr>
<td>Enter Replace String</td>
<td>Ctrl-Shift-E</td>
<td>Shift-Command-E</td>
</tr>
<tr>
<td>Find Selection</td>
<td>Ctrl-H</td>
<td>Command-H</td>
</tr>
<tr>
<td>Find Previous Selection</td>
<td>Ctrl-Shift-H</td>
<td>Shift-Command-H</td>
</tr>
</tbody>
</table>
## Default CodeWarrior Key Bindings

### Project Menu

The key bindings in Table 12.5 contains the key bindings for managing projects, controlling compilations, and much more.

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace</td>
<td>Ctrl-=</td>
<td>Command-=</td>
</tr>
<tr>
<td>Replace &amp; Find Next</td>
<td>Ctrl-L</td>
<td>Command-L</td>
</tr>
<tr>
<td>Replace &amp; Find Previous</td>
<td>Ctrl-Shift-L</td>
<td>Shift-Command-L</td>
</tr>
<tr>
<td>Replace All</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Find Definition</td>
<td></td>
<td>Option-Command-`</td>
</tr>
<tr>
<td>Find Reference</td>
<td></td>
<td>Shift-Command-`</td>
</tr>
<tr>
<td>Go Back</td>
<td>Ctrl-Shift-B</td>
<td>Shift-Command-B</td>
</tr>
<tr>
<td>Go Forward</td>
<td>Ctrl-Shift-F</td>
<td>Shift-Command-F</td>
</tr>
<tr>
<td>Goto Line</td>
<td>Ctrl-G</td>
<td>Command-,</td>
</tr>
<tr>
<td>Compare Files</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apply Difference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unapply Difference</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 12.5  Project key bindings

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Files</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Default CodeWarrior Key Bindings

### Project Menu

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Group/Segment/Target</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove Selected Items</td>
<td>Ctrl-Del</td>
<td>Command-del</td>
</tr>
<tr>
<td>Check Syntax</td>
<td>Ctrl-;</td>
<td>Command-;</td>
</tr>
<tr>
<td>Preprocess</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precompile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile</td>
<td>Ctrl-F7</td>
<td>Command-K</td>
</tr>
<tr>
<td>Dissassemble</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bring Up To Date</td>
<td>Ctrl-U</td>
<td>Command-U</td>
</tr>
<tr>
<td>Make</td>
<td>F7</td>
<td>Command-M</td>
</tr>
<tr>
<td>Remove Object Code</td>
<td>Ctrl---</td>
<td>Command---</td>
</tr>
<tr>
<td>Remove Object Code &amp; Compact</td>
<td>Ctrl-Shift---</td>
<td>Option-Command---</td>
</tr>
<tr>
<td>Re-search For Files</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset Project Entry Paths</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synchronize Modification Dates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable Debugging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disable Debugging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>F5</td>
<td>Command-R</td>
</tr>
<tr>
<td>Debug</td>
<td>Ctrl-F5</td>
<td>Option-Command-R</td>
</tr>
</tbody>
</table>

This table lists the default key bindings for various operations within the CodeWarrior IDE. The columns indicate the key bindings for Windows and Mac OS environments.
Default CodeWarrior Key Bindings

Debug

The key bindings in Table 12.9 contains the key bindings for handling commands on the Debug menu in the CodeWarrior IDE.

Table 12.6 Debug menu key bindings

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td></td>
<td>Control-P</td>
</tr>
<tr>
<td>Kill</td>
<td>Shift-F5</td>
<td>Control-K</td>
</tr>
<tr>
<td>Step Over</td>
<td>F10</td>
<td>Control-S</td>
</tr>
<tr>
<td>Step Into</td>
<td>F11</td>
<td>Control-T</td>
</tr>
<tr>
<td>Step Out</td>
<td>Shift-F11</td>
<td>Control-U</td>
</tr>
<tr>
<td>View Memory</td>
<td>Alt-6</td>
<td>Control-M</td>
</tr>
<tr>
<td>View Memory As</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch to Monitor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data

The key bindings in Table 12.9 contains the key bindings for handling debugger variable displays in the IDE.

Table 12.7 Data menu key bindings

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Expression</td>
<td></td>
<td>Control-N</td>
</tr>
<tr>
<td>Open Variable Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open Array Window</td>
<td></td>
<td>Control-A</td>
</tr>
</tbody>
</table>
## Default CodeWarrior Key Bindings

### Window Menu

The key bindings in Table 12.8 contains the key bindings for handling many common windows in the CodeWarrior IDE.

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>View As</td>
<td>Control-Y</td>
<td></td>
</tr>
<tr>
<td>Set Watchpoint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear Current</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Watchpoint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signed Decimal</td>
<td>Alt-Shift-D</td>
<td>Control-Shift-D</td>
</tr>
<tr>
<td>Unsigned Decimal</td>
<td>Alt-Shift-U</td>
<td>Control-Shift-U</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>Alt-Shift-H</td>
<td>Control-Shift-H</td>
</tr>
<tr>
<td>Character</td>
<td>Alt-Shift-C</td>
<td>Control-Shift-C</td>
</tr>
<tr>
<td>C String</td>
<td>Alt-Shift-S</td>
<td>Control-Shift-S</td>
</tr>
<tr>
<td>Pascal String</td>
<td>Alt-Shift-P</td>
<td>Control-Shift-P</td>
</tr>
<tr>
<td>Floating Point</td>
<td>Alt-Shift-F</td>
<td>Control-Shift-F</td>
</tr>
<tr>
<td>Enumeration</td>
<td>Alt-Shift-E</td>
<td>Control-Shift-E</td>
</tr>
<tr>
<td>Fixed</td>
<td></td>
<td>Control-Shift-I</td>
</tr>
<tr>
<td>Fract</td>
<td></td>
<td>Control-Shift-R</td>
</tr>
</tbody>
</table>

### Table 12.8 Window menu key bindings

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tile Vertical</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Default CodeWarrior Key Bindings

### Window Menu

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoom Window</td>
<td>Ctrl-/-</td>
<td>Command-/-</td>
</tr>
<tr>
<td>Save Default Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser Catalog Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class Hierarchy Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Class Browser</td>
<td>Alt-F12</td>
<td></td>
</tr>
<tr>
<td>Build Progress Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Errors &amp; Warnings Window</td>
<td>Ctrl-I</td>
<td>Command-I</td>
</tr>
<tr>
<td>Project Inspector</td>
<td>Alt-Enter</td>
<td></td>
</tr>
<tr>
<td>Processes Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expressions Window</td>
<td>Alt-3</td>
<td></td>
</tr>
<tr>
<td>Global Variables Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Breakpoints Window</td>
<td>Alt-F9</td>
<td></td>
</tr>
<tr>
<td>Watchpoints Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registers Window</td>
<td>Alt-5</td>
<td></td>
</tr>
<tr>
<td>FPU Registers Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close All Variables Windows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ToolServer Worksheet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Default Project</td>
<td>Ctrl-0</td>
<td>Command-0</td>
</tr>
</tbody>
</table>
## Default CodeWarrior Key Bindings

### Window Menu

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Document 1</td>
<td>Ctrl-1</td>
<td>Command-1</td>
</tr>
<tr>
<td>Select Document 2</td>
<td>Ctrl-2</td>
<td>Command-2</td>
</tr>
<tr>
<td>Select Document 3</td>
<td>Ctrl-3</td>
<td>Command-3</td>
</tr>
<tr>
<td>Select Document 4</td>
<td>Ctrl-4</td>
<td>Command-4</td>
</tr>
<tr>
<td>Select Document 5</td>
<td>Ctrl-5</td>
<td>Command-5</td>
</tr>
<tr>
<td>Select Document 6</td>
<td>Ctrl-6</td>
<td>Command-6</td>
</tr>
<tr>
<td>Select Document 7</td>
<td>Ctrl-7</td>
<td>Command-7</td>
</tr>
<tr>
<td>Select Document 8</td>
<td>Ctrl-8</td>
<td>Command-8</td>
</tr>
<tr>
<td>Select Document 9</td>
<td>Ctrl-9</td>
<td>Command-9</td>
</tr>
</tbody>
</table>
## Misc.

The key bindings in Table 12.9 contains the key bindings for handling many common windows in the CodeWarrior IDE.

**Table 12.9 Misc. key bindings**

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goto Header/Source File</td>
<td>Ctrl-Tab</td>
<td>Command-Tab</td>
</tr>
<tr>
<td>Goto Previous Error Message</td>
<td>Ctrl-Shift-Up Arrow</td>
<td>Option-Command-Up Arrow</td>
</tr>
<tr>
<td>Goto Next Error Message</td>
<td>Ctrl-Shift-Down Arrow</td>
<td>Option-Command-Down Arrow</td>
</tr>
</tbody>
</table>

## Editor Commands

The key bindings in Table 12.10 contains the key bindings for handling many common windows in the CodeWarrior IDE.

**Table 12.10 Editor window key bindings**

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move Character Left</td>
<td>Left Arrow</td>
<td>Left Arrow</td>
</tr>
<tr>
<td>Move Character Right</td>
<td>Right Arrow</td>
<td>Right Arrow</td>
</tr>
<tr>
<td>Move Word Left</td>
<td>Ctrl-Left Arrow</td>
<td>Option-Left Arrow</td>
</tr>
<tr>
<td>Move Word Right</td>
<td>Ctrl-Right Arrow</td>
<td>Option-Right Arrow</td>
</tr>
<tr>
<td>Move Sub-word Left</td>
<td>Control-Left Arrow</td>
<td></td>
</tr>
</tbody>
</table>
## Default CodeWarrior Key Bindings

*Editor Commands*

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move Sub-word Right</td>
<td></td>
<td>Control-Right</td>
</tr>
<tr>
<td></td>
<td>Home</td>
<td>Arrow</td>
</tr>
<tr>
<td>Move to Start of Line</td>
<td></td>
<td>Command-Left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Arrow</td>
</tr>
<tr>
<td>Move to End of Line</td>
<td>End</td>
<td>Command-Right</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Arrow</td>
</tr>
<tr>
<td>Move Line Up</td>
<td>Up Arrow</td>
<td>Up Arrow</td>
</tr>
<tr>
<td>Move Line Down</td>
<td>Down Arrow</td>
<td>Down Arrow</td>
</tr>
<tr>
<td>Move to Top of Page</td>
<td>Page Up</td>
<td>Option-Up Arrow</td>
</tr>
<tr>
<td>Move to Bottom of Page</td>
<td>Page Down</td>
<td>Option-Down Arrow</td>
</tr>
<tr>
<td>Move to Top of File</td>
<td>Ctrl-Home</td>
<td>Command-Up Arrow</td>
</tr>
<tr>
<td>Move to Bottom of File</td>
<td>Ctrl-End</td>
<td>Command-Down Arrow</td>
</tr>
<tr>
<td>Delete Character Left</td>
<td>Backspace</td>
<td>Delete</td>
</tr>
<tr>
<td>Delete Character Right</td>
<td>Del</td>
<td>del</td>
</tr>
<tr>
<td>Character Select Left</td>
<td>Shift-Left Arrow</td>
<td>Shift-Left Arrow</td>
</tr>
<tr>
<td>Character Select Right</td>
<td>Shift-Right Arrow</td>
<td>Shift-Right Arrow</td>
</tr>
<tr>
<td>Select Word Left</td>
<td></td>
<td>Option-Shift-Left Arrow</td>
</tr>
<tr>
<td>Select Word Right</td>
<td></td>
<td>Option-Shift-Right Arrow</td>
</tr>
<tr>
<td>Select Sub-word Left</td>
<td></td>
<td>Control-Shift-Left Arrow</td>
</tr>
<tr>
<td>Select Sub-word Right</td>
<td></td>
<td>Control-Shift-Right Arrow</td>
</tr>
</tbody>
</table>
# Default CodeWarrior Key Bindings

## Editor Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Line Up</td>
<td>Shift-Up Arrow</td>
<td>Shift-Up Arrow</td>
</tr>
<tr>
<td>Select Line Down</td>
<td>Shift-Down Arrow</td>
<td>Shift-Down Arrow</td>
</tr>
<tr>
<td>Select to Start of Line</td>
<td>Shift-Home</td>
<td>Shift-Command-Left Arrow</td>
</tr>
<tr>
<td>Select to End of Line</td>
<td>Shift-End</td>
<td>Shift-Command-Right Arrow</td>
</tr>
<tr>
<td>Select to Start of Page</td>
<td>Shift-Page Up</td>
<td>Option-Shift-Up Arrow</td>
</tr>
<tr>
<td>Select to End of Page</td>
<td>Shift-Page Down</td>
<td>Option-Shift-Down Arrow</td>
</tr>
<tr>
<td>Select to Start of File</td>
<td>Ctrl-Shift-Home</td>
<td>Shift-Command-Up Arrow</td>
</tr>
<tr>
<td>Select to End of File</td>
<td>Ctrl-Shift-End</td>
<td>Shift-Command-Down Arrow</td>
</tr>
<tr>
<td>Scroll Line Up</td>
<td>Ctrl-Up Arrow</td>
<td>Control-Up Arrow</td>
</tr>
<tr>
<td>Scroll Line Down</td>
<td>Ctrl-Down Arrow</td>
<td>Control-Down Arrow</td>
</tr>
<tr>
<td>Scroll Page Up</td>
<td>Page Up</td>
<td></td>
</tr>
<tr>
<td>Scroll Page Down</td>
<td>Page Down</td>
<td></td>
</tr>
<tr>
<td>Scroll to Top of File</td>
<td>Home</td>
<td></td>
</tr>
<tr>
<td>Scroll to End of File</td>
<td>End</td>
<td></td>
</tr>
<tr>
<td>Find Symbols with Prefix</td>
<td>Ctrl-`</td>
<td>Control-`</td>
</tr>
<tr>
<td>Find Symbols with Substring</td>
<td>Ctrl-Shift-`</td>
<td>Control-Shift-`</td>
</tr>
<tr>
<td>Get Next Symbol</td>
<td>Ctrl-`</td>
<td>Control-`</td>
</tr>
<tr>
<td>Get Previous Symbol</td>
<td>Ctrl-`</td>
<td>Control-`</td>
</tr>
</tbody>
</table>
Prefix Keys

The key bindings in Table 12.11 contains the key bindings for handling prefixes.

<table>
<thead>
<tr>
<th>Command</th>
<th>Windows</th>
<th>Mac OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quote</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prefix Key 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prefix Key 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prefix Key 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This chapter introduces and discusses the Apple Event and AppleScript support provided by the Metrowerks CodeWarrior IDE.

CodeWarrior Apple Events Overview

This chapter discusses the AppleScript and Apple Event commands and classes supported in CodeWarrior. You should read this chapter if you would like to learn how to enhance and extend the capabilities of the CodeWarrior IDE.

The CodeWarrior IDE supports Apple Events. By scripting these Apple Events using AppleScript or another scripting editor, such as Frontier, it is possible to execute many CodeWarrior IDE commands without using the IDE directly. Scripting the CodeWarrior IDE is a way to automate repetitive tasks that do not need user interaction. There are many exciting things that you can do with AppleScript to harness the power of the IDE, such as automate builds, generate files automatically, and configure settings.

If you are primarily interested in writing scripts that manipulate and automate the IDE, then you are probably most interested in using AppleScript to put together an ensemble of Apple Events. If you would like to write program code to drive the CodeWarrior IDE from within your own computer program or tools, then you are probably most interested in the lower-levels of Apple Events, and not in AppleScript. This chapter is oriented toward working with
AppleScript, but there is a discussion of low-level Apple Event coding in the last section.

**TIP:** Look at the AppleScripts in the \( \text{(Scripts)} \) folder of the Metrowerks CodeWarrior folder for lots of cool AppleScripts. Reviewing these scripts will save you time when learning to write your own.

This chapter is not a tutorial. If you want to learn how to edit, save, and run AppleScripts you will not find tutorial information here. Instead, refer to other tools and sources of information listed in “AppleScript Tools and Reference Material” on page 406 for more information.

The topics in this chapter are:

- AppleScript Tools and Reference Material
- Writing Your First CodeWarrior IDE AppleScript
- CodeWarrior IDE AppleScript Events
- CodeWarrior IDE AppleScript Classes
- Coding with CodeWarrior IDE and Apple Events

**TIP:** You can run AppleScripts from within the CodeWarrior IDE. To learn about how to do this, read the section of this manual entitled “Scripts Menu (Mac OS)” on page 387.

---

**AppleScript Tools and Reference Material**

You can find the tools provided by Apple Computer for editing running AppleScripts on the CodeWarrior Reference CD in the MacOS System Extensions folder. Use the installer provided there to install the tools on your system.

Other Editing and debugging tools are available from third-party vendors. This is not an exhaustive list, or an endorsement of these
products, but these are worth evaluating if you are going to do much AppleScripting.

- Script Debugger (Late Night Software)
- Scripter 2.0 (Main Event Software)

If you are a subscriber to the Apple Developer CD program or Apple Developer Mailing, you will find good information on Apple Events and AppleScripting on the CD’s.

For more information on using and writing AppleScripts, you may want to consult other publications, such as:

- *AppleScript Language Guide: English Dialect* (Addison-Wesley)
- *Danny Goodman’s AppleScript Handbook* (Random House)
- *The Tao of AppleScript* (Hayden Books)
- *Applied Mac Scripting* (M & T Books)

For information on more advanced topics such as writing your own Scripting Additions, or how to use the standard Scripting Additions, refer to *AppleScript Scripting Additions Guide* (Apple Computer).


Finally, for documentation on using low-level AppleEvents in program code, refer to *Inside Macintosh: Interapplication Communication* (Addison-Wesley).

**Writing Your First CodeWarrior IDE AppleScript**

To get started with AppleScript and the CodeWarrior IDE, let’s take a look at a simple script that opens the IDE, brings it to the foreground on the Mac, opens a project, removes the binaries, and starts a build of the project. This script, shown in Listing 12.1, is something that could be double-clicked to automatically do all these operations unattended.
Listing 12.1  My First CodeWarrior AppleScript

tell application “CodeWarrior IDE 2.1” (* go! *)
activate (* bring CW to the front *)
open file “SD:MyProj:MyProject.68K.µ”
Remove Binaries
Make Project
end tell

You can imagine how convenient it will be to automate many tasks with AppleScript from this short example. Try entering this script in your Editor, such as Apple’s Script Editor that comes with the AppleScript 1.1 software on the CodeWarrior Reference CD, and get it to run.

After getting this short example to run, you will probably be motivated to try some more extensive examples.

CodeWarrior IDE AppleScript Events

In general, Apple Events are grouped in categories or “suites” of events that provide some common theme for the events. There is a “Required” suite of events that includes open, print, quit and run. All scriptable applications should support the required suite. There are other suites of events defined in the Apple Event Registry document. In addition, there are other suites of events that are application-specific.

For many of the things that you probably want to do with the CodeWarrior IDE, it really isn’t a concern which suite an event is from most of the time. However, you can view the “dictionary” of Apple Events that an application supports using the Open Dictionary command of your Script Editor. See the documentation that came with your editor for information about viewing the dictionary.

In this section, we discuss how to handle errors in AppleScript, and several categories of events that you can use to control the CodeWarrior IDE.

- Processing Errors
Mac OS CodeWarrior Scripting

CodeWarrior IDE AppleScript Events

- Required Events
- File Handling Events
- Building Events
- Status/Query Events
- Navigation Events

Parameters

Some Apple Events listed in this section require a parameter called filename-list. The filename-list represents a single filename or a list of filenames and/or aliases. A single filename is a quoted character string. A list of filenames is enclosed in braces, {}, with the filenames separated by commas.

Listing 12.2 Example values for filename

```script
"myprogram.c"
{"startup.p", "printout.p", "drawbox.p"}
{"codechecker.cpp"}
{"HD:CodeWarrior f:My Projects:hello.c"}
file "myprogram.c"
alias "myprogram.c"
```

Processing Errors

When an AppleEvent is sent to CodeWarrior, errors may be returned to the script. Errors are not always evil occurrences, as sometimes you will want to trap errors to make your script do other things in response to the current conditions. Errors can be generated from the operating system, or from the application you’re trying to script. Errors for the operating system are documented in Appendix C of the AppleScript Language Guide. Errors generated by the CodeWarrior IDE are documented here.

Errors are usually returned through the normal Error-return channel. However, for events that process a list of files, the errors are returned in the result (a built-in AppleScript variable). The list,
with each member corresponding to an input file, is returned as the event’s result with each list member.

In addition to operating system errors, such as out of memory errors, the error codes listed in Table 12.12 may also be returned.

**Table 12.12 CodeWarrior keyAEResult result codes (typeShortInteger)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>noErr</td>
<td>0</td>
</tr>
<tr>
<td>errShell_ActionFailed</td>
<td>1</td>
</tr>
<tr>
<td>errShell_FileNotFound</td>
<td>2</td>
</tr>
<tr>
<td>errShell_DuplicateFile</td>
<td>3</td>
</tr>
<tr>
<td>errShell_CompileError</td>
<td>4</td>
</tr>
<tr>
<td>errShell_MakeFailed</td>
<td>5</td>
</tr>
<tr>
<td>(compile or link error)</td>
<td></td>
</tr>
<tr>
<td>errShell_NoOpenProject</td>
<td>6</td>
</tr>
<tr>
<td>errShell_WindowNotOpen</td>
<td>7</td>
</tr>
<tr>
<td>errShell_SegmentNotFound</td>
<td>8</td>
</tr>
</tbody>
</table>

The result parameter, keyAEResult, is not set if there is an error while interpreting the AppleEvent (running out of memory, supplying a bad parameter type, and so on). In such cases, an error code is returned in the standard keyErrorNumber parameter.

**Listing 12.3** gives an example of an AppleScript that handles an error.

**Listing 12.3 Error handling in AppleScript**

```plaintext
try
tell application "CodeWarrior IDE 2.1"
  set doclist to (Get Open Documents)
end tell
on error number errnum
```
To see more examples of error handling in scripts, review some of the scripts in the (Scripts) folder in the same folder as your CodeWarrior IDE application.

**Required Events**

There are four events that are required for every application to implement that claims to be AppleScriptable. This section discusses these events and their syntax.

The events covered in this section are:

- Open
- Print
- Quit
- Run

**Open**

**Purpose** This event tells the CodeWarrior IDE to open the specified files.

`Open filename-list [ converting ] expression`

If `converting` is specified, any project files that were created with previous versions of the CodeWarrior IDE will be updated.

**Listing 12.4 Example for Open**

```
Open "HD:MyProject.µ" converting yes
Open "HD:MyProject.µ"
```

**Print**

**Purpose** This event tells the CodeWarrior IDE to print the specified files.
Quit

**Purpose**  This event tells the CodeWarrior IDE to quit.

Run

**Purpose**  This event is sent to an application when it is double-clicked. When an application receives this event it should launch itself.

**File Handling Events**

You will want to use the File Handling Apple Events of the CodeWarrior IDE to do things like add and remove files in a project, close a window, create and close a project, and save copies of files.

Here are the events covered in this section:

- Add Files
- Add New Files
- Close Project
- Close Window
- Create Project
- Remove Files
- Remove Target Files
- Save Error Window As
- Select
- Set Modification Date

**Add Files**

**Purpose**  Adds the specified files to the current project.

```
Add Files filename-list  [ to segment number ]
```
Description

This event is equivalent to the `Add Files` command in the Project Menu. The `filename-list` parameter describes a single filename or list of filenames to add to the current project. The optional `segment` parameter specifies the segment in the project in which to add the files. Replace `number` with the segment number to place the files in. The default is to create a new segment.

Returns

A list of errors. The result code for each file added to the project can either return the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`
- `errShell_FileNotFound`
- `errShell_DuplicateFile`
- `errShell_NoOpenProject`

Listing 12.5  Examples for add files

```
add files "MyFile.c"
add files "MyFile.c" to segment 2
add files {"MyFile.c", "MyFile2.c"}
add files {"MyFile.c", "MyFile2.c"} to segment 3
```

Add New Files

Purpose

Adds the specified files to the current project or target.

```
Add New Files  filename-list  [ to segment number ]
```

Description

This event is equivalent to the `Add Files` command in the Project Menu. The `filename-list` parameter describes a single filename or list of filenames to add to the current project. The optional `segment` parameter specifies the segment in the project in which to add the files. Replace `number` with the segment number to place the files in. The default is to create a new segment.
Returns  A list of errors. The result code for each file added to the project can either return the value of an OSErr (Operating System Error) or one of the following values:

- noerr
- errShell_FileNotFound
- errShell_DuplicateFile
- errShell_NoOpenProject

Listing 12.6  Examples for add new files

```markdown
add new files "MyFile.c"
add new files "MyFile.c" to segment 2
add new files {"MyFile.c", "MyFile2.c"}
add new files {"MyFile.c", "MyFile2.c"} to segment 3
```

Close Project

Purpose  Closes the current project.

Close Window

Purpose  Closes editor windows.

Description  This event is equivalent to the Close command in the File Menu. If filename is a string, Close Window first tries to find the first matching window name, searching front to back. If no window name matches filename, then Close Window causes a search for a matching filename. To specify read/only windows, add “[r/o 1]” to the end of filename.
Close Window "hello.c [r/o 1]"

Table 12.13 lists file saving options used with the close window AppleScript command.

### Table 12.13 Saving options (‘savo’)

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes (Save changes)</td>
<td>‘yes’</td>
</tr>
<tr>
<td>no (Do not save changes)</td>
<td>‘no’</td>
</tr>
<tr>
<td>ask (Ask the user whether to save)</td>
<td>‘ask’</td>
</tr>
</tbody>
</table>

The optional saving parameter determines if the windows contents are saved before closing the window.

- If `status` is `yes`, save the window’s contents.
- If `status` is `no`, discard changes made to the file.
- If `status` is `ask`, prompt the user whether or not to save the file.

**Returns**  None

**Listing 12.7 AppleScript Example for Close Window**

```apple-script
Close Window "untitled"
-- Closes first "untitled" window.
Close Window "hello.c" saving yes
Close Window "main.c [r/o 1]"
-- Closes read/only window.
```

---

**Create Project**

**Purpose**  Creates a new project file.

Create Project `filename` [from stationery ]alias
**Mac OS CodeWarrior Scripting**

*CodeWarrior IDE AppleScript Events*

**Description**
Performs the **New Project** command in the **File Menu**. Replace `filename` with a single filename for the new project. If `from stationery` is specified, project stationery specified by `alias` is used to create the new project.

**Returns**
The result code can have the following values:

- `noErr`
- `errShell_ActionFailed`

**Listing 12.8** 
**Examples for create project**

Create Project "HardDisk:Projects:MyProject.m"
Create Project "Foobe" from "stationery "HD:Dev:Metrowerks" & "CodeWarrior:(Project Stationery):MacOS:C/C++:" & "Basic Toolbox 68k:Basic Toolbox 68k.m"
Create Project "::sample project:sample.m"

---

**Remove Files**

**Purpose**
Remove the specified file(s) from the current project.

**Remove Files** *filename*

**Description**
Performs the equivalent of the **Remove Selected Items** command in the **Project Menu**. Replace `filename` with a single file or a list of files to remove from the current project.

**Returns**
A list of errors. The result code can have one of the following values:

- `noErr`
- `errShell_FileNotFound`
- `errShell_NoOpenProject`
### Listing 12.9 Examples for remove files

Remove Files "MyFile.c"
Remove Files { "MyFile.c", "YourFile.c" }

### Remove Target Files

**Purpose**  
Remove files from the current target or project.

**Description**  
Performs the equivalent of the Remove Selected Items command in the Project Menu.

**Returns**  
None.

### Save Error Window As

**Purpose**  
Save the contents of the message window as a text file.

**Description**  
Performs the equivalent of the Save A Copy As command in the File Menu when the Message window is active. The contents of the Message window are saved with the name filename.

**Returns**  
None.

### Listing 12.10 Example for Save Error Window As

Precompile "main.c"
Save Error Window As "main.c results"

### Select

**Purpose**  
Selects an object from an open document in the CodeWarrior Editor.

**Select reference**
**Description**  The parameter *reference* is the object to select.

**Listing 12.11  Select AppleScript Example**

```applescript
select text from character 5 to character 10 of line 8 of document 1
```

**Set Modification Date**

**Purpose**  Sets the modification date of the specified file(s).

```applescript
Set Modification Date filename-list to date
```

**Returns**  A list of results of type short integer, or, if the external editor option is specified, a list of records of type ‘ErrM’.

**Building Events**

Here are the events discussed in this section:

- Bring Up to Date
- Build
- Check File Syntax
- Check Syntax
- Compile
- Compile File
- Disassemble File
- Make Project
- Precompile
- Preprocess
- Remove Binaries
- Remove Object Code
- Run
- Run Project
• Touch
• Update Project

Bring Up to Date

Purpose Bring the current target or project up to date.

Description Performs the Bring up to Date command in the Project Menu.

Returns By default, this command returns nothing.

Build

Purpose Build the current target or project.

Description Performs the Make command in the Project Menu.

Returns By default, Build returns nothing.

Check File Syntax

Purpose Checks the syntax of the specified file(s).

Check File Syntax filename-list

Description This event is equivalent to performing the Check Syntax command in the Project Menu. Replace filename-list with a single filename or a list of filenames in the project.

By default, Check File Syntax returns a list of short integer result codes for each file checked. A result code can either be the value of an OSErr (Operating System Error) or one of the following values:

• noerr
• errShell_FileNotFound
Check Syntax

Purpose
Checks the syntax of the specified file(s).

Description
This event is equivalent to performing the Check Syntax command in the Project Menu. Replace `filename-list` with a single filename or a list of filenames in the project.

By default, Check Syntax returns a list of short integer result codes for each file checked. A result code can either be the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

If the `ExternalEditor` option is used, the environment returns the Message window contents instead of the usual list of short integer results. The AppleEvent keyword for `ExternalEditor` is `'Errs'`. It takes a boolean parameter.

Returns
A list of results of type short integer, or, if the `ExternalEditor` option is specified, a list of records of type `'ErrM'`.

Listing 12.12 Examples for Check Syntax

Check File Syntax "MyFile.c"
Check File Syntax "{"MyFile.c", "YourFile.c"} with ExternalEditor

Returns
A list of results of type short integer.
Listing 12.13  Examples for Check Syntax

Check Syntax "MyFile.c"
Check Syntax {"MyFile.c", "YourFile.c"} with ExternalEditor

Compile

Purpose  Compile the specified file(s).

Compile  filename-list  [with  ExternalEditor]

Description  This event is equivalent to performing the Compile command on the Project Menu. Replace filename-list with a single filename or a list of filenames.

By default, Compile returns a list of short integer result codes for each compiled file. A result code can be an OSErr value (Operating System Error) or one of the following:

- noerr
- errShell_FileNotFound
- errShell_CompileError
- errShell_NoOpenProject

If the ExternalEditor option is specified, the environment returns the Message window contents as a list of 'ErrM' objects.

Returns  A list of errors of type short integer or, if ExternalEditor is specified, of type 'ErrM'.

Listing 12.14  Examples for Compile

Compile "MyFile.c"
Compile {"MyFile.c", "YourFile.c"}

Compile File

Purpose  Compile the specified file(s).
**Mac OS CodeWarrior Scripting**

*CodeWarrior IDE AppleScript Events*

---

**Compile File**  
`filename-list`

**Description**  
This event is equivalent to performing the `Compile` command on the **Project Menu**. Replace `filename-list` with a single filename or a list of filenames.

By default, `Compile` returns a list of short integer result codes for each compiled file. A result code can be an OSErr value (Operating System Error) or one of the following:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

**Returns**  
A list of errors of type short integer.

**Listing 12.15  Examples for Compile**

- Compile File "MyFile.c"
- Compile File {"MyFile.c", "YourFile.c"}

**Disassemble File**

**Purpose**  
Disassemble the specified file(s).

**Disassemble File**  
`filename-list`

**Description**  
This event is equivalent to performing the Disassemble command on the **Project Menu**. Replace `filename-list` with a single filename or a list of filenames.

**Returns**  
A list of errors of type short integer.
Listing 12.16  Examples for Disassemble File

Disassemble File "MyFile.c"
Disassemble File {"MyFile.c", "YourFile.c"}

Make Project

Purpose  Make the current project.

Make Project [with ExternalEditor]

Description  Performs the Make command in the Project menu.

If the ExternalEditor option is used, the environment returns the Message window contents.

Returns  By default, Make Project returns nothing. If ExternalEditor is specified, Make Project returns a list of errors of type 'ErrM'.

Precompile

Purpose  Precompile the specified file.

Precompile source saving as destination [ with ExternalEditor ]

Description  This event is equivalent to the Precompile command in the Project Menu. Replace source with the name of a file to precompile. Replace destination with the filename of the precompiled header.

If the ExternalEditor option is used, the environment returns the Message window contents.

Returns  By default, Precompile returns nothing. If ExternalEditor is specified, Precompile returns a list of errors of type 'ErrM'.
Listing 12.17  AppleScript Example for Precompile

Precompile "MyHeaders.c" saving as "MyHeaders"
Precompile "tip.c" saving as "tip.pch" with ExternalEditor

Preprocess

Purpose  Preprocess the specified file.

Description  This event is equivalent to the Preprocess command in the Project Menu. Replace source with the name of a file to preprocess.

If the ExternalEditor option is used, the environment returns the Message window contents.

Returns  By default, Preprocess returns nothing. If ExternalEditor is specified, Preprocess returns a list of errors of type 'ErrM'.

Listing 12.18  AppleScript Example for Preprocess

Preprocess "MyHeaders.c"
Precompile "tip.c" with ExternalEditor

Remove Binaries

Purpose  Remove the binary object code from the current project.

Description  Performs the equivalent of the Remove Object Code command in the Project Menu.

Returns  None.
Remove Object Code

**Purpose**
Remove the binary object code from the current target or project.

**Description**
Performs the equivalent of the Remove Object Code command in the Project Menu.

**Returns**
None.

Run

**Purpose**
Run the current project or target.

**Description**
Performs the equivalent of the Run command in the Project menu. This event builds then executes the current target if there are no compile or link errors.

**Returns**
By default, Run Project returns nothing.

Run Project

**Purpose**
Run the current project

**Description**
Performs the equivalent of the Run command in the Project menu. This event builds then executes the current project if there are no compile or link errors.

If the ExternalEditor option is used, the environment returns the Message window contents.

If the SourceDebugger option is used, the environment launches the successfully built project into the source-level debugger.
Mac OS CodeWarrior Scripting
CodeWarrior IDE AppleScript Events

Returns
By default, `Run Project` returns nothing. If `ExternalEditor` is specified, `Run Project` returns a list of errors that occurred when running the project, of type 'ErrM'.

Listing 12.19  Examples for Run Project

Run Project with SourceDebugger
Run Project with ExternalEditor

Touch

Purpose
Touch the specified file(s).

Touch  `filename`

Description
Performs the equivalent of clicking the `Touch column` in a Project window. Touching a file forces it to be recompiled during a make operation. Replace `filename` with a single file or a list of files to touch.

For more on touching a file to be recompiled, consult

Returns
A list of errors. Each result code can have one of the following values:

- `noErr`
- `errShell_FileNotFound`
- `errShell_NoOpenProject`

Listing 12.20  Examples for Touch

Touch "MyFile.c"
Touch { "MyFile.c", "YourFile.c" }

Update Project

Purpose
Update the current project.

Update Project [ with ExternalEditor ]
**Description**  
This command is equivalent to the Update Project command in the Project menu. If the *ExternalEditor* option is used, the environment returns the Message window contents.

**Returns**  
By default, *Update Project* returns nothing. If *ExternalEditor* is specified, *Update Project* returns a list of errors of type 'ErrM'.

### Status/Query Events

Here are the events discussed in this section:

- **Get**
- **Set**
- **Get Definition**
- **Get Member Function Names**
- **Get Nonsimple Classes**
- **Get Open Documents**
- **Get Preferences**
- **Get Project File**
- **Set Current Target**
- **Set Default Project**
- **Get Project Specifier**
- **Get Segments**
- **Is In Project**
- **Reset File Paths**
- **Close**
- **Count**
- **Make**

**Get**

**Purpose**  
Get the object referenced.
Mac OS CodeWarrior Scripting
CodeWarrior IDE AppleScript Events

Get reference [ as list of typeclass ]

Description
The parameter reference is the object whose data is to be returned. The parameter typeclass is the desired types for the data, in order of preference.

Set

Purpose
Set the object referenced.

Set reference to anything

Description
The parameter reference is the object whose data is to be changed. The parameter anything is the new value for the object.

Listing 12.21 Set AppleScript Example

tell application "CodeWarrior IDE 1.7.4" to
set numClasses to the count of classes

Get Definition

Purpose
Query the location(s) of a globally-scoped function or data object for the current project.

Get Definition string

Description
The string is the name of the symbol you are interested in.

Returns
Record containing a list of the function information.

Listing 12.22 Example for Get Definition

Get Definition “main”
Get Member Function Names

**Purpose**
Get a list of all the member functions of a class object.

**Returns**
List containing a list of the information.

**Listing 12.23 Get Member Function Names AppleScript Example**
Get Member Function Names class "CPowerTelnetApp"

Get Nonsimple Classes

**Purpose**
Get a list of all the member functions of a class object.

**Returns**
List containing a list of the information.

**Listing 12.24 Get Nonsimple Classes AppleScript Example**
Get Nonsimple Classes class "CPowerTelnetApp"

Get Open Documents

**Purpose**
Get the list of open documents

**Returns**
List of documents in records of type 'docu'. See Table 12.51 for more information

Get Preferences

**Purpose**
Get settings from a panel
Get Preferences [ of pref-list ] from panel panel-name

Description
The panel-name must be the name of the preference panel file and not the name that appears in the preferences dialog. For example, to set C/C++ Language options, use "C/C++ Compiler" as the panel-name and not "C/C++ Language".

Returns
Record containing a list of the requested preferences. If you do not include pref-list, it returns all the preferences for panel-name.

Listing 12.25 Examples for Get Preferences
Get Preferences from panel "C/C++ Compiler"
Get Preferences of {File Name, SIZE Flags} from panel "PPC Project"

Set Preferences
Purpose
Specify the settings for a panel.

Set Preferences of panel panel-name to record

Description
Performs the equivalent of setting options using either the Preferences or Settings dialog boxes. This event lets you set the properties of the current project. It is not necessary to specify every preference, those not mentioned in the record retain their settings. The properties for different panels are listed in various tables from Table 12.14 to Table 12.49.

The panel-name must be the name of the preference panel file and not the name that appears in the preferences dialog. For example, to set C/C++ Language options, set panel-name to "C/C++ Compiler" and not "C/C++ Language".

Returns
None.
Listing 12.26  Examples for Set Preferences

Set Preferences of panel "PPC Project" to { ¬
    File Name:"MyProgram", File Creator:"Mine", SIZE Flags:23008 ¬
}
Set Preferences of panel "C/C++ Compiler" to { ¬
    Prefix File: "MacHeaders", ¬
    Activate CPlusPlus: TRUE, ¬
    Require Function Prototypes: FALSE ¬
}
Set Preferences of panel "C/C++ Warnings" to { ¬
    Extended Error Checking: TRUE ¬
}

---

Get Project File

**Purpose**  Get information on a project entry.

Get Project File  *file-number*  *segment*  *seg-number*

**Returns**  The information of the specified entry in the current project as a record of type 'SrcF'. The *file-number* parameter specifies a file within its segment or group. The *seg-number* parameter specifies a segment or group within the project. Numbering for both parameters are short integers beginning at 1.

Listing 12.27  Examples for Get Project File

get project file 1 segment 1
    -- First entry in project
get project file 1 segment 2
    -- First entry in 2nd segment

---

Set Project File

**Purpose**  Sets information on a project entry.
Set Project File *filename* to *record*

**Description**
Changes the settings for the specified entry in the open project. The *record* parameter is of type 'SrcF'. Only the symbols and weak link fields are allowed. *Listing 12.28* shows how to set weak linking for a library InterfaceLib that is in a project.

**Returns**
None.

*Listing 12.28*  **Example for Set Project File**

Set Project File “InterfaceLib” to {weak link: true}

**Set Current Target**

**Purpose**
Sets the current target for a project.

Set Current Target *name-of-target*

**Description**
This AppleEvent causes the target to change. This event would be useful when changing the target in a project, so that a new target can be built or otherwise operated on with other AppleEvents. To learn more information about setting the current target, refer to “Set Current Target” on page 372.

**Returns**
None.

*Listing 12.29*  **Example for Set Current Target**

Set Current Target “Muscle 68K”

**Set Default Project**

**Purpose**
Sets the default project.

Set Default Project *name-of-target*
Description

This AppleEvent causes the default project to change. To learn more information about setting a default project, refer to “Set Default Project” on page 372.

Returns

None.

Listing 12.30  Example for Set Default Project

Set Current Project “Muscle µ”

Get Project Specifier

Purpose

Get the filename of the project.

Returns

The name of the current project.

Get Segments

Purpose

Get the descriptions of all segments/groups in the open project.

Returns

List of documents in records of type 'Seg'. Refer to “Segment” on page 448 for more information

Set Segment

Purpose

Set preferences for the current project.

Description

Sets information for a segment or group in the open project. Segment numbering starts at 1. The record parameter is an object of type 'Seg'. Listing 12.31 shows how to rename a segment/group in a project. Refer to “Segment” on page 448 for more information.
Returns  None.

Listing 12.31  Example for Set Segment

Set Segment 1 to {name:"New Sources"}

Is In Project

Purpose  Are the specified file(s) are in the project?

Is In Project  filename

Description  Replace filename with a single filename or a list of filenames.

Returns  A list of errors. The result code for each specified file can have the following values:
    • noErr if the file is in the project
    • errShell_FileNotFound if file is not in the project.

Listing 12.32  Examples for Is In Project

Is In Project "SillyBalls.c"
Is In Project { "SillyBalls.c", "Initialize.c" }  

Reset File Paths

Purpose  Resets access paths for all files belonging to the open project.

Reset File Paths

Returns  None.

Close

Purpose  Close an object. The saving and saving  in parameters are optional, and have the range of values listed here.
Close reference [ saving yes/no/ask ] [ saving in alias ]

Returns  None.

Count

Purpose  Count the number of elements within an object.

Returns  An integer indicating the number of elements.

Make

Purpose  Make a new element. The as list of, as, with data, and with properties parameters are optional, and have the range of values listed here.

Returns  A reference to the new object(s).

Navigation Events

Here are the events discussed in this section:

• Goto Function
• Goto Line
• Open Browser

Goto Function

Purpose  Jumps to the specified function defined in the active editor window.

Goto Function name
Description  This event is equivalent to selecting a routine name in the active editor window's function pop-up menu. The insertion point does not move when a function is selected with this event.

Returns  None.

Listing 12.33  Examples for Goto Function

Goto Function "main"
Goto Function "SkipBlanks"

Description  Goto Line moves the insertion point to the specified line number, number, in the active editor window. If the line number specified exceeds the last line number, the insertion point is placed at the last line.

Returns  None.

Listing 12.34  Examples for Goto Line

Goto Line 1
Goto Line 493

Purpose  Display a class, member function, or data member object in a single class browser window. You cannot display a procedural function.

Returns  None.
CodeWarrior IDE AppleScript Classes

CodeWarrior events have several classes to let you control the CodeWarrior IDE actions and option settings. These classes are based on the items available in the Preferences and Target Settings dialog boxes.

The available classes are determined by the compiler (C/C++, Java, or Pascal) and for which processor it is generating object code (68k-based, Intel x86 or PowerPC-based Macintosh). In other words, the preference classes available for the Metrowerks C/C++ compiler that generates object code for the PowerPC are different from those available for the Metrowerks C compiler that generates 68K object code.

AppleScript classes for the Metrowerks environment are, for the most part, separated by preferences panel, project settings, and then by miscellaneous environment items.

- **Project Classes**—contains properties for each aspect of a project, from target parameters to final application settings.
- **Compiler Classes**—contains properties for each language and compiler, to configure compilation settings and warnings.
- **CodeGen Classes**—contains properties for each possible kind of code that can be generated by the CodeWarrior compilers.
- **Disassembler Classes**—contains properties for the disassemblers.
- **Linker Classes**—contains properties for the linker settings.
- **Build Classes**—contains properties for build environment settings and errors.
- **Browser Classes**—contains properties for the code browser.
Mac OS CodeWarrior Scripting
CodeWarrior IDE AppleScript Classes

- **Editor Classes**—contains properties for the CodeWarrior IDE text editor.
- **Object Classes**—contains properties that describe objects, including data members, classes and base classes, and member functions.
- **Misc Classes**—contains properties that describe miscellaneous aspects of the CodeWarrior IDE environment.

Many options that use a pop-up menu in the preference panel now use an enumerated type to specify their values, instead of an integer. For example, to set the Code Model, you should now use small, smart, or large, instead of 1, 2, or 3.

**WARNING!** Your script will not work if you use integer to set a property that expects a symbol.

**Project Classes**

- **68K Project**
- **PowerPC Project**
- **Java Project**
- **Win32/x86 Project**
- **Access Paths**
- **Path Information**
- **Target Settings**
- **File Mapping Information**
- **Segment**
- **Project File**

**68K Project**

_Table 12.14_ lists the 68K Project Class properties.
Table 12.14  68K Project Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Type</td>
<td>standard application</td>
</tr>
<tr>
<td></td>
<td>CFM68K application</td>
</tr>
<tr>
<td></td>
<td>code resource</td>
</tr>
<tr>
<td></td>
<td>library</td>
</tr>
<tr>
<td></td>
<td>shared library</td>
</tr>
<tr>
<td></td>
<td>MPW Too</td>
</tr>
<tr>
<td></td>
<td>Pilot Application</td>
</tr>
<tr>
<td></td>
<td>Pilot Code Resource</td>
</tr>
<tr>
<td>File Name</td>
<td>string</td>
</tr>
<tr>
<td>File Creator</td>
<td>string</td>
</tr>
<tr>
<td>File Type</td>
<td>string</td>
</tr>
<tr>
<td>Minimum Size</td>
<td>integer</td>
</tr>
<tr>
<td>Preferred Size</td>
<td>integer</td>
</tr>
<tr>
<td>SIZE Flags</td>
<td>integer (SIZE flag bits must be computed as an integer value)</td>
</tr>
<tr>
<td>SYM File</td>
<td>string</td>
</tr>
<tr>
<td>Resource Name</td>
<td>string</td>
</tr>
<tr>
<td>Display Dialogs</td>
<td>boolean</td>
</tr>
<tr>
<td>Merge To File</td>
<td>boolean</td>
</tr>
<tr>
<td>Resource Flags</td>
<td>small integer</td>
</tr>
<tr>
<td>Resource Type</td>
<td>small integer</td>
</tr>
<tr>
<td>Resource ID</td>
<td>small integer</td>
</tr>
<tr>
<td>Multi Segment</td>
<td>boolean</td>
</tr>
<tr>
<td>Name</td>
<td>Property Type</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Library Type</td>
<td>· A4 relative</td>
</tr>
<tr>
<td></td>
<td>· A5 relative</td>
</tr>
<tr>
<td></td>
<td>· CFM68K</td>
</tr>
<tr>
<td></td>
<td>· Pilot Library</td>
</tr>
<tr>
<td>Seg Type</td>
<td>string</td>
</tr>
<tr>
<td>CFM68K Code Generation</td>
<td>boolean</td>
</tr>
<tr>
<td>Stack Size</td>
<td>integer</td>
</tr>
<tr>
<td>Start-up Code</td>
<td>· standard</td>
</tr>
<tr>
<td></td>
<td>· The Debugger Aware</td>
</tr>
<tr>
<td></td>
<td>· custom</td>
</tr>
<tr>
<td>Header Type</td>
<td>· standard</td>
</tr>
<tr>
<td></td>
<td>· device driver</td>
</tr>
<tr>
<td></td>
<td>· desk accessory</td>
</tr>
<tr>
<td></td>
<td>· custom</td>
</tr>
<tr>
<td>RSEG Application</td>
<td>boolean</td>
</tr>
</tbody>
</table>

**PowerPC Project**

*Table 12.15* lists the PPC Project Class properties.
### Table 12.15 PPC Project Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Type</td>
<td>• standard application</td>
</tr>
<tr>
<td></td>
<td>• code resource</td>
</tr>
<tr>
<td></td>
<td>• library</td>
</tr>
<tr>
<td></td>
<td>• shared library</td>
</tr>
<tr>
<td></td>
<td>• MPW Tool</td>
</tr>
<tr>
<td></td>
<td>• Pilot Application</td>
</tr>
<tr>
<td></td>
<td>• Pilot Code Resource</td>
</tr>
<tr>
<td>File Name</td>
<td>string</td>
</tr>
<tr>
<td>File Creator</td>
<td>string</td>
</tr>
<tr>
<td>File Type</td>
<td>string</td>
</tr>
<tr>
<td>Minimum Size</td>
<td>integer</td>
</tr>
<tr>
<td>Preferred Size</td>
<td>integer</td>
</tr>
<tr>
<td>SIZE Flags</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>SIZE flag bits must be computed as an integer value</td>
</tr>
<tr>
<td>SYM File</td>
<td>string</td>
</tr>
<tr>
<td>Resource Name</td>
<td>string</td>
</tr>
<tr>
<td>Display Dialogs</td>
<td>boolean</td>
</tr>
<tr>
<td>Merge To File</td>
<td>boolean</td>
</tr>
<tr>
<td>Resource Flags</td>
<td>boolean</td>
</tr>
<tr>
<td>Resource Type</td>
<td>small integer</td>
</tr>
<tr>
<td>Resource ID</td>
<td>small integer</td>
</tr>
</tbody>
</table>
Mac OS CodeWarrior Scripting
CodeWarrior IDE AppleScript Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack Size</td>
<td>long integer</td>
</tr>
<tr>
<td>Header Type</td>
<td>· standard</td>
</tr>
<tr>
<td></td>
<td>· device driver</td>
</tr>
<tr>
<td></td>
<td>· desk accessory</td>
</tr>
<tr>
<td></td>
<td>· custom</td>
</tr>
</tbody>
</table>

Java Project

Table 12.16 lists the Java Project Class properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Class</td>
<td>string</td>
</tr>
<tr>
<td>Project Type</td>
<td>· java applet</td>
</tr>
<tr>
<td></td>
<td>· java application</td>
</tr>
<tr>
<td></td>
<td>· java library</td>
</tr>
<tr>
<td>Arguments</td>
<td>string</td>
</tr>
<tr>
<td>Compress</td>
<td>boolean</td>
</tr>
<tr>
<td>HTML Helper App</td>
<td>string</td>
</tr>
</tbody>
</table>

Win32/x86 Project

Table 12.17 lists the x86 Project Class properties.
Table 12.17  x86 Project Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Type</td>
<td>application</td>
</tr>
<tr>
<td></td>
<td>shared library</td>
</tr>
<tr>
<td></td>
<td>library</td>
</tr>
<tr>
<td>File Name</td>
<td>string</td>
</tr>
<tr>
<td>Min Heap Size</td>
<td>integer</td>
</tr>
<tr>
<td>Preferred Heap Size</td>
<td>integer</td>
</tr>
<tr>
<td>Base Address</td>
<td>integer</td>
</tr>
<tr>
<td>Max Stack Size</td>
<td>integer</td>
</tr>
<tr>
<td>Min Stack Size</td>
<td>integer</td>
</tr>
</tbody>
</table>

Access Paths

Table 12.18 lists the properties for the Access Paths Class.

Table 12.18  Access Paths Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Paths</td>
<td>list of path information records</td>
</tr>
<tr>
<td>Always Full Search</td>
<td>boolean</td>
</tr>
<tr>
<td>Convert Paths</td>
<td>boolean</td>
</tr>
<tr>
<td>System Paths</td>
<td>list of path information records</td>
</tr>
</tbody>
</table>

Path Information

A path information record may contain the properties shown in Table 12.19. It must contain at least the name field.
Table 12.19 Path Information Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>recursive</td>
<td>boolean</td>
</tr>
<tr>
<td>origin</td>
<td>- absolute</td>
</tr>
<tr>
<td></td>
<td>- project relative</td>
</tr>
<tr>
<td></td>
<td>- shell relative</td>
</tr>
<tr>
<td>host flags</td>
<td>- 1 (for the Mac OS)</td>
</tr>
<tr>
<td></td>
<td>- 2 (for Windows)</td>
</tr>
</tbody>
</table>

If you use a string instead of a path information record, CodeWarrior sets recursive to true and origin to project relative.

To clear all the access path entries listed in the Access Paths preference panel, set the User Paths or System Paths property to an empty list. For example, in AppleScript, this statement removes all entries, including the default entries, {Project f} and {Compiler f}:

```
Set Preferences of panel "Access Paths" to ¬
 {User Paths: {}, System Paths: {} }
```

To add a default entry back, add an access path record with the name set to ":" and with the origin set to project relative (for {Project f}) or shell relative (for {Compiler f}). For example, this statement sets User Paths to {Project f} and System Paths to {Compiler f}:

```
Set Preferences of panel "Access Paths" to ¬
 {User Paths:  {{name: ":", ¬
     origin: project relative}},¬
 System Paths:  {{name: ":", ¬
     origin: shell relative}}} }
```

For more information on the meaning of the properties listed in Table 12.18 and Table 12.19, see “Access Paths” on page 280.
Target Settings

You set the current target parameters in the Target Settings options panel with the properties shown in Table 12.20.

Table 12.20 Target Settings Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Name</td>
<td>string</td>
</tr>
<tr>
<td>Linker</td>
<td>string</td>
</tr>
<tr>
<td>Post Linker</td>
<td>string</td>
</tr>
<tr>
<td>Output Directory</td>
<td>string</td>
</tr>
<tr>
<td>Pre Linker</td>
<td>string</td>
</tr>
</tbody>
</table>

The Target Name string is the name of the target (you choose this name). The Linker string must be the name of one of the files in the Linkers folder of the CodeWarrior Plugins folder. The Post Linker string must be the name of one of the files in the Post Linkers folder of the CodeWarrior Plugins folder. The Output Directory Path string is a string that points to a location on your hard disk where the output files should be placed after linking. You can make this path absolute, or you can make it relative to the location of the project (project relative), compiler (compiler relative), or system (system relative).

The following is a list of the names of the linkers included with CW:

- MacOS 68K Linker
- MacOS PPC Linker
- MacOS Merge
For example, this statement changes the current target to Macintosh 68K.

```
Set Preferences of panel "Target Settings" to ¬
{Linker: "MacOS 68K Linker"}
```

### File Mapping Information

The File Mapping Information is a list of all the types of files you can include in the current project. It contains records described in Table 12.21.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Type</td>
<td>string</td>
</tr>
<tr>
<td>Extension</td>
<td>string</td>
</tr>
<tr>
<td>Precompiled</td>
<td>boolean</td>
</tr>
<tr>
<td>Resource File</td>
<td>boolean</td>
</tr>
<tr>
<td>Launchable</td>
<td>boolean</td>
</tr>
<tr>
<td>Ignored by Make</td>
<td>boolean</td>
</tr>
<tr>
<td>Compiler</td>
<td>string</td>
</tr>
</tbody>
</table>

The Compiler string must be the name of one of the files in the Compilers folder of the CodeWarrior Plugins folder. These names are different from the names that appear in the Compiler pop-up menu of the Target preference panel. **Table 12.22** shows you which name to use for the compilers included with CW Gold.
### Choosing a compiler

<table>
<thead>
<tr>
<th>To target...</th>
<th>with...</th>
<th>specify this string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>68K Macintosh</td>
<td>Metrowerks C/C++</td>
<td>&quot;MW C/C++ 68K&quot;</td>
</tr>
<tr>
<td></td>
<td>Metrowerks Pascal</td>
<td>&quot;MW Pascal 68K&quot;</td>
</tr>
<tr>
<td></td>
<td>Rez</td>
<td>&quot;Rez&quot;</td>
</tr>
<tr>
<td></td>
<td>Library Importer</td>
<td>&quot;Lib Import 68K&quot;</td>
</tr>
<tr>
<td></td>
<td>MPW .o Importer</td>
<td>&quot;MPW Import 68K&quot;</td>
</tr>
<tr>
<td></td>
<td>PEF Importer</td>
<td>&quot;PEF Import 68K&quot;</td>
</tr>
<tr>
<td>Power</td>
<td>Metrowerks C/C++</td>
<td>&quot;MW C/C++ PPC&quot;</td>
</tr>
<tr>
<td>Macintosh</td>
<td>Metrowerks Pascal</td>
<td>&quot;MW Pascal PPC&quot;</td>
</tr>
<tr>
<td></td>
<td>Rez</td>
<td>&quot;Rez&quot;</td>
</tr>
<tr>
<td></td>
<td>Library Importer</td>
<td>&quot;Lib Import PPC&quot;</td>
</tr>
<tr>
<td></td>
<td>PEF Importer</td>
<td>&quot;PEF Import PPC&quot;</td>
</tr>
<tr>
<td></td>
<td>XCOFF Importer</td>
<td>&quot;XCOFF Import PPC&quot;</td>
</tr>
<tr>
<td>Win32/x86</td>
<td>Metrowerks C/C++</td>
<td>&quot;MW C/C++ x86&quot;</td>
</tr>
<tr>
<td></td>
<td>Resource Compiler</td>
<td>&quot;MW WinRC&quot;</td>
</tr>
<tr>
<td></td>
<td>Resource Importer</td>
<td>&quot;WinRes Import&quot;</td>
</tr>
<tr>
<td></td>
<td>x86 Lib Importer</td>
<td>&quot;Lib Import x86&quot;</td>
</tr>
<tr>
<td></td>
<td>x86 Obj Import</td>
<td>&quot;Obj Import x86&quot;</td>
</tr>
<tr>
<td>Java</td>
<td>Java</td>
<td>&quot;MW Java&quot;</td>
</tr>
</tbody>
</table>

To specify that a file isn’t compiled, use the empty string "" for the compiler. For example, these statements show how to add an entry for text files that end in .txt and are not compiled.

```plaintext
set currPrefs to Get Preferences from panel "Target"
set Mappings of currPrefs to ¬
    Mappings of currPrefs & ¬
    {{File Type:"TEXT", Extension:".txt", ¬
```
Set Preferences of panel "Target" to currPrefs

### Segment

The Segment Class properties, as shown in Table 12.23, contain information about a segment or group in the open project.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>filecount</td>
<td>short integer</td>
</tr>
<tr>
<td>(read only)</td>
<td></td>
</tr>
<tr>
<td>preloaded</td>
<td>boolean</td>
</tr>
<tr>
<td>(68K only)</td>
<td></td>
</tr>
<tr>
<td>protected</td>
<td>boolean</td>
</tr>
<tr>
<td>(68K only)</td>
<td></td>
</tr>
<tr>
<td>locked</td>
<td>boolean</td>
</tr>
<tr>
<td>(68K only)</td>
<td></td>
</tr>
<tr>
<td>purgeable</td>
<td>boolean</td>
</tr>
<tr>
<td>(68K only)</td>
<td></td>
</tr>
<tr>
<td>system heap</td>
<td>boolean</td>
</tr>
<tr>
<td>(68K only)</td>
<td></td>
</tr>
</tbody>
</table>

### Project File

The Project File Class contains information about an entry in a project file. Table 12.24 illustrates the available properties.
### Project File Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>filetype (read only)</td>
<td>· source</td>
</tr>
<tr>
<td></td>
<td>· unknown</td>
</tr>
<tr>
<td>name (read only)</td>
<td>string</td>
</tr>
<tr>
<td>disk file (read only)</td>
<td>file specification</td>
</tr>
<tr>
<td>codesize (read only)</td>
<td>long integer</td>
</tr>
<tr>
<td>datasize (read only)</td>
<td>long integer</td>
</tr>
<tr>
<td>up to date (read only)</td>
<td>boolean</td>
</tr>
<tr>
<td>symbols</td>
<td>boolean</td>
</tr>
<tr>
<td>initialize before</td>
<td>boolean</td>
</tr>
<tr>
<td>includes (read only)</td>
<td>file specification</td>
</tr>
<tr>
<td>weak link (PPC only)</td>
<td>boolean</td>
</tr>
</tbody>
</table>

### Compiler Classes

- [C/C++ Compiler](#)
- [Java Compiler](#)
- [Pascal Compiler](#)
- [Rez Resource Compiler](#)
- [Windows Resource Compiler](#)

**C/C++ Compiler**

*Table 12.25* lists the Metrowerks C/C++ Class properties.

**NOTE:** In AppleScript, you must refer to the C/C++ Language preference panel as panel "C/C++ Compiler".
<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix File</td>
<td>string</td>
</tr>
<tr>
<td>Activate CPlusPlus</td>
<td>boolean</td>
</tr>
<tr>
<td>ARM Conformance</td>
<td>boolean</td>
</tr>
<tr>
<td>ANSI Keywords Only</td>
<td>boolean</td>
</tr>
<tr>
<td>Require Function Prototypes</td>
<td>boolean</td>
</tr>
<tr>
<td>Expand Trigraph Sequences</td>
<td>boolean</td>
</tr>
<tr>
<td>Enums Always Ints</td>
<td>boolean</td>
</tr>
<tr>
<td>MPW Pointer Type Rules</td>
<td>boolean</td>
</tr>
<tr>
<td>Exception Handling</td>
<td>boolean</td>
</tr>
<tr>
<td>Autoinlining</td>
<td>boolean</td>
</tr>
<tr>
<td>Pool Strings</td>
<td>boolean</td>
</tr>
<tr>
<td>Dont Reuse Strings</td>
<td>boolean</td>
</tr>
<tr>
<td>ANSI Strict</td>
<td>boolean</td>
</tr>
<tr>
<td>MPW Newlines</td>
<td>boolean</td>
</tr>
<tr>
<td>RTTI</td>
<td>boolean</td>
</tr>
<tr>
<td>Multibyte Aware</td>
<td>boolean</td>
</tr>
<tr>
<td>Use Unsigned Chars</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Table 12.26 lists the Metrowerks C/C++ Warnings Class properties.

Table 12.26  

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inlining</td>
<td>inline_none</td>
</tr>
<tr>
<td></td>
<td>inline_smart</td>
</tr>
<tr>
<td></td>
<td>inlinedepth_1</td>
</tr>
<tr>
<td></td>
<td>inlinedepth_2</td>
</tr>
<tr>
<td></td>
<td>inlinedepth_3</td>
</tr>
<tr>
<td></td>
<td>inlinedepth_4</td>
</tr>
<tr>
<td></td>
<td>inlinedepth_5</td>
</tr>
<tr>
<td></td>
<td>inlinedepth_6</td>
</tr>
<tr>
<td></td>
<td>inlinedepth_7</td>
</tr>
<tr>
<td></td>
<td>inlinedepth_8</td>
</tr>
<tr>
<td></td>
<td>auto</td>
</tr>
<tr>
<td></td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>normal</td>
</tr>
<tr>
<td>Enable bool Support</td>
<td>boolean</td>
</tr>
<tr>
<td>Direct To SOM</td>
<td>SOMoff</td>
</tr>
<tr>
<td></td>
<td>SOMon</td>
</tr>
<tr>
<td></td>
<td>SOMonWithEnv</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused Variables</td>
<td>boolean</td>
</tr>
<tr>
<td>Unused Arguments</td>
<td>boolean</td>
</tr>
<tr>
<td>Illegal Pragmas</td>
<td>boolean</td>
</tr>
<tr>
<td>Empty Declarations</td>
<td>boolean</td>
</tr>
<tr>
<td>Possible Errors</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Mac OS CodeWarrior Scripting  
CodeWarrior IDE AppleScript Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra Commas</td>
<td>boolean</td>
</tr>
<tr>
<td>Extended Error Checking</td>
<td>boolean</td>
</tr>
<tr>
<td>Treat Warnings As Errors</td>
<td>boolean</td>
</tr>
<tr>
<td>Hidden Virtual Functions</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Java Compiler

Table 12.27 lists the Metrowerks Java Compiler Class properties.

Table 12.27 Java Compiler Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Inlining</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Pascal Compiler

Table 12.28 lists the Pascal Language Class options. (In AppleScript, you must refer to the Pascal Language preference panel as panel "Pascal Compiler").

Table 12.28 Pascal Compiler Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate Range Checking</td>
<td>boolean</td>
</tr>
<tr>
<td>Use Propagation</td>
<td>boolean</td>
</tr>
<tr>
<td>Activate Overflow Checking</td>
<td>boolean</td>
</tr>
<tr>
<td>Case Sensitive</td>
<td>boolean</td>
</tr>
<tr>
<td>ANS Conformance</td>
<td>boolean</td>
</tr>
<tr>
<td>Activate ObjectPascal</td>
<td>boolean</td>
</tr>
<tr>
<td>Strings copy using length byte</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Table 12.29 lists the Pascal Warning Class properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pool Strings</td>
<td>boolean</td>
</tr>
<tr>
<td>Dont Reuse Strings</td>
<td>boolean</td>
</tr>
<tr>
<td>Pool Sets</td>
<td>boolean</td>
</tr>
<tr>
<td>Dont Reuse Sets</td>
<td>boolean</td>
</tr>
<tr>
<td>Prefix File</td>
<td>string</td>
</tr>
<tr>
<td>Relax Pointer Compatibility</td>
<td>boolean</td>
</tr>
<tr>
<td>Optimize class hierarchy</td>
<td>boolean</td>
</tr>
<tr>
<td>Pointer based objects</td>
<td>boolean</td>
</tr>
<tr>
<td>Expand method tables</td>
<td>boolean</td>
</tr>
<tr>
<td>Inline method dispatching</td>
<td>boolean</td>
</tr>
<tr>
<td>Activate NilChecking</td>
<td>boolean</td>
</tr>
<tr>
<td>Trap Unmatched Cases</td>
<td>boolean</td>
</tr>
<tr>
<td>Copy Value Parameter</td>
<td>boolean</td>
</tr>
<tr>
<td>Turbo Pascal IO</td>
<td>small integer</td>
</tr>
</tbody>
</table>

Table 12.29  Pascal Warnings Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified ForLoop Indexes</td>
<td>boolean</td>
</tr>
<tr>
<td>Function Returns</td>
<td>boolean</td>
</tr>
<tr>
<td>Undefined Routines</td>
<td>boolean</td>
</tr>
<tr>
<td>GotoAndLabels</td>
<td>boolean</td>
</tr>
<tr>
<td>BranchingIntoWith</td>
<td>boolean</td>
</tr>
<tr>
<td>BranchingIntoFor</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Mac OS CodeWarrior Scripting
CodeWarrior IDE AppleScript Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BranchingBetweenCase</td>
<td>boolean</td>
</tr>
<tr>
<td>BranchingBetweenIfAndElse</td>
<td>boolean</td>
</tr>
<tr>
<td>Unused Variables</td>
<td>boolean</td>
</tr>
<tr>
<td>Unused Arguments</td>
<td>boolean</td>
</tr>
<tr>
<td>Check string param sizes</td>
<td>boolean</td>
</tr>
</tbody>
</table>

**Rez Resource Compiler**

Table 12.30 lists the properties for the Metrowerks Rez Compiler Class.

**Table 12.30 Rez Compiler Class**

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redeclared Types</td>
<td>boolean</td>
</tr>
<tr>
<td>RezPrefix File</td>
<td>string</td>
</tr>
<tr>
<td>Escape Control Chars</td>
<td>boolean</td>
</tr>
<tr>
<td>Max width</td>
<td>small integer</td>
</tr>
<tr>
<td>Filter Mode</td>
<td>Skip, or Only</td>
</tr>
<tr>
<td>Filtered Types</td>
<td>string</td>
</tr>
<tr>
<td>Alignment</td>
<td>small integer</td>
</tr>
<tr>
<td>Script Mode</td>
<td>Roman, Japanese, Korean, SimpChinese, or TradChinese</td>
</tr>
</tbody>
</table>

**Windows Resource Compiler**

Table 12.31 lists the properties for the Windows Resource Compiler Class.
Table 12.31 Windows Resource Compiler Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix File</td>
<td>string</td>
</tr>
</tbody>
</table>

CodeGen Classes

- 68K CodeGen
- PPC CodeGen
- Win32/x86 CodeGen

68K CodeGen

Table 12.32 lists the 68K Processor Class properties. In AppleScript, you must refer to the 68K Processor preference panel as panel "68K CodeGen".

Table 12.32 68K CodeGen Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Struct Alignment</td>
<td>· mc68k</td>
</tr>
<tr>
<td></td>
<td>· mc68k4</td>
</tr>
<tr>
<td></td>
<td>· PowerPC</td>
</tr>
<tr>
<td>Peephole Optimizer</td>
<td>boolean</td>
</tr>
<tr>
<td>CSE Optimizer</td>
<td>boolean</td>
</tr>
<tr>
<td>Optimize For Size</td>
<td>boolean</td>
</tr>
<tr>
<td>Use Profiler</td>
<td>boolean</td>
</tr>
<tr>
<td>Code Model</td>
<td>· small</td>
</tr>
<tr>
<td></td>
<td>· smart</td>
</tr>
<tr>
<td></td>
<td>· large</td>
</tr>
<tr>
<td>MC68020 CodeGen</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Table 12.33 lists the PPC Processor preference panel Class properties. In AppleScript, you must refer to the PPC Processor preference panel as panel "PPC CodeGen".

### PPC CodeGen

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating Point CodeGen</td>
<td>· SANE</td>
</tr>
<tr>
<td></td>
<td>· MC6881</td>
</tr>
<tr>
<td></td>
<td>· Library</td>
</tr>
<tr>
<td></td>
<td>· PalmOS</td>
</tr>
<tr>
<td>Far Method Tables</td>
<td>boolean</td>
</tr>
<tr>
<td>Far String Constants</td>
<td>boolean</td>
</tr>
<tr>
<td>Four Bytes Ints</td>
<td>boolean</td>
</tr>
<tr>
<td>Eight Byte Double</td>
<td>boolean</td>
</tr>
<tr>
<td>Far Data</td>
<td>boolean</td>
</tr>
<tr>
<td>PC Relative Strings</td>
<td>boolean</td>
</tr>
<tr>
<td>Global Register Allocation</td>
<td>boolean</td>
</tr>
<tr>
<td>MPW Calling Conventions</td>
<td>boolean</td>
</tr>
</tbody>
</table>

### Table 12.33 PPC CodeGen Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Struct Alignment</td>
<td>· mc68k</td>
</tr>
<tr>
<td></td>
<td>· mc68k4</td>
</tr>
<tr>
<td></td>
<td>· PowerPC</td>
</tr>
<tr>
<td>Peephole Optimizer</td>
<td>boolean</td>
</tr>
<tr>
<td>Optimize For Size</td>
<td>boolean</td>
</tr>
<tr>
<td>Use Profiler</td>
<td>boolean</td>
</tr>
<tr>
<td>Name</td>
<td>Property Type</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Make String ReadOnly</td>
<td>boolean</td>
</tr>
<tr>
<td>Instruction Scheduling</td>
<td>off</td>
</tr>
<tr>
<td></td>
<td>· PowerPC601</td>
</tr>
<tr>
<td></td>
<td>· PowerPC603</td>
</tr>
<tr>
<td></td>
<td>· PowerPC604</td>
</tr>
<tr>
<td>Optimization Level</td>
<td>short integer, from 1 to 4</td>
</tr>
<tr>
<td>Global Optimization</td>
<td>boolean</td>
</tr>
<tr>
<td>Store Data in TOC</td>
<td>boolean</td>
</tr>
<tr>
<td>Use FMADD Instructions</td>
<td>boolean</td>
</tr>
<tr>
<td>Traceback Tables</td>
<td>boolean</td>
</tr>
</tbody>
</table>

**IR Optimizer**

*Table 12.34* lists the IR Optimizer Class properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimize Space</td>
<td>boolean</td>
</tr>
<tr>
<td>Optimize Speed</td>
<td>boolean</td>
</tr>
<tr>
<td>Common Subexpressions</td>
<td>boolean</td>
</tr>
<tr>
<td>Loop Invariants</td>
<td>boolean</td>
</tr>
<tr>
<td>Propagation</td>
<td>boolean</td>
</tr>
<tr>
<td>Dead Store Elimination</td>
<td>boolean</td>
</tr>
<tr>
<td>Strength Reduction</td>
<td>boolean</td>
</tr>
<tr>
<td>Dead Code Elimination</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Mac OS CodeWarrior Scripting
CodeWarrior IDE AppleScript Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lifetime Analysis</td>
<td>boolean</td>
</tr>
<tr>
<td>Optimizations Log</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Win32/x86 CodeGen

Table 12.35 lists the x86 CodeGen Class properties.

Table 12.35  Win32/x86 CodeGen Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peephole Optimizer</td>
<td>boolean</td>
</tr>
<tr>
<td>Machine Code Listing</td>
<td>boolean</td>
</tr>
<tr>
<td>Byte Alignment</td>
<td>small integer</td>
</tr>
<tr>
<td>Sym Debug Information</td>
<td>boolean</td>
</tr>
<tr>
<td>CodeView Debug Info</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Disassembler Classes

- 68K Disassembler
- PowerPC Disassembler

68K Disassembler

Table 12.36 lists the properties for 68K Disassembly Class.

Table 12.36  68K Disassembler Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Code</td>
<td>boolean</td>
</tr>
<tr>
<td>Show Source</td>
<td>boolean</td>
</tr>
<tr>
<td>Dont show hex</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Table 12.37 PowerPC Disassembly Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Code</td>
<td>boolean</td>
</tr>
<tr>
<td>Show Source</td>
<td>boolean</td>
</tr>
<tr>
<td>Don't show hex</td>
<td>boolean</td>
</tr>
<tr>
<td>Show Data</td>
<td>boolean</td>
</tr>
<tr>
<td>Show Exceptions</td>
<td>boolean</td>
</tr>
<tr>
<td>Show SYM</td>
<td>boolean</td>
</tr>
<tr>
<td>Show Names</td>
<td>boolean</td>
</tr>
</tbody>
</table>

PowerPC Disassembler

Table 12.37 lists the properties for PowerPC Disassembly Class.

Linker Classes

- 68K Linker
- CFM68K Linker
- Java Linker
- Mac OS Merge Linker
- PowerPC Linker
- PowerPC PEF Linker
Mac OS CodeWarrior Scripting
CodeWarrior IDE AppleScript Classes

- Win32/x86 Linker

68K Linker

Table 12.38 lists the 68K Linker Class properties.

Table 12.38 68K Linker Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate SYM File</td>
<td>boolean</td>
</tr>
<tr>
<td>Full Path In Sym Files</td>
<td>boolean</td>
</tr>
<tr>
<td>Generate Link Map</td>
<td>boolean</td>
</tr>
<tr>
<td>Fast Link</td>
<td>boolean</td>
</tr>
<tr>
<td>Suppress Warnings</td>
<td>boolean</td>
</tr>
<tr>
<td>MacsBug Symbols</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>· oldsymbols</td>
</tr>
<tr>
<td></td>
<td>· newsymbols</td>
</tr>
<tr>
<td>Generate A6 Stack Frames</td>
<td>boolean</td>
</tr>
<tr>
<td>Link Single Segment</td>
<td>boolean</td>
</tr>
<tr>
<td>Merge Compiler Glue</td>
<td>boolean</td>
</tr>
<tr>
<td>Strip Static Init Code</td>
<td>boolean</td>
</tr>
</tbody>
</table>

CFM68K Linker

Table 12.39 lists the CFM68K Class properties.
Table 12.39  CFM68K Linker Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export Symbols</td>
<td></td>
</tr>
<tr>
<td>· none</td>
<td></td>
</tr>
<tr>
<td>· expfile</td>
<td></td>
</tr>
<tr>
<td>· all</td>
<td></td>
</tr>
<tr>
<td>· pragma</td>
<td></td>
</tr>
<tr>
<td>Old Definition</td>
<td>integer</td>
</tr>
<tr>
<td>Old Implementation</td>
<td>integer</td>
</tr>
<tr>
<td>Current Version</td>
<td>integer</td>
</tr>
<tr>
<td>Share Data Section</td>
<td>boolean</td>
</tr>
<tr>
<td>Expand Uninitialized Data</td>
<td>boolean</td>
</tr>
<tr>
<td>Fragment Name</td>
<td>string</td>
</tr>
<tr>
<td>Initialization Name</td>
<td>string</td>
</tr>
<tr>
<td>Main Name</td>
<td>string</td>
</tr>
<tr>
<td>Termination Name</td>
<td>string</td>
</tr>
<tr>
<td>Force Indirect Access</td>
<td>boolean</td>
</tr>
<tr>
<td>Far Data Threshold</td>
<td>Long integer</td>
</tr>
<tr>
<td>Global Data Alignment</td>
<td></td>
</tr>
<tr>
<td>· align1byte</td>
<td></td>
</tr>
<tr>
<td>· align2byte</td>
<td></td>
</tr>
<tr>
<td>· align4byte</td>
<td></td>
</tr>
<tr>
<td>· align8byte</td>
<td></td>
</tr>
<tr>
<td>Library Folder ID</td>
<td>small integer</td>
</tr>
</tbody>
</table>

Java Linker

Table 12.40 lists the Java Linker Class properties.
Table 12.40  Java Linker Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>File Creator</td>
<td>string</td>
<td>Possible values include ‘JAVA’ or ‘MWZP’ as well as the creator types for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Metrowerks Java or ClassWrangler.</td>
</tr>
<tr>
<td>File Type</td>
<td>string</td>
<td>Possible values include ‘ZIP’</td>
</tr>
<tr>
<td>Output Type</td>
<td></td>
<td>· zip file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· runnable zip file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· droplet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· folder</td>
</tr>
<tr>
<td>Compress Zip</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Mac OS Merge Linker

Table 12.41 lists the Mac OS Merge Linker Class properties.

Table 12.41  Mac OS Merge Linker Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Type</td>
<td>constant</td>
<td>The type of the project.</td>
</tr>
<tr>
<td>File Name</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>File Creator</td>
<td>string</td>
<td>The creator type of the finished binary.</td>
</tr>
</tbody>
</table>
Table 12.42 lists the PPC Linker Class properties.

### Table 12.42 PPC Linker Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Type</td>
<td>string</td>
</tr>
<tr>
<td>Suppress Warnings</td>
<td>boolean</td>
</tr>
<tr>
<td>Copy Fragments</td>
<td>boolean</td>
</tr>
<tr>
<td>Copy Resources</td>
<td>boolean</td>
</tr>
<tr>
<td>Skip Resource Types</td>
<td>string</td>
</tr>
<tr>
<td>Generate SYM File</td>
<td>boolean</td>
</tr>
<tr>
<td>Full Path In Sym Files</td>
<td>boolean</td>
</tr>
<tr>
<td>Generate Link Map</td>
<td>boolean</td>
</tr>
<tr>
<td>Link Mode</td>
<td>fast, normal, slow</td>
</tr>
<tr>
<td>Suppress Warnings</td>
<td>boolean</td>
</tr>
<tr>
<td>Initialization Name</td>
<td>string</td>
</tr>
<tr>
<td>Main Name</td>
<td>string</td>
</tr>
<tr>
<td>Termination Name</td>
<td>string</td>
</tr>
<tr>
<td>Strip Static Init Code</td>
<td>boolean</td>
</tr>
<tr>
<td>Duplicate Item Warning</td>
<td>boolean</td>
</tr>
</tbody>
</table>
PowerPC PEF Linker

Table 12.43 lists the PPC PEF Class properties.

Table 12.43 PPC PEF Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export Symbols</td>
<td>· none</td>
</tr>
<tr>
<td></td>
<td>· expfile</td>
</tr>
<tr>
<td></td>
<td>· all</td>
</tr>
<tr>
<td></td>
<td>· pragma</td>
</tr>
<tr>
<td>Old Definition</td>
<td>integer</td>
</tr>
<tr>
<td>Old Implementation</td>
<td>integer</td>
</tr>
<tr>
<td>Current Version</td>
<td>integer</td>
</tr>
<tr>
<td>Code Sorting</td>
<td>· nosort</td>
</tr>
<tr>
<td></td>
<td>· pragmas</td>
</tr>
<tr>
<td></td>
<td>· depth</td>
</tr>
<tr>
<td></td>
<td>· breadth</td>
</tr>
<tr>
<td></td>
<td>· sortfile</td>
</tr>
<tr>
<td>Share Data Section</td>
<td>boolean</td>
</tr>
<tr>
<td>Expand Uninitialized Data</td>
<td>boolean</td>
</tr>
<tr>
<td>Fragment Name</td>
<td>string</td>
</tr>
<tr>
<td>Library Folder ID</td>
<td>short integer</td>
</tr>
</tbody>
</table>

Win32/x86 Linker

Table 12.44 lists the x86 Linker Class properties.
Table 12.44  x86 Linker Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate SYM File</td>
<td>boolean</td>
</tr>
<tr>
<td>Entry Point Usage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• none</td>
</tr>
<tr>
<td></td>
<td>• default</td>
</tr>
<tr>
<td></td>
<td>• user specified</td>
</tr>
<tr>
<td>Entry Point</td>
<td>string</td>
</tr>
<tr>
<td>SubSystem</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• unknown</td>
</tr>
<tr>
<td></td>
<td>• native</td>
</tr>
<tr>
<td></td>
<td>• Windows GUI</td>
</tr>
<tr>
<td></td>
<td>• Windows CUI</td>
</tr>
<tr>
<td>SubSystem Major Id</td>
<td>small integer</td>
</tr>
<tr>
<td>SubSystem Minor Id</td>
<td>small integer</td>
</tr>
<tr>
<td>User Major Id</td>
<td>small integer</td>
</tr>
<tr>
<td>User Minor Id</td>
<td>small integer</td>
</tr>
<tr>
<td>Generate Link Map</td>
<td>boolean</td>
</tr>
<tr>
<td>Generate CV Info</td>
<td>boolean</td>
</tr>
<tr>
<td>Command Line File</td>
<td>string</td>
</tr>
</tbody>
</table>

**Build Classes**

- [Build Extras](#)
- [Error Information](#)

**Build Extras**

*Table 12.45* describes the properties for the Build Extras Class.
Table 12.45  Build Extras Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser active</td>
<td>boolean</td>
</tr>
<tr>
<td>Modification date caching</td>
<td>boolean</td>
</tr>
<tr>
<td>Dump Browser Info (read-only)</td>
<td>boolean</td>
</tr>
<tr>
<td>Cache Subproject Data (read-only)</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Error Information

This class describes a single error or warning from the compiler or the linker. This class is used by all compilers for all processors. This class properties are listed in Table 12.46.

Table 12.46  Error Information Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>messageKind (read only)</td>
<td>· information</td>
</tr>
<tr>
<td></td>
<td>· compiler error</td>
</tr>
<tr>
<td></td>
<td>· compiler warning</td>
</tr>
<tr>
<td></td>
<td>· definition</td>
</tr>
<tr>
<td></td>
<td>· linker error</td>
</tr>
<tr>
<td></td>
<td>· linker warning</td>
</tr>
<tr>
<td></td>
<td>· find result</td>
</tr>
<tr>
<td></td>
<td>· generic error</td>
</tr>
<tr>
<td>message (read only)</td>
<td>string</td>
</tr>
<tr>
<td>disk file (read only)</td>
<td>file specification</td>
</tr>
<tr>
<td>line Number (read only)</td>
<td>long integer</td>
</tr>
</tbody>
</table>
Browser Classes

- **Browser Coloring**
- **Browser Catalog**
- **Function Information**

Browser Coloring

Table 12.47 lists the Browser preference panel properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser Keywords</td>
<td>boolean</td>
</tr>
<tr>
<td>Classes Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Constants Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Enums Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Functions Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Globals Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Macros Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Templates Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Typedefs Color</td>
<td>RGB values list</td>
</tr>
</tbody>
</table>

Browser Catalog

The Browser Catalog Class elements may be referred to by numeric index, and by name.

Function Information

The Function Information Class properties are described in Table 12.48.
Table 12.48  Function Information Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>disk file</td>
<td>file specification</td>
</tr>
<tr>
<td>(read only)</td>
<td></td>
</tr>
<tr>
<td>lineNumber</td>
<td>integer</td>
</tr>
<tr>
<td>(read only)</td>
<td></td>
</tr>
</tbody>
</table>

Editor Classes

- Editor
- Font
- Document
- Character
- Insertion Point
- Custom Keywords
- Line
- Text
- Selection-Object
- Syntax Coloring
- Window

Editor

Table 12.49 lists the Editor Class properties.

Table 12.49  Editor Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remember window</td>
<td>boolean</td>
</tr>
<tr>
<td>Main Text Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Background Color</td>
<td>RGB values list</td>
</tr>
<tr>
<td>Context Popup Delay</td>
<td>boolean</td>
</tr>
</tbody>
</table>
An RGB values list is a list of three numbers, from 0 to 65,535, specifying how much red, green, and blue a color contains. For example, this example code sets the main text color to red.

```applescript
set Prefs to Get Preferences from panel "Editor"
set Main Text Color of Prefs to {65535,0,0}
Set Preferences of panel "Editor" to Prefs
```

### Font

Table 12.50 lists the Font Class properties.

#### Table 12.50 Font Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Indent</td>
<td>boolean</td>
</tr>
<tr>
<td>Tab size</td>
<td>short integer</td>
</tr>
<tr>
<td>Text font</td>
<td>string</td>
</tr>
<tr>
<td>Text size</td>
<td>short integer</td>
</tr>
</tbody>
</table>
Document

This class, shown in Table 12.51, contains class properties about a text file opened with the CodeWarrior Editor. The plural form for this class would be Documents.

The elements for a document are:

- character by numeric index, before/after another element, as a range of elements, or satisfying a test
- insertion point before/after another element
- line by numeric index, as a range of elements, before/after another element
- text as a range of elements

Table 12.51 Document Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (read only)</td>
<td>string</td>
</tr>
<tr>
<td>kind</td>
<td>· project</td>
</tr>
<tr>
<td></td>
<td>· editor</td>
</tr>
<tr>
<td></td>
<td>· message</td>
</tr>
<tr>
<td></td>
<td>· file compare</td>
</tr>
<tr>
<td></td>
<td>· catalog document</td>
</tr>
<tr>
<td></td>
<td>· class browser</td>
</tr>
<tr>
<td></td>
<td>· single class browser</td>
</tr>
<tr>
<td></td>
<td>· symbol browser</td>
</tr>
<tr>
<td></td>
<td>· class hierarchy</td>
</tr>
<tr>
<td></td>
<td>· single class hierarchy</td>
</tr>
<tr>
<td></td>
<td>· project inspector</td>
</tr>
<tr>
<td></td>
<td>· ToolServer worksheet</td>
</tr>
<tr>
<td></td>
<td>· build progress document</td>
</tr>
</tbody>
</table>
### Character Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>file permissions</td>
<td>• read write</td>
</tr>
<tr>
<td></td>
<td>• read only</td>
</tr>
<tr>
<td></td>
<td>• checked out read write</td>
</tr>
<tr>
<td></td>
<td>• checked out read only</td>
</tr>
<tr>
<td></td>
<td>• checked out read modify</td>
</tr>
<tr>
<td></td>
<td>• locked</td>
</tr>
<tr>
<td></td>
<td>• none</td>
</tr>
<tr>
<td>location (read only)</td>
<td>file specification</td>
</tr>
<tr>
<td>index (read only)</td>
<td>long integer</td>
</tr>
<tr>
<td>window (read only)</td>
<td>window</td>
</tr>
</tbody>
</table>

### Insertion Point

Table 12.53 describes the Insertion Point Class properties.
### Table 12.53 Insertion Point Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>length (read only)</td>
<td>integer</td>
</tr>
<tr>
<td>offset (read only)</td>
<td>integer</td>
</tr>
</tbody>
</table>

### Custom Keywords

Table 12.54 describes the Custom Keywords Class properties.

### Table 12.54 Custom Keywords Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom color 1</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>Custom color 2</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>Custom color 3</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>Custom color 4</td>
<td>RGB color values list</td>
</tr>
</tbody>
</table>

### Line

Table 12.55 describes the properties for the Line Class. The plural form of the class should be referred to as Lines. This class has elements that may be described as follows:

- character by numeric index, as a range of elements, and before/after another element

### Table 12.55 Line Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>index (read only)</td>
<td>integer</td>
</tr>
<tr>
<td>offset (read only)</td>
<td>integer</td>
</tr>
<tr>
<td>length (read only)</td>
<td>integer</td>
</tr>
</tbody>
</table>
Text

Table 12.56 describes the Text Class properties. The Text Class has the following elements:

- **character** by numeric index, before/after another element, as a range of elements
- **insertion point** before/after another element
- **line** by numeric index, as a range of elements, before/after another element
- **text** as a range of elements

Table 12.56 Text Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>offset (read only)</td>
<td>integer</td>
</tr>
<tr>
<td>length (read only)</td>
<td>integer</td>
</tr>
</tbody>
</table>

Selection-Object

Table 12.57 describes the Selection-Object Class properties. The elements of this object are:

- **character** by numeric index, before/after another element, as a range of elements, or satisfying a test
- **line** by numeric index, as a range of elements, or before/after another element
- **text** as a range of elements

Table 12.57 Selection-Object Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>contents</td>
<td>type class</td>
</tr>
<tr>
<td>length (read only)</td>
<td>short integer</td>
</tr>
<tr>
<td>offset (read only)</td>
<td>short integer</td>
</tr>
</tbody>
</table>
Syntax Coloring

Table 12.59 describes the Syntax Coloring Class properties.

### Table 12.58 Syntax Coloring Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax coloring</td>
<td>boolean</td>
</tr>
<tr>
<td>Comment color</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>Keyword color</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>String color</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>Custom color 1</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>Custom color 2</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>Custom color 3</td>
<td>RGB color values list</td>
</tr>
<tr>
<td>Custom color 4</td>
<td>RGB color values list</td>
</tr>
</tbody>
</table>

Window

Table 12.59 describes the Window Class properties. The plural form of this object is Windows.

### Table 12.59 Window Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>index</td>
<td>short integer</td>
</tr>
<tr>
<td>bounds</td>
<td>bounding rectangle</td>
</tr>
<tr>
<td>document (read only)</td>
<td>document</td>
</tr>
<tr>
<td>position (read only)</td>
<td>point</td>
</tr>
<tr>
<td>visible (read only)</td>
<td>boolean</td>
</tr>
<tr>
<td>zoomed</td>
<td>boolean</td>
</tr>
</tbody>
</table>
Object Classes

The object classes describe the properties of the objects in the project.

- **Member Function Class**
- **Base Class**
- **Class Class**
- **Data Member Class**

### Member Function Class

Table 12.60 lists the Member Function AppleScript class properties. The plural reference to use would be Member Functions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (read only)</td>
<td>string</td>
</tr>
<tr>
<td>access (read only)</td>
<td>public</td>
</tr>
<tr>
<td></td>
<td>protected</td>
</tr>
<tr>
<td></td>
<td>private</td>
</tr>
<tr>
<td>virtual (read only)</td>
<td>boolean</td>
</tr>
<tr>
<td>static (read only)</td>
<td>boolean</td>
</tr>
<tr>
<td>declaration file (read only)</td>
<td>file specification</td>
</tr>
<tr>
<td>declaration start offset (read only)</td>
<td>long integer</td>
</tr>
<tr>
<td>declaration end offset (read only)</td>
<td>small integer</td>
</tr>
<tr>
<td>implementation file (read only)</td>
<td>file specification</td>
</tr>
</tbody>
</table>
Table 12.61 lists the Base Class AppleScript class properties. The plural reference to use would be Base Classes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>implementation end offset (read only)</td>
<td>small integer</td>
</tr>
<tr>
<td>implementation start offset (read only)</td>
<td>small integer</td>
</tr>
</tbody>
</table>

**Base Class**

Table 12.61 lists the Base Class AppleScript class properties. The plural reference to use would be Base Classes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>class (read only)</td>
<td>reference</td>
</tr>
<tr>
<td>access (read only)</td>
<td>public</td>
</tr>
<tr>
<td></td>
<td>protected</td>
</tr>
<tr>
<td></td>
<td>private</td>
</tr>
<tr>
<td>virtual (read only)</td>
<td>boolean</td>
</tr>
</tbody>
</table>

**Class Class**

Table 12.62 lists the Class AppleScript class properties. The plural reference to use would be Classes. The elements of this class include:

- base class by numeric index
- member function by numeric index, and by name
- data member by numeric index, and by name
Table 12.62 Class AppleScript Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (read only)</td>
<td>string</td>
</tr>
<tr>
<td>language (read only)</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C++</td>
</tr>
<tr>
<td></td>
<td>Pascal</td>
</tr>
<tr>
<td></td>
<td>Object Pascal</td>
</tr>
<tr>
<td></td>
<td>Java</td>
</tr>
<tr>
<td></td>
<td>Assembler</td>
</tr>
<tr>
<td></td>
<td>Unknown</td>
</tr>
<tr>
<td>declaration file (read only)</td>
<td>file specification</td>
</tr>
<tr>
<td>declaration start offset (read only)</td>
<td>small integer</td>
</tr>
<tr>
<td>declaration end offset (read only)</td>
<td>small integer</td>
</tr>
<tr>
<td>subclasses (read only)</td>
<td>list of class</td>
</tr>
<tr>
<td>all subclasses (read only)</td>
<td>list of class</td>
</tr>
</tbody>
</table>

Data Member Class

Table 12.63 lists the Data Member AppleScript class properties. The plural reference to use would be Data Members.

Table 12.63 Data Member AppleScript Class Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (read only)</td>
<td>string</td>
</tr>
<tr>
<td>access (read only)</td>
<td>public</td>
</tr>
<tr>
<td></td>
<td>private</td>
</tr>
<tr>
<td></td>
<td>protected</td>
</tr>
</tbody>
</table>
### Misc Classes

The Miscellaneous classes allow configuration of Version Control Systems, and Extras for the project settings.

- Extras
- Target
- Target File
- Text Document
- Version Control System Setup

Other classes are used for inheritance functions:

- Application
- Build Progress Document
- Catalog Document
- Class Browser
- Class Hierarchy
- Editor Document
- File
- File Compare Document
- Message Document
- Project Document
- Project Inspector
- Single Class Browser
- Single Class Hierarchy
• Editor Document
• Symbol Browser
• ToolServer Worksheet

Extras

Table 12.64 lists the Extras class properties.

Table 12.64 Extras Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completion sound</td>
<td>boolean</td>
</tr>
<tr>
<td>Success sound</td>
<td>string</td>
</tr>
<tr>
<td>Failure sound</td>
<td>string</td>
</tr>
<tr>
<td>Use Script Menu</td>
<td>boolean</td>
</tr>
<tr>
<td>Use Editor Extensions</td>
<td>boolean</td>
</tr>
<tr>
<td>Use External Editor</td>
<td>boolean</td>
</tr>
<tr>
<td>Honor Projector State for</td>
<td>boolean</td>
</tr>
<tr>
<td>Projects</td>
<td></td>
</tr>
</tbody>
</table>

Target

Table 12.65 lists the properties for the Target class. The plural form of Target is Targets. This class inherits all properties and elements of the given class.

Table 12.65 Target Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>index (read-only)</td>
<td>integer</td>
</tr>
<tr>
<td>project document (read-only)</td>
<td>project document</td>
</tr>
</tbody>
</table>
Target File

Table 12.66 lists the properties for the Target File class. The plural form of Target File is Target Files.

Table 12.66 Target File Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (read-only)</td>
<td>integer</td>
</tr>
<tr>
<td>type (read-only)</td>
<td>• library file</td>
</tr>
<tr>
<td></td>
<td>• project file</td>
</tr>
<tr>
<td></td>
<td>• resource file</td>
</tr>
<tr>
<td></td>
<td>• text file</td>
</tr>
<tr>
<td></td>
<td>• unknown file</td>
</tr>
<tr>
<td>index (read-only)</td>
<td>integer</td>
</tr>
<tr>
<td>location (read-only)</td>
<td>file specification</td>
</tr>
<tr>
<td>path (read-only)</td>
<td>string</td>
</tr>
<tr>
<td>linked (read-only)</td>
<td>boolean</td>
</tr>
<tr>
<td>link index (read-only)</td>
<td>integer</td>
</tr>
<tr>
<td>modified date (read-only)</td>
<td>date</td>
</tr>
<tr>
<td>compiled date (read-only)</td>
<td>date</td>
</tr>
<tr>
<td>code size (read-only)</td>
<td>integer</td>
</tr>
<tr>
<td>data size (read-only)</td>
<td>integer</td>
</tr>
<tr>
<td>debug</td>
<td>boolean</td>
</tr>
<tr>
<td>weak link (read-only)</td>
<td>boolean</td>
</tr>
<tr>
<td>init before</td>
<td>boolean</td>
</tr>
<tr>
<td>prerequisites (read-only)</td>
<td>list of list</td>
</tr>
<tr>
<td>dependents (read-only)</td>
<td>list</td>
</tr>
</tbody>
</table>
Text Document

Table 12.67 lists the properties for the Text Document class. The plural form of Text Document is Text Documents.

**Table 12.67 Text Document Class**

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>inherits (read-only)</td>
<td>document</td>
</tr>
<tr>
<td>modified (read-only)</td>
<td>boolean</td>
</tr>
<tr>
<td>selection</td>
<td>selection-object</td>
</tr>
</tbody>
</table>

Version Control System Setup

Table 12.68 lists the Version Control System class properties.

**Table 12.68 Version Control System Class**

<table>
<thead>
<tr>
<th>Name</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCS Active</td>
<td>boolean</td>
</tr>
<tr>
<td>Connection Method</td>
<td>string</td>
</tr>
<tr>
<td>Username</td>
<td>string</td>
</tr>
<tr>
<td>Password</td>
<td>string</td>
</tr>
<tr>
<td>Auto Connect</td>
<td>boolean</td>
</tr>
<tr>
<td>Store Password</td>
<td>boolean</td>
</tr>
<tr>
<td>Always Prompt</td>
<td>boolean</td>
</tr>
<tr>
<td>Mount Volume</td>
<td>boolean</td>
</tr>
<tr>
<td>Database Path</td>
<td>path information</td>
</tr>
<tr>
<td>Local Root</td>
<td>path information</td>
</tr>
</tbody>
</table>
Application

The Application class elements are:

- `document` by numeric index, by name, and as a range of elements
- `window` by numeric index, by name, and as a range of elements
- `class` by numeric index, or by name

Build Progress Document

The plural form of Build Progress Document is Build Progress Documents. This class inherits all properties and elements of the given class.

Catalog Document

The plural form of Catalog Document is Catalog Documents. This class inherits all properties and elements of the given class.

Class Browser

The plural form of Class Browser is Class Browsers. This class inherits all properties and elements of the given class.

Class Hierarchy

The plural form of Class Hierarchy is Class Hierarchies. This class inherits all properties and elements of the given class.

Editor Document

The plural form of Editor Document is Editor Documents. This class inherits all properties and elements of the given class.

File

The File Class plural to use in AppleScripts is Files.
File Compare Document

The plural form of File Compare Document is File Compare Documents. This class inherits all properties and elements of the given class.

Message Document

The plural form of Message Document is Message Documents. This class inherits all properties and elements of the given class.

Project Document

The plural form of Project Document is Project Documents. This class inherits all properties and elements of the given class.

Project Inspector

The plural form of Project Inspector is Project Inspectors. This class inherits all properties and elements of the given class.

Single Class Browser

The plural form of Single Class Browser is Single Class Browsers. This class inherits all properties and elements of the given class.

Single Class Hierarchy

The plural form of Single Class Hierarchy is Single Class Hierarchies. This class inherits all properties and elements of the given class.

Symbol Browser

The plural form of Symbol Browser is Symbol Browsers. This class inherits all properties and elements of the given class.

ToolServer Worksheet

The plural form of ToolServer Worksheet is ToolServer Worksheets. This class inherits all properties and elements of the given class.
Coding with CodeWarrior IDE and Apple Events

You may want to use low-level Apple Events instead of writing AppleScripts if you are producing tools or programs that need to control the CodeWarrior IDE while they are running. Third-party editors or browsers, and other tools, might require this capability.

For documentation on using low-level Apple Events in your program code, refer to *Inside Macintosh: Interapplication Communication* (Addison-Wesley) for a discussion of how to use the Apple Events portion of the Mac OS Toolbox.

There is some example code available that shows how to send Apple Events to the CodeWarrior IDE. You can find it on the CodeWarrior Reference CD in the CodeWarrior Examples folder, under the MacOS Examples folder. This code is a starter project for your work, and you will need to verify the code for proper operation. It is not intended to be a commercially-shipping product.

Largely, you will need to inspect the CodeWarrior IDE’s ‘aet’ and ‘aedt’ resources using a resource editor to see what the low-level codes are to control the IDE. Figure 12.5 shows an example view of what this might look like using the Resorcerer 2.0 resource editor. Rather than document all the low-level codes required to control the IDE, using a resource editor is the best solution to learn the low-level codes for now. With new innovations for the IDE on the horizon, the low-level codes may be documented at a later date.
Figure 12.5  Resorcerer 2.0 View of the CodeWarrior IDE ‘aete’ resource
This chapter describes using the MPW ToolServer development environment from within the CodeWarrior IDE.

Using MPW ToolServer Overview

The topics in this chapter are:

- About ToolServer
- Installing ToolServer
- Starting ToolServer
- Executing MPW Commands
- Looking Up MPW 411 Documentation
- Creating MPW Tool Projects

About ToolServer

ToolServer is a stand-alone version of Apple’s MPW (Macintosh Programmer’s Workshop) development environment. You can use ToolServer to execute MPW commands, MPW tools, and MPW scripts from the ToolServer Worksheet in CodeWarrior. You can also use it to look up information in MPW’s 411 on-line reference database from any CodeWarrior Editor window. This database contains information from *Inside Macintosh*, DTS (Developer Technical Support) Technical Notes, or the MPW Command Reference.

Table 12.69 describes some of the more popular MPW tools.
### Table 12.69 Useful ToolServer tools

<table>
<thead>
<tr>
<th>This command</th>
<th>Does this</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directory folder</strong></td>
<td>Without an argument, displays the current folder. With an argument, sets the current folder to the argument.</td>
</tr>
<tr>
<td><strong>Files</strong></td>
<td>Lists the files in the current folder. Use the wildcard <code>=</code> to limit the search (To type <code>=</code> press Option-x). For example, the command <code>Files w=</code> lists all files that begin with the letter w.</td>
</tr>
<tr>
<td><strong>Duplicate files target</strong></td>
<td>Makes duplicates of files in the folder target.</td>
</tr>
<tr>
<td><strong>Backup -from f1 -to f2</strong></td>
<td>Backs up the files in folder f1 to the folder f2. You can specify which files to back up.</td>
</tr>
<tr>
<td><strong>Rez file</strong></td>
<td>Creates a resource file by compiling the textual description in file.</td>
</tr>
<tr>
<td><strong>Derez file</strong></td>
<td>Creates a textual description of resources by disassembling the resource fork of file.</td>
</tr>
<tr>
<td><strong>Compare file1 file2</strong></td>
<td>Compares two versions of a file and displays any differing lines.</td>
</tr>
</tbody>
</table>
Mac OS CodeWarrior and MPW ToolServer

Installing ToolServer

This section explains how to install the ToolServer application on your computer’s hard drive. To complete this installation, you will need to copy some of the files from the MPW folder on the CodeWarrior CD.

1. Copy the ToolServer 3.4.1.sit StuffIt archive file from the CodeWarrior Reference CD onto your hard drive.

   This archive is located in the Apple Development Tools folder.

### ToolServer and MPW Shell Differences

Unlike the MPW Shell, MPW ToolServer does not let you execute commands that add menus or control windows. However, you can use the File and Line commands to open a CodeWarrior editor window and find a line in it.

<table>
<thead>
<tr>
<th>This command</th>
<th>Does this</th>
</tr>
</thead>
<tbody>
<tr>
<td>File <code>file</code></td>
<td>Opens <code>file</code> in a CodeWarrior Editor window. For example, the command <code>File Test.c</code> opens the file <code>Test.c</code> in the current folder.</td>
</tr>
<tr>
<td>Line <code>number</code></td>
<td>Selects the line numbered <code>number</code> in the front-most CodeWarrior Editor. You’ll usually use Line in a script with File.</td>
</tr>
<tr>
<td>Search <code>files pattern</code></td>
<td>Displays a list of all lines within <code>files</code> that contain <code>pattern</code>. For example, <code>Search /≈Th≈/ ≈.c</code> searches all files ending in <code>.c</code> for words containing the characters Th (To type ≈, press Option-x).</td>
</tr>
</tbody>
</table>
2. Install the Stuffit Expander application (from the CodeWarrior Tools CD) to your hard drive.

You will need to use the Stuffit Expander to unpack the ToolServer archive. Just drag the ToolServer 3.4.1.sit file onto Stuffit Expander.

You can put the ToolServer folder anywhere on your hard drive. CodeWarrior will find it no matter where you place it.

3. Make sure that you only have one copy of the ToolServer application on your system.

4. If you are using a PowerPC Mac OS computer, make sure that the StdCLibInit extension is installed.

   This extension is located in the Extensions folder of your System Folder. You can drag it to the Extensions folder from the CodeWarrior Tools CD, found in the System Folder Items folder.

5. Create an alias to the ToolServer Tools folder.

   The Tools folder is located inside the folder that you just installed, and is the same folder that contains the ToolServer application. Then move this alias to the Metrowerks CodeWarrior folder, the same folder that the CodeWarrior IDE is in.

6. Copy any MPW tools you’ll need into the ToolServer Tools folder that you just installed.

   If you installed the MPW Shell onto your hard drive, you can make aliases of the tools in the MPW Tools folder and place them into the ToolServer Tools folder, which is in the same folder as the CodeWarrior IDE. You can copy any tool you want, but you must copy at least Commando, GetFileName, and GetListItem. If you want to use the MPW 411 database, copy the Get tool also. The 411 database is useful for looking up Macintosh Toolbox calls and other information. If you’ll be using the Rez or Derez resource tools from MPW, copy over these tools too.

7. Create a folder named Interfaces.

   Create a folder named Interfaces in the same folder that contains the ToolServer application you installed.
8. If you do not want to use the MPW 411 on-line database, delete the MWStartup\411 script.

The MWStartup\411 file is in your ToolServer folder. It asks you to locate the MPW 411 database.

9. Copy the MPW folder named RIncludes into the new Interfaces folder.

This folder is located inside the Interfaces folder in the MPW folder on the CodeWarrior CD.

ToolServer is now ready for use with CodeWarrior.

Starting ToolServer

To start ToolServer, select Start ToolServer from the Tools Menu (Mac OS). CodeWarrior starts up ToolServer, creates a ToolServer Worksheet window and adds the ToolServer menu to the menubar. The title of the ToolServer menu is the icon [icon].

When ToolServer starts up, it executes these scripts, in the order listed in Table 12.70.

<table>
<thead>
<tr>
<th>Table 12.70 ToolServer initialization scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This script...</strong></td>
</tr>
<tr>
<td>StartupTS</td>
</tr>
<tr>
<td>UserStartupTS, and all other scripts whose names start with UserStartup\</td>
</tr>
</tbody>
</table>
If you didn’t delete the MWStartup¥411 script, you will be prompted to help it find the 411 database by following these steps:

When ToolServer beeps, choose ToolServer from the application menu.

ToolServer displays a dialog, shown in Figure 12.6, asking you to find the 411 database. If don’t want to use 411, click Cancel.

The dialog disappears. ToolServer continues executing the rest of your startup scripts, and CodeWarrior disables the commands that use 411.

**TIP:** Delete the MWStartup¥411 file as soon as possible if you don’t plan to use 411 with ToolServer, so that ToolServer won’t display this dialog the next time you start it.
Executing MPW Commands

CodeWarrior provides several ways to execute MPW commands, tools, and scripts:

- Using the ToolServer WorkSheet
Mac OS CodeWarrior and MPW ToolServer
Executing MPW Commands

- Using the ToolServer Tools Menu
- Using a Commando Dialog Box

Using the ToolServer Worksheet

The ToolServer Worksheet is a special window. It lets you enter commands, which CodeWarrior sends to ToolServer. CodeWarrior then writes the results of the command back into the window.

To edit text in the worksheet, use the same commands that you use in any Editor window.

Executing a command

To execute a command, script, or tool that you have installed in ToolServer from the worksheet, type the command and press the Enter key. If you don’t have an Enter key on your keyboard, you can use Command-Return instead.

CodeWarrior sends the command to ToolServer and prints ToolServer’s response directly below the command’s line.

If you press Return, instead of Enter or Command-Return, CodeWarrior just moves the cursor to the next line and does not execute your command.

Re-executing a command, script, or tool

To re-execute a command, script or tool, place the cursor somewhere on the command’s line and press the Enter key or Command-Return.

CodeWarrior sends the command to ToolServer and prints ToolServer’s response directly below the command’s line. The new response appears above any old response.

You can also edit a command in the worksheet window before you re-execute it.
Using the ToolServer Tools Menu

An example of the ToolServer Tools Menu (Mac OS), shown in Figure 12.7, is a submenu under the ToolServer menu. This section describes how to add commands to that menu and execute them.

![ToolServer Tools Menu](image)

Figure 12.7 A ToolServer Tools submenu

Executing a script that’s in a CodeWarrior Editor window

Bring the script’s Editor window to the front, then choose Execute as a Script from the , as shown in Figure 12.8.
Mac OS CodeWarrior and MPW ToolServer

Executing MPW Commands

Figure 12.8 ToolServer Menu

Adding a command, script, or tool to the ToolServer Tools submenu

To add a command, tool, or script to the ToolServer Tools submenu, make an alias to the command, tool, or script on your hard disk.

Although you can use the actual command or script, using an alias helps save disk space.

Move the alias into the (ToolServer Tools) folder inside your Metrowerks CodeWarrior folder. The (ToolServer Tools) folder is in the folder that contains the CodeWarrior IDE, and not in the folder that contains ToolServer.

The next time you look at the ToolServer Tools submenu of the ToolServer menu, the command or script will appear.

Executing a command, script, or tool from the ToolServer Tools menu

To execute a command, tool, or script from the ToolServer Tools submenu, first choose the command or script from the ToolServer Tools submenu of the ToolServer menu (Figure 12.7).

If the item is a command with a commando dialog box, ToolServer displays the commando dialog box. If the item is a command with no commando dialog box, ToolServer displays a dialog box that lets
you enter command-line arguments. If the item is a script, ToolServer executes it immediately.

CodeWarrior displays ToolServer’s response in the ToolServer Worksheet.

**Using a Commando Dialog Box**

ToolServer lets you use Commando dialog boxes, which are dialog boxes that help you compose a syntactically perfect command, even when you don’t know the command’s syntax.

This section describes how to bring up these dialogs with the Commando command in the ToolServer menu. You can also bring up a the Commando dialog by type an ellipsis (Option-;) after a command, like this:

```
rez ...
```

Note that not all commands have Commando dialogs. In order for a command to emit a Commando dialog box for you to use, the developer of the command had to do special things to make their tool Commando-aware.

**Executing a command, script, or tool with Commando**

Choose Commando from the ToolServer menu or enter the Commando command in the ToolServer worksheet (press Enter or Command-Return to execute your command).

ToolServer comes to the front and displays the Commando dialog box, as shown in Figure 12.9.
Next, choose the command or script to execute. You can execute either a built-in command, or a tool or script.

To execute a built-in command, choose it from the pop-up menu named Shell Built-in to Execute.

To execute a tool or script, choose Select a tool or script to execute... from the pop-up menu named “Tool or script to execute”. ToolServer displays an open file dialog, as shown in Figure 12.10. Select the tool or script you want and click OK.
When you have finished configuring the Commando dialog box, click the Commando button in the dialog.

CodeWarrior writes a Commando command-line in the ToolServer WorkSheet.

Then, go back to CodeWarrior by selecting CodeWarrior from the application menu, or clicking on a CodeWarrior window to bring the IDE application to the front. The ToolServer Worksheet may appear as shown in Figure 12.11.
Place the cursor in the Commando command-line and press Enter or Command-Return.

After you press Enter, ToolServer comes to the front and displays the Commando dialog for the command, tool, or script. For example, the dialog for Rez is shown in Figure 12.12.
Figure 12.12 The Rez Commando dialog box

Fill in the dialog items to correspond to the options you want, and press the button that contains the command’s name (the Rez button in this case).

After you press the button, CodeWarrior writes a command line in the ToolServer Worksheet. It will look something like Figure 12.13.
Looking Up MPW 411 Documentation

CodeWarrior lets you use MPW's 411 database, which allows you to look up information from *Inside Macintosh*, DTS Technical Notes, or *MPW Command Reference*. Before you use it, make sure the MWStartup•411 script is in your ToolServer folder when you start up ToolServer.

Looking up documentation for a symbol

In a CodeWarrior Editor window, select the name of the symbol.
You can select the name of almost anything that might be in *Inside Macintosh*, including functions, types, and variables.

Next, choose Lookup Symbol from the ToolServer menu.

If CodeWarrior finds the symbol in the 411 database, it displays the information in a CodeWarrior Editor window named Help. The cursor is at the beginning of the information. You may need to scroll to see all of it.

If CodeWarrior can’t find the symbol in the 411 database, it writes an error message in the ToolServer Worksheet and displays information on the last symbol you looked up.

The Help window contains all the 411 information you’ve ever looked up. To remove old information, just delete it. CodeWarrior won’t display it the next time you look something up.

When you’re finished reading the information, close the Help window.

CodeWarrior cannot look up more 411 information if the Help window is open.

**Looking up a template for a function**

In MPW 411, a template is a sample function call which contains the names of the function’s arguments.

In a CodeWarrior Editor window, select the name of the function.

You can select the name of any function described in *Inside Macintosh*.

Choose [Insert Template](#) from the [Tools Menu](#).

CodeWarrior inserts the parameter list after the selected text.
Creating MPW Tool Projects

An MPW tool is a program that you can run under Apple’s MPW (Macintosh Programmers Workshop). MPW tools will work with ToolServer, so you can write your own custom tools to enhance the ToolServer environment.

To learn how to write MPW Tools, refer to the Targeting Mac OS book on your CodeWarrior Reference CD.

To learn how to install your tool into ToolServer, refer to “Installing ToolServer” on page 489.

To learn how to execute your tool after installing it in ToolServer, refer to “Executing MPW Commands” on page 493.
Index

Symbols
#pragma precompile_target 310
(ToolServer Tools) folder 496
.dump 315
'ckid' resources 340

Numerics
411, MPW 502
80x86 16

A
About Metrowerks 384
About Metrowerks command 382
Absolute Path 286
Add Default 284
Add Default button 284
Add File command 366
Add Files command 366
Add Window command 73, 366
Advanced VCS operations 344
Alert Yourself After Build option 308
Alpha 79
Always Search User Paths 283
Always Show Login Dialog option 336
Anchor Global Toolbar (Mac OS) 384
Anchor Global Toolbar command 384
Apple Event 21
Apple Guide Menu 382
Apple Menu (Mac OS) 351
Apple menu, IDE 384
AppleScript 21
Arrow Keys 135
arrow keys 41
Attempt to use dynamic type of C++, Object Pascal and SOM objects 228
Automatic updating 311
Automatically launch applications when SYM file opened 230
Automatically Target Applications 231

B
Background Color 235
backup files 62, 106
Backup MPW command 488
Balance command 139, 358
Balance While Typing option 236
Balancing punctuation 139
Batch search 172
BBEdit 79
BBEdit Extensions Menu 389
BeOS 16, 23
boxes 205
Bring Up To Date command 308, 368
browser 189-221
activating 190, 288
analyzing inheritance 219
base classes in hierarchy 208
base classes in single-class 205
catalog view 192, 196
classes pane 200
customizing windows 221
data members pane 201
declaration button 205
ingediting code 219
file button 203
finding function overrides 219
hierarchy view 194
identifier icon 203
including subprojects 289
interface 195-212
lines in hierarchy 207
member functions pane 201
multi-class 197-203
multi-class hierarchy 206-208
navigating code with 211
opening source file 217
orientation button 202
resize bar 200
saving windows 221
seeing declaration 218
seeing function definition 218
seeing PowerPlant in 220
show hierarchy button 206
showing data in single-class 205
showing subclasses in hierarchy 207
single-class 203
single-class hierarchy 208
source pane 202
strategy 190–195
symbol window 209
synchronized class selection 201, 204, 206
using 213–221
viewing options 190
browser view 192
Build Progress Window command 378

C
C 16
C String 375
C++ 16
Cache Edited Files Between Debug Sessions 229
catalog view 192
catalog window
browser 196
Catch 68K Debugger and DebugStr Traps 232
Catch PowerPC Debugger and DebugStr Traps 232
category pop-up menu, in category window 196
CFM68K 330
Change 287
Changing syntax highlighting colors 241
Character 375
Check Syntax command 367
Checkout Status Column 46
ckid resources 340
classes pane in browser 200
Clear All Breakpoints 359
Clear command 138, 358
Clear Current Watchpoint 374
Clear Floating Toolbar 384
Clear Floating Toolbar command 384
Clear Window Toolbar 383
Clear Window Toolbar command 383
Close All command 354
Close All Variable Windows 380
Close command 354
code navigation pop-up menu 211
CodeWarrior IDE 21
Collapse All 374
Color Syntax option
and printing 111
Commando 385
Commando command 385, 497
comments, coloring 241
Common Sub-Expressions 294
Compare MPW command 488
Compile column 77
Compile command 368
Compiler 291
Compiler Relative 286
Compiler Thread Stack (K) 245
Compiling 297–328
project 299
compiling
source files 301
Compiling and Linking
choosing a compiler 298
compiling files 300–301
debugging 305
disassembling 315
guided tour 316–319
link map 306
making a project 303
options 308
overview 297
plugin compilers 298
precompiling headers 308–314
preprocessing 315
removing binaries 306, 308
removing object code 307
running 304
setting file extension 299
speeding 308
Synchronizing Modification Dates 306
compiling one file 301
compiling selected files 301
Confirm “Kill Process” when closing or quitting 230
Connect on Open option. 336
Contents command 382
Context Popup Delay option 238
Control Menu 351
Controlling syntax highlighting within a window 241
conventions 17
figures 18
host terminology 18
keyboard shortcuts 19
Copy command 138, 358
Copy to Expression 358
Copy/Const Propagation 294
Create Folder 49
Create New Group command 366
Create New Segment (Mac OS) 367
Create New Segment command 367
Create New Target command 366
custom keyword, coloring 241
Cut command 138, 357

D
data members pane, in browser 201
Data Menu 351
date caching 308
Dead Code Elimination 295
Dead Store Elimination 295
Debug all Java class files in directory hierarchy 230
Debug Column 91
Debug Info Marker 91
debugger
generating information 91
Debugger, Display panel 227–228
Debugger, MetroNub panel 231–233
Debugger, Settings panel 229–231
Debugging configuration 304
Debugging, Project 35
Default 375
Default size for unbounded arrays 228
Default Text File Format 238
Default Text File Format option 238
Defining Symbols for C/C++ 312
Defining Symbols for Pascal 313
derez MPW command 488
directory MPW command 488
Disable Debugger 370
Disassemble command 368
Disassembling source code 315
Display panel, Debugger 227–228
documentation
viewers 23

Don’t step into runtime support code 231
DOS text files 108
Download Executable to Remote Machine 232
Drag & Drop editing support 237
Drag and Drop text 138
dump internal browse information after compile 289
dump memory 373
Duplicate MPW command 488
Dvorak KCHR Support 254
Dynamic Scroll option 236

e
Edit Menu 351, 356–360
Balance command 358
Clear command 358
Copy command 358
Cut command 357
Insert Reference Template command 359, 360
Multiple Redo 357
Multiple Undo 357
Paste command 358
Preferences command 360
Redo command 357
Select All command 358
Shift Left command 359
Shift Right command 359
Target Settings command 360
Undo command 357
Version Control Settings command 360
editing
in browser 219
editing, redoing 140
editing, undoing 140
Editor 121–146
adding text 135
balancing punctuation 139
color syntax 141
configuration 129–133
deleting text 135
drag and drop 237
finding a routine in 142
font 130
font preferences 235
Go Back and Go Forward 146
go to line number 146
guided tour 122–129
markers 142–145
moving text 138
navigating text 142–146
opening related file 145
overview 121, 331
panes 131–132
saving window settings 132
selecting text 136
text editing 133–141
text size 130
undoing changes 140–141
user interface elements 122

Editor Extensions Menu 389
Editor Extensions Menu (Mac OS) 351
editors, third party 79
Enable Automatic Toolbar Help (Mac OS) 248
Enable Debugger command 371
Enable Debugging command 370
End Key 41
End key 135
Enter ÔFindÕ String command 165, 362
Enter ÔReplaceÕ String command 362
Entire Word Check Box 169
Enumeration 375
Error Button 317
Error Messages 320
error messages
compiler 321
Errors and Warnings Window command 378
Execute as a Script command 495
Execute as ToolServer Script command 386
Expand 374
extension 290

F
Factory Settings button 225, 274
fat libraries 330
figure conventions 18
file button, in browser 203
File extension 290

File Mappings List 290
File Menu 352–356
Close All command 354
Close command 354
New command 352
New Empty Project command 353
New Project command 353
Open command 353
Open Recent command 353
Open Selection command 353
Print command 356
Print Setup command 356
Quit command 356
Revert command 355
Save A Copy As command 355
Save All command 355
Save As command 355
Save command 354
Switch to MW Debugger command 354

File MPW command 489
file name suffix 290
File Path Caption 127
File Sets List 160
File Sets Pop-up Menu 160
Files 95–112
closing 108–110
creating 95
opening 145
opening existing 96–102
overview 95
reverting to saved 112
saving 103–108, 238–239
files
printing 110–112
printing options 110
read-only 340
selecting 65
source code control 337–349

Files MPW command 488
Files Sets
saving 179
Find command 361
Find Definition & Reference command 364
Find Definition command 364
Find in Next File command 361
Find in Previous File command 362
Find Next command 165, 361
Find Previous command 165, 167, 361
Find Previous Selection command 363
Find Reference & Definition command 148
Find Reference Using (Mac OS) 249
Find Selection command 362
Find Window
  guided tour 153–164
  finding all implementations of function 209
Fixed 375
Flash Delay option 238
Floating Point 375
Floating Toolbar 265
Font Preferences 235
MPW
  411 502
  411, MPW 502
Fract 376
function, finding all 209

G
Get Options 348
Go Back command 365
Go Forward command 365
Go To Line command 365
grep 182–188
groups 73–77
  creating 74
  moving 73
  naming 76
  removing 75
  selecting 65

H
header files
  opening 103
  precompiling 308–314
Help Menu 351
  About Metrowerks command 382
  Contents command 382
  How to Use Help command 382
  Keys command 382
Help menu 382, 382–383
Hexadecimal 375
Hide Breakpoints 359
Hide Global Toolbar 384
Hide Global Toolbar command 384
Hide Window Toolbar 383
Hide Window Toolbar command 383
hierarchy view
  browser 194
Home Key 41
Home key 135
host terminology conventions 18
How to Use Help command 382

I
IDE 15, 21, 28
  guided tour 32–37
  installation 28
identifier icon in browser 203
Ignore Case Check Box 169
Ignore file modification dates when debugging 230
Ignored by Make 292
Ignored by Make option 292
Import “weak” command 330
In variable panes, show variable types by default 227
Include File Cache Size (K) (Mac OS) 245
Initialize Before command 330
Insert Reference Template command 359, 360
Insert Template command 386
Installation 28
Integrated Development Environment 15
Intel 16
Interface Pop-up Menu 124
Interfaces File Pop-up menu 78
Interpret DOS and Unix paths (Mac OS) 284

J
Java 16, 23

K
Keep Program in Background while Stepping 232
Keep Program in Background while Stepping (Mac OS) 232
key bindings 250–261
keyboard conventions 19
Keys command 382
keywords, coloring 241

L
Launchable 292
Launchable option 292
Lifetime Analysis 295
line button in hierarchy browser 207
Line MPW command 489
Line Number Button 128
line number, going to 146
Linker 279
linker
shared libraries 328
linker option 279
Linking 297–328
list button, in browser
browser
list button 201
Lookup Symbol command 386
Loop Invariants 295

M
Mac OS 24, 28
Mac OS Help 382–383
Magic Cap 24
Main Text 241
Main Text Color 235
Make 303
Make command 308, 369
manual style 17
MAP 306
Marker Pop-up Menu 126
member functions pane, in browser 201
memory dump 373
memory requirements 28
Merge into Output command 330
Merge shared libraries option 330
Message List Pane 318
Message Window
command 378
correcting compiler errors 321
error and warning messages 320
stepping through messages 321
using 319–328
Method option 335
MetroNub panel, Debugger 231–233
Metrowerks Visual SourceSafe for Macintosh 46
Microsoft Visual SourceSafe 46
MIPS 24
Modification dates, synchronizing 78
MPW 109, 340
function documentation 502
MPW commands 493
multi-class browser 197–203
multi-class hierarchy, browser 206–208
Multi-file searches 160–164
multiple redo 357
multiple undo 357
multiple Undo command 140

N
New Class Browser command 377
New command 352
New Empty Project command 353
New Expression 374
New Project command 48, 54, 353
Nucleus 24
number, going to line 146

O
Object Master 79
Open Array Window 374
Open command 353
Open Recent command 353
Open Scripts Folder command 388
Open Selection command 353
Open Variable Window 374
opening file with browser 217
Optimize Space 294
Optimize Speed 295
Options 223–270, 271–295
advanced compile options 308
browser coloring 233
Editor settings 234
fonts and tabs 239
overview 223, 271
preferences 226
syntax coloring 240–243
target settings 275–334
options
Access Paths 280–288
Build Extras 288–289
Custom Keywords panel 292
Options Pop-up Menu 126
orientation button
in browser 202
OS-9 24
Others Button 164
Output Directory 280

P
Page Down key 41, 135
Page Setup command 356
Page Up key 41, 135
Palm OS 16, 24
Pane Resize Bar 319
Pane Splitter Controls 128
Panes
in editor window 131–132
Pascal 17
Pascal String 375
Password field 336
Paste command 138, 358
PEF container 330
Play Sound After ‘Bring Up To Date’ & ‘Make’
(Mac OS) 246
PlayStation 16, 24
PlayStation Net Yaroze 24
Port ID 232
Post Linker 280
Post Linker option 280
PowerPC 28, 330
PowerPC Embedded 24
Precompile command 310, 310, 368
precompile_target, #pragma 310
Precompiled 291
Precompiled headers
automatic updating 311
creating 309
precompiled headers
automatic updating of 311
precompiling 308–314
Preferences
choosing 226
Preferences command 360
Prefix key 251
prefix key
Quote Key 252
timeout 253
Preprocess command 368
Preprocessing code 315
preprocessor 315
Print command 356
Print Selection Only 111
Print Setup command 356
Print using Syntax Highlighting option 111
printing
window 111
with syntax coloring 111
project
naming 50
Project Headers Check Box 164
Project Information Caption 318
Project Inspector 328
Project Inspector command 378
Project Menu 351, 366–371
Add File command 366
Add Window command 366
Bring Up To Date command 368
Check Syntax command 367
Compile command 368
Create New Group command 366
Create New Segment command 367
Create New Target command 366
Disassemble command 368
Enable Debugger command 371
Enable Debugging command 370
Make command 369
Precompile command 368
Preprocess command 368
Remove Binaries & Compact command 369
Remove Binaries command 369
Remove Object Code command 369
Remove Selected Items command 367
Reset File Paths command 367
Resume command 371
Run command 371
Set Current Target command 372
Set Default Project command 372
Synchronize Modification Dates command 370
Project Relative Path 286
Project stationery
creating 54
project stationery 53
choosing 48
folder 53
Project Switch List submenu 56
Project Window
guided tour 40–48
navigating 41
Projector 339, 340
Projects 39–94
adding files 67–73
building 52, 299
choosing stationery 48
creating 52–55
creating groups 74
debug enabling 304
debugging 305
expanding and collapsing groups 64
Items Saved with 61
making 303
managing files in 63–90
modifying 52
moving around 41
moving files and groups 73
new 48
opening existing 56–58
opening with Project Switch List 57
removing files and groups 75
running 304
saving 61–62
selecting files and groups 65–66
settings 275–334
stationery folder 53
stationery, about 53
touching and untouching files 77
updating 302
using stationery 48
projects
adding preprocessor symbols to 94
building 308
closing 62
compiling 301, 308
debbug setup 91–92
items saved with 61
naming groups 76
revision control 46
saving 61
selecting files 65
selecting groups 65
switching between 56

Q
QuickStart 22, 28
Quit command 356
Quote Key prefix 252

R
read-only files 340
Recent Documents 247
Recent Strings pop-up menu 167, 171
recompiling 301
recompiling files 301
Redo command 140, 357
Reduction in Strength 294
regular expressions 182–188
Relaxed C Popup Parsing 237
Release notes 17
Remember Password option 336
Remote IP Address 232
Remove 288
Remove a file set command 180
Remove Binaries & Compact command 369
Remove Binaries command 369
Remove Files command 75
Remove Object Code command 369
Remove Selected Items command 367
Replace & Find Next command 363
Replace & Find Previous command 364
Replace All command 170, 364
Replace and Find Previous command 172
Replace command 363
replacing
  in multiple files 173–182
Reset File Paths command 367
Reset Floating Toolbar command 384
Reset Floating Toolbar command 384
Reset Window Toolbar 383
Reset Window Toolbar command 383
resize bar, in browser 200
Resource File 292
Resource file option 292
Resume command 371
Revert command 355
Revert Panel button 226, 274
revision control 337–349, 380
  advanced operations 344
revision control system 46
revision control systems
  options 332
Rez MPW command 488
Routine Pop-up Menu 142
routine pop-Up menu 124
Run command 371

S
Save A Copy As command 62, 106, 355
Save All Before "Update" option 237
Save All command 355
Save As command 104, 355
Save command 354
Save Default Window command 377
Save this File Set command 179
Script menu, IDE 387
Scripts Menu
  Open Scripts Folder command 388
Scripts Menu (Mac OS) 351
Search Menu 351, 360–365
  Enter 'Find' String command 362
  Enter 'Replace' String command 362
  Find command 361
  Find Definition 364
  Find Definition & Reference command 364
Find in Next File command 361
Find in Previous File command 362
Find Next command 361
Find Previous command 361
Find Previous Selection command 363
Find Selection command 362
Go Back command 365
Go Forward command 365
Go To Line command 365
Replace & Find Next command 363
Replace & Find Previous command 364
Replace All command 364
Replace command 363
Search Menu Find Selection command 362
Search MPW command 489
searching 164
  for selection 165
  multi-file 160–164
  multiple files 173–182
  selected text 164–166
Select All 358
Select All command 358
Select stack crawl window when task is
  stopped 230
selected text search 164
selection
  finding 165
  printing 111
Selection Position 235
Set Current Target command 372
Set Default Project command 372
Set Watchpoint 374
Settings panel, Debugger 229–231
shared libraries 330
  linker options 328
Shift Left command 140, 359
Shift Right command 140, 359
shortcut conventions 19
Show Breakpoints 359
Show Catalog Window command 377
Show Global Toolbar 384
Show Global Toolbar command 384
Show Hierarchy Window command 377
Show tasks in separate windows 228
Show Types 374
Show variable values in source code 228
Show Window Toolbar 383
Show Window Toolbar command 383
Signed Decimal 375
single-class browser 203
single-class hierarchy, in browser 208
Sort Function Pop-Up 237
Sort functions by method name in browser 228
source code control 337–349
  advanced operations 344
files 337–349
Source Code Disclosure Triangle 318
Source Code Pane 319
source files
  compiling 301
  precompiling 308–314
source pane, in browser 202
Sources Check Box 162
SourceSafe 332
Stack command 376
Start ToolServer command 385, 491
stationery 53
Stepping Buttons 318
Stop at beginning of ‘main’ when launching applications 230
Stop at EOF option 161, 162, 181
Stop for Debugger and DebugStr traps 231
Stop ToolServer command 385
Switch to Monitor 373
Switch to MW Debugger command 354
symbol window, in browser 209
Synchronize Modification Dates command 370
Synchronizing modification dates 78
syntax coloring
  and printing 111
Syntax coloring, table of 241
System Headers Check Box 163
System Include Path Pane 284
System Paths (Windows) 283
System Relative 286
System Requirements 28

T
Target Name 279
target name option 279
Target Settings command 360
Target Settings panel 278
Targets
  documentation 23
  IDE 23
  settings 275–334
Text
  drag and drop of 138
Text Editing Area 123
text replace
  multiple file 173–182
  Replace All 170
  replacing found text 170
  selective replace 171
  single file 166–172
text search 182–188
  activating multi-file 174
  Batch search 172
  choosing file sets 179
  choosing files 175
  controlling range 168
  controlling search parameters 169
  controlling search range 181
  finding selection 165
  finding text 167
  for selection 165
  multi-file 160–164
  multiple file 173–182
  overview 153
  regular expressions 182–188
  removing file sets 180
  saving file sets 179
  selected text search 164
  single file 166–172
THINK Reference 151
third-party editors 79
__throw() 373
Tile command 376
Tile Vertical command 376
Toolbar Disclosure Button 128
Toolbar Elements Window command 383
Toolbar Submenu 383–384
  Anchor Global Toolbar command 384
  Clear Floating Toolbar command 384
  Clear Window Toolbar command 383
Hide Global Toolbar command 384
Hide Window Toolbar command 383
Reset Floating Toolbar command 384
Reset Window Toolbar command 383
Show Global Toolbar command 384
Show Window Toolbar command 383
Toolbar Elements Window command 383
Toolbar submenu 377

Toolbox Assistant, Macintosh Programmer’s 150

Tools Menu
  Commando command 385
  Execute as ToolServer Script command 386
  Insert Template command 386
  Lookup Symbol command 386
  Start ToolServer command 385
  Stop ToolServer command 385

Tools Menu (Mac OS) 351
Tools Submenu 383-384
ToolServer 21, 502-503
  installing 489
  (ToolServer Tools) folder 496
  ToolServer Tools submenu 496
  ToolServer Worksheet command 378
  Treat #include as #include "..." option 283
  Turbo Pascal 17
  Tutorial Resources 23
  typographical conventions 17

U
Undo command 357
UNIX text files 108
Unsigned Decimal 375
upating projects 301, 302, 308
Use BBEdit Extensions (Mac OS) 249
Use External Editor (Mac OS) 248
Use Modification Dates Caching option 288
Use Multiple Document Interface (Windows) 247
Use Multiple Undo option 237
Use Script Menu (Mac OS) 248
Use Third Party Debugger 290
Use ToolServer Menu (Mac OS) 249
Use Version Control option 335
User Include Path Pane 284
User Paths (Windows) 283
Username field 335

V
Variable change hilite 228
VCS Window 349
Version Control Settings command 360
Version Control System
  advanced operations 344
  Windw 349
Version Control System (VCS) Menu 351, 380
View As 374
View Memory 373
View Memory As 373
viewers 23
Visual SourceSafe for Mac OS 46, 332

W
Warning Button 318
Warning Messages 320
Watchpoint 379
Watchpoint hilite 228
watchpoint window 379
wildcard searching 182-188
Win32/x86 16, 24
Window Menu 376-380
  Build Progress Window command 378
  Errors and Warnings Window command 378
  New Class Browser command 377
  Project Inspector command 378
  Save Default Window command 377
  Show Catalog Window command 377
  Show Hierarchy Window command 377
  Stack command 376
  Tile command 376
  Tile Vertical command 376
  Toolbar submenu 377
  ToolServer Worksheet command 378
  Zoom Window command 376
Window Position and Size 235
Window Toolbar 266
Windows 95 16, 28
Windows Help 382
Windows NT 16, 28
Z
Zoom Window command 376
Zoom Windows To Full Screen (Mac OS) 247
CodeWarrior

IDE User Guide

Credits

writing lead: L. Frank Turovich

other writers: Marc Paquette, Chris Magnuson, Jim Trudeau

engineering: Kevin Bell, Matt Fassio, Matt Henderson, Glenn Meter, Dan Podwall, Nalini Prakash, Rob Vaterlaus, Chris Wiebe

frontline warriors: Ron Liechty, Jim Trudeau, and CodeWarrior users everywhere
Guide to CodeWarrior Documentation

CodeWarrior documentation is modular, like the underlying tools. There are manuals for the core tools, languages, libraries, and targets. The exact documentation provided with any CodeWarrior product is tailored to the tools included with the product. Your product will not have every manual listed here. However, you will probably have additional manuals (not listed here) for utilities or other software specific to your product.

<table>
<thead>
<tr>
<th>Core Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE User Guide</td>
</tr>
<tr>
<td>Debugger User Guide</td>
</tr>
<tr>
<td>CodeWarrior Core Tutorials</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Language/Compiler Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Compilers Reference</td>
</tr>
<tr>
<td>Pascal Compilers Reference</td>
</tr>
<tr>
<td>Error Reference</td>
</tr>
<tr>
<td>Pascal Language Reference</td>
</tr>
<tr>
<td>Assembler Guide</td>
</tr>
<tr>
<td>Command-Line Tools Reference</td>
</tr>
<tr>
<td>Plugin API Manual</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Library Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSL C Reference</td>
</tr>
<tr>
<td>MSL C++ Reference</td>
</tr>
<tr>
<td>Pascal Library Reference</td>
</tr>
<tr>
<td>MFC Reference</td>
</tr>
<tr>
<td>Win32 SDK Reference</td>
</tr>
<tr>
<td>The PowerPlant Book</td>
</tr>
<tr>
<td>PowerPlant Advanced Topics</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Targeting Manuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Targeting BeOS</td>
</tr>
<tr>
<td>Targeting Java VM</td>
</tr>
<tr>
<td>Targeting Mac OS</td>
</tr>
<tr>
<td>Targeting MIPS</td>
</tr>
<tr>
<td>Targeting NEC V810/830</td>
</tr>
<tr>
<td>Targeting Net Yaroze</td>
</tr>
<tr>
<td>Targeting Nucleus</td>
</tr>
<tr>
<td>Targeting OS-9</td>
</tr>
<tr>
<td>Targeting Palm OS</td>
</tr>
<tr>
<td>Targeting PlayStation OS</td>
</tr>
<tr>
<td>Targeting PowerPC Embedded Systems</td>
</tr>
<tr>
<td>Targeting VxWorks</td>
</tr>
<tr>
<td>Targeting Win32</td>
</tr>
</tbody>
</table>