

# **CodeWarrior™**

# **Development Tools**

# **IDE 5.1 User's Guide**

Revised 2002/10/14



Metrowerks, the Metrowerks logo, and CodeWarrior are registered trademarks of Metrowerks Corp. in the US and/or other countries. All other tradenames and trademarks are the property of their respective owners.

Copyright © Metrowerks Corporation. 2002. ALL RIGHTS RESERVED.

**The reproduction and use of this document and related materials are governed by a license agreement media, it may be printed for non-commercial personal use only, in accordance with the license agreement related to the product associated with the documentation. Consult that license agreement before use or reproduction of any portion of this document. If you do not have a copy of the license agreement, contact your Metrowerks representative or call 800-377-5416 (if outside the US call +1 512-997-4700). Subject to the foregoing non-commercial personal use, no portion of this documentation may be reproduced or transmitted in any form or by any means, electronic or mechanical, without prior written permission from Metrowerks.**

Metrowerks reserves the right to make changes to any product described or referred to in this document without further notice. Metrowerks makes no warranty, representation or guarantee regarding the merchantability or fitness of its products for any particular purpose, nor does Metrowerks assume any liability arising out of the application or use of any product described herein and specifically disclaims any and all liability. **Metrowerks software is not authorized for and has not been designed, tested, manufactured, or intended for use in developing applications where the failure, malfunction, or any inaccuracy of the application carries a risk of death, serious bodily injury, or damage to tangible property, including, but not limited to, use in factory control systems, medical devices or facilities, nuclear facilities, aircraft or automobile navigation or communication, emergency systems, or other applications with a similar degree of potential hazard.**

USE OF ALL SOFTWARE, DOCUMENTATION AND RELATED MATERIALS ARE SUBJECT TO THE METROWERKS END USER LICENSE AGREEMENT FOR SUCH PRODUCT.

## How to Contact Metrowerks

Corporate Headquarters	Metrowerks Corporation 9801 Metric Blvd. Austin, TX 78758 U.S.A.
World Wide Web	<a href="http://www.metrowerks.com">http://www.metrowerks.com</a>
Ordering & Technical Support	Voice: (800) 377-5416 Fax: (512) 997-4901

# Table of Contents

---

## I Introduction

<b>1 IDE User's Guide Overview</b>	<b>17</b>
Release Notes . . . . .	17
CodeWarriorU.com . . . . .	17
Documentation Structure . . . . .	18
Documentation Formats . . . . .	18
Documentation Types . . . . .	19
Manual Conventions . . . . .	19
Figure Conventions . . . . .	19
Keyboard Conventions . . . . .	20
Special note for Solaris users and Linux users . . . . .	20

<b>2 CodeWarrior IDE Overview</b>	<b>21</b>
Development Cycle . . . . .	21
CodeWarrior IDE Advantages . . . . .	23
IDE Tools Overview . . . . .	24

## II Projects

<b>3 Working with Projects</b>	<b>29</b>
About Projects . . . . .	29
Project Manager . . . . .	29
Build Targets . . . . .	31
Managing Projects . . . . .	32
Advanced Projects . . . . .	39
Custom Project Stationery . . . . .	40
Subprojects . . . . .	41

## Table of Contents

---

Strategies . . . . .	42
<b>4 Project Window</b>	<b>45</b>
About the Project Window . . . . .	45
Project Window Views . . . . .	47
Files View . . . . .	47
Designs View . . . . .	49
Link Order View . . . . .	49
Targets View . . . . .	50
File, Group, Layout, and Target Management . . . . .	51
Build-Target Management . . . . .	56
<b>5 Working with Files</b>	<b>61</b>
Managing Files . . . . .	61
<b>6 Dockable Windows</b>	<b>69</b>
About Dockable Windows . . . . .	69
Working with Dockable Windows . . . . .	71
Dock Bars . . . . .	76
<b>7 Workspaces</b>	<b>81</b>
About Workspaces . . . . .	81
Using Workspaces. . . . .	81
<b>8 Creating Console Applications</b>	<b>85</b>
About Console Applications. . . . .	85
Creating Console Applications. . . . .	86
<b>III Editor</b>	
<b>9 The CodeWarrior Editor</b>	<b>91</b>
Editor Window . . . . .	91
Editor Toolbar . . . . .	93

---

Interfaces Menu . . . . .	95
Functions Menu . . . . .	95
Markers Menu . . . . .	95
Document Settings Menu . . . . .	96
Version Control System Menu . . . . .	96
Other Editor Window Components . . . . .	97
Path Caption . . . . .	97
File Modification Icon . . . . .	97
Breakpoints Column . . . . .	98
Text Editing Area . . . . .	98
Line and Column Indicator . . . . .	98
Pane Splitter Controls . . . . .	98
<b>10 Editing Source Code</b>	<b>101</b>
Text Manipulation . . . . .	101
Symbol Editing Shortcuts . . . . .	105
Punctuation Balancing . . . . .	105
Code Completion . . . . .	107
Code Completion Configuration . . . . .	107
Code Completion Window . . . . .	110
<b>11 Navigating Source Code</b>	<b>115</b>
Finding Interface Files, Functions, and Lines . . . . .	115
Finding Interface Files . . . . .	116
Locating Functions . . . . .	116
Going Back and Forward . . . . .	118
Using Markers . . . . .	119
Remove Markers Window . . . . .	119
Symbol Definitions . . . . .	121
Reference Templates (Macintosh) . . . . .	123
<b>12 Finding and Replacing Text</b>	<b>125</b>
About Find and Replace . . . . .	125
Single-File Find . . . . .	126

---

---

## Table of Contents

---

Single-File Find and Replace . . . . .	127
Multiple-File Find and Replace . . . . .	129
Search Results Window . . . . .	136
Text-Selection Find . . . . .	137
Regular-Expression Find . . . . .	140
Matching Any Character . . . . .	141
Repeating Expressions . . . . .	141
Grouping Expressions . . . . .	142
Choosing One Character from Many . . . . .	142
Matching Line Beginnings and Endings . . . . .	143
Using the Find String in the Replace String . . . . .	143
Remembering Sub-expressions . . . . .	144

# IV Browser

<b>13 Using the Browser</b>	<b>147</b>
Browser Database . . . . .	147
Browser Data . . . . .	148
Browser Symbols . . . . .	151
Browser Contextual Menu . . . . .	152
<b>14 Using Class Browser Windows</b>	<b>155</b>
Class Browser window . . . . .	155
Classes pane . . . . .	161
Member Functions pane . . . . .	163
Data Members pane . . . . .	164
Source pane . . . . .	165
Status Area . . . . .	165
<b>15 Using Other Browser Windows</b>	<b>167</b>
Multiple-Class Hierarchy Window . . . . .	167
Single-Class Hierarchy Window . . . . .	170

---

Browser Contents window . . . . .	172
Symbols window . . . . .	174
Symbols toolbar . . . . .	175
Symbols pane . . . . .	175
Source pane . . . . .	176
<b>16 Using Browser Wizards</b>	<b>177</b>
The New Class Wizard . . . . .	177
The New Member Function Wizard . . . . .	182
The New Data Member Wizard . . . . .	185
<b>V Debugger</b>	
<b>17 Introduction to Debugging</b>	<b>191</b>
What is a Debugger? . . . . .	191
What is a Symbolics File? . . . . .	192
<b>18 Basic Debugging</b>	<b>193</b>
Thread Window. . . . .	193
Common Debugging Actions . . . . .	195
<b>19 Advanced Debugging</b>	<b>201</b>
What is Advanced Debugging? . . . . .	202
Symbol Hint . . . . .	202
Contextual Menus. . . . .	203
Symbolics Window . . . . .	205
Processes Window . . . . .	208
Expressions Window. . . . .	210
Global Variables Window. . . . .	213
Registers Window. . . . .	214
General Registers . . . . .	215
FPU Registers . . . . .	215

---

---

## Table of Contents

Host-specific Registers . . . . .	216
Log Window . . . . .	218
Variable Window . . . . .	220
Array Window . . . . .	221
Memory Window . . . . .	223
Multi-core Debugging . . . . .	228
<b>20 Using Breakpoints</b>	<b>231</b>
About Breakpoints . . . . .	231
Breakpoints Window . . . . .	231
Common Breakpoint Tasks . . . . .	233
<b>21 Using Watchpoints</b>	<b>239</b>
About Watchpoints . . . . .	239
Using Watchpoints in the Memory Window . . . . .	240
Using Watchpoints in the Watchpoints Window . . . . .	242
Using Watchpoints in Other Windows . . . . .	245
Using Watchpoints in the Thread Window . . . . .	245
Using Watchpoints in the Variable Window . . . . .	247
Using Watchpoints in the Symbolics Window . . . . .	249
<b>22 Using Eventpoints</b>	<b>251</b>
About Eventpoints . . . . .	251
Log Point . . . . .	253
Pause Point . . . . .	254
Script Point . . . . .	254
Skip Point . . . . .	254
Sound Point . . . . .	254
Using Eventpoints in the Breakpoints Window . . . . .	256
Using Eventpoints in Other Windows . . . . .	259
Using Eventpoints in the Thread Window . . . . .	259
Using Eventpoints in the Symbolics Window . . . . .	261

---

---

# VI Compilers and Linkers

<b>23 Compilers</b>	<b>267</b>
Choosing a Compiler . . . . .	267
Compiling Projects . . . . .	268
<b>24 Linkers</b>	<b>273</b>
Choosing Linkers . . . . .	273
Linking Projects . . . . .	274

# VII RAD (Rapid Application Development)

<b>25 Creating RAD Projects</b>	<b>277</b>
RAD Wizards . . . . .	277
Working with Designs . . . . .	278
Working with Layouts . . . . .	282
<b>26 Layout Editing</b>	<b>285</b>
Layout Wizards . . . . .	285
Layout Editor . . . . .	286
Component Palette . . . . .	290
Object Inspector . . . . .	294
<b>27 Component Editing</b>	<b>299</b>
Component Catalog Window . . . . .	299
Working with Catalogs . . . . .	302
Working with Menu Editors . . . . .	304
Menu Editor . . . . .	305
<b>28 Object Wiring</b>	<b>309</b>
About Object Wiring. . . . .	309
Working with Object Wires . . . . .	310

---

## Table of Contents

Using the Create Wire Wizard . . . . .	315
Inspecting Wires . . . . .	317
<b>29 RAD Browsing</b>	<b>321</b>
RAD Browser Window . . . . .	321
Tab Control . . . . .	321
Properties Tab . . . . .	322
Methods Tab . . . . .	322
Events Tab . . . . .	322
Browser RAD Wizards . . . . .	323
<b>VIII Preferences and Target Settings</b>	
<b>30 Customizing the IDE</b>	<b>327</b>
Customizing IDE Commands . . . . .	327
Commands Tab . . . . .	329
Pre-defined Variables in Command Definitions . . . . .	333
Customize Toolbars . . . . .	338
Kinds of Toolbars . . . . .	339
Toolbar Elements . . . . .	340
Modify a Toolbar . . . . .	340
Customize Key Bindings . . . . .	344
<b>31 Working with IDE Preferences</b>	<b>349</b>
IDE Preferences Window . . . . .	349
General Panels . . . . .	351
Build Settings . . . . .	351
Concurrent Compiles . . . . .	353
IDE Extras . . . . .	354
Help Preferences . . . . .	358
Plugin Settings . . . . .	358
Shielded Folders . . . . .	359

---

Source Trees . . . . .	361
Editor Panels . . . . .	365
Code Completion . . . . .	366
Editor Settings . . . . .	366
Font & Tabs . . . . .	368
Text Colors . . . . .	371
Debugger Panels . . . . .	375
Display Settings . . . . .	375
Windowing . . . . .	377
Global Settings . . . . .	378
Remote Connections . . . . .	380
RAD Tools Panels . . . . .	383
Layout Editor . . . . .	384
<b>32 Working with Target Settings</b>	<b>385</b>
Target Settings Window . . . . .	385
Target Panels . . . . .	387
Target Settings . . . . .	388
Access Paths . . . . .	389
Build Extras . . . . .	392
Runtime Settings . . . . .	394
File Mappings . . . . .	396
Source Trees . . . . .	398
Code Generation Panels . . . . .	398
Global Optimizations . . . . .	398
Editor Panels . . . . .	401
Custom Keywords . . . . .	401
Debugger Panels . . . . .	403
Other Executables . . . . .	403
Debugger Settings . . . . .	406
Remote Debugging . . . . .	408

---

<b>33 Preference and Target Settings Options</b>	<b>411</b>
--	------------

## IX Menus

<b>34 IDE Menus</b>	<b>451</b>
---------------------	------------

Windows Menu Layout . . . . .	451
File Menu . . . . .	451
Edit Menu . . . . .	453
View Menu . . . . .	454
Search Menu . . . . .	455
Project Menu . . . . .	457
Debug Menu . . . . .	459
Data Menu . . . . .	461
Window Menu . . . . .	462
Help Menu . . . . .	463
Macintosh Menu Layout . . . . .	464
Apple Menu . . . . .	464
CodeWarrior Menu . . . . .	464
File Menu . . . . .	464
Edit Menu . . . . .	466
Search Menu . . . . .	467
Project Menu . . . . .	469
Debug Menu . . . . .	471
Data Menu . . . . .	473
Window Menu . . . . .	475
VCS Menu . . . . .	476
Tools Menu . . . . .	476
Scripts Menu . . . . .	477
Help Menu . . . . .	477
Rapid Application Development (RAD) Menus . . . . .	478
Align Menu . . . . .	478

Layout Menu . . . . .	478
Resize Menu . . . . .	479
<b>35 Menu Commands</b>	<b>481</b>
<b>Index</b>	<b>521</b>

## Table of Contents

---

# Introduction

---

This section contains these chapters:

- [IDE User's Guide Overview](#)
- [CodeWarrior IDE Overview](#)



# IDE User's Guide Overview

---

This chapter of the *CodeWarrior™ IDE User's Guide* is a high-level description of documentation and training resources for learning to use the IDE:

- CodeWarriorU.com—free, Internet-based instruction for CodeWarrior products. Use this resource to learn more about the CodeWarrior Integrated Development Environment (IDE) and computer programming.
- Documentation structure—a guide to the various CodeWarrior manuals available. This guide notes the location of generic and specific product documentation.
- Common conventions—some common typographical conventions used in this manual and other Metrowerks documentation.

This chapter contains these sections:

- [“Release Notes” on page 17](#)
- [“CodeWarriorU.com” on page 17](#)
- [“Documentation Structure” on page 18](#)
- [“Manual Conventions” on page 19](#)

## Release Notes

Please read the release notes. They contain important last-minute additions to the documentation. The `Release Notes` folder on the CodeWarrior CD contains the information.

## CodeWarriorU.com

CodeWarriorU.com offers a wide range of free, Internet-based courses in a wide variety of computer programming topics. Use this supplement to the CodeWarrior documentation to acquire more experience using CodeWarrior products.

CodeWarriorU.com courses feature:

- Text-based instruction
- Expert instructors
- A variety of self-assessment and study materials
- Interactive message boards for communicating with instructors and fellow students

Courses now available at CodeWarriorU include:

- Learn Programming in C

For beginning programmers. It teaches how to create C software.

- Introduction to Java

For beginning and experienced programmers. It teaches how to create Java software.

- Introduction to C++

For beginning and experienced programmers. It teaches how to create C++ software.

- Intermediate C++

For programmers who completed the Introduction to C++ course and have basic C++ programming knowledge. It provides the foundation needed to create more sophisticated C++ software.

To find out more, visit this web site:

<http://www.CodeWarriorU.com/>

## Documentation Structure

CodeWarrior products include an extensive documentation library of user guides, targeting manuals, and reference manuals. Take advantage of this library to learn how to efficiently develop software using the CodeWarrior programming environment.

## Documentation Formats

CodeWarrior documentation presents information in various formats:

- **Print**—Printed versions of CodeWarrior manuals, including the *IDE User's Guide*, *MSL C Reference*, *C/C++ Reference*, and product-focused *Targeting* manuals.

- **PDF** (Portable Document Format)—Electronic versions of CodeWarrior manuals. The CodeWarrior CD Documentation folder contains the electronic PDF manuals.
- **HTML** (Hypertext Markup Language)—HTML or Compressed HTML (.CHM) versions of CodeWarrior manuals.

## Documentation Types

Each CodeWarrior manual focuses on a particular information type:

- **User guides**—User guides provide basic information about the CodeWarrior user interface. User guides include information that supports all host platforms on which the software operates, but do not include in-depth platform-specific information. An example is the *PowerParts User Guide*.
- **Targeting manuals**—Targeting manuals provide specific information required to create software that operates on a particular platform or microprocessor. Examples include the *Targeting Windows*, *Targeting Java*, and *Targeting DSP56800* manuals.
- **Reference manuals**—Reference manuals provide specialized information that supports coding libraries, programming languages, and the IDE. Examples include the *C Compiler Reference*, *MSL C Reference*, and *Extending the CodeWarrior IDE* manuals.
- **Core manuals**—Core manuals explain the core technologies available in the CodeWarrior IDE. Examples include:
  - *IDE User's Guide*
  - *C/C++ Compilers Reference*
  - *MSL C Reference* and *MSL C++ Reference*
  - *Extending the CodeWarrior IDE*
  - *Command-Line Tools Reference*

## Manual Conventions

This section explains conventions in the *IDE User's Guide*.

### Figure Conventions

The CodeWarrior IDE employs a virtually identical user interface across multiple hosts. For this reason, illustrations of common interface elements use images from any

host. However, some interface elements are unique to a particular host. In such cases, clearly labelled images identify the specific host.

## Keyboard Conventions

The CodeWarrior IDE accepts keyboard shortcuts, or *key bindings*, for frequently used operations. For each operation, this manual lists corresponding key bindings by platform. Hyphens separate multiple keystrokes in each key binding.

## Special note for Solaris users and Linux users

The Solaris and Linux IDE use Macintosh symbols to represent modifier keys in key bindings. [Table 1.1](#) shows the relationship between the Macintosh symbols and the equivalent modifier keys on Solaris and Linux computers.

**Table 1.1 Macintosh modifier-key equivalents for Solaris and Linux**

Symbol	Macintosh Name	Solaris Equivalent	Linux Equivalent
⌘	Control	Control	Ctrl
⌥	Option	Alt	Alt
⌘	Command	Meta	Alt
⇧	Shift	Shift	Shift

Solaris and Linux computers can map a modifier key to any key on the keyboard. The preceding table reflects the default modifier key configuration for these computers. Remember that custom mappings supersede the default configuration noted in the table.

# CodeWarrior IDE Overview

---

The CodeWarrior™ Integrated Development Environment (IDE) provides an efficient and flexible software-development tool suite. This chapter explains the advantages of using the CodeWarrior IDE and provides brief descriptions of the major tools that make up the IDE.

This chapter contains these sections:

- [“Development Cycle” on page 21](#)
- [“CodeWarrior IDE Advantages” on page 23](#)
- [“IDE Tools Overview” on page 24](#)

## Development Cycle

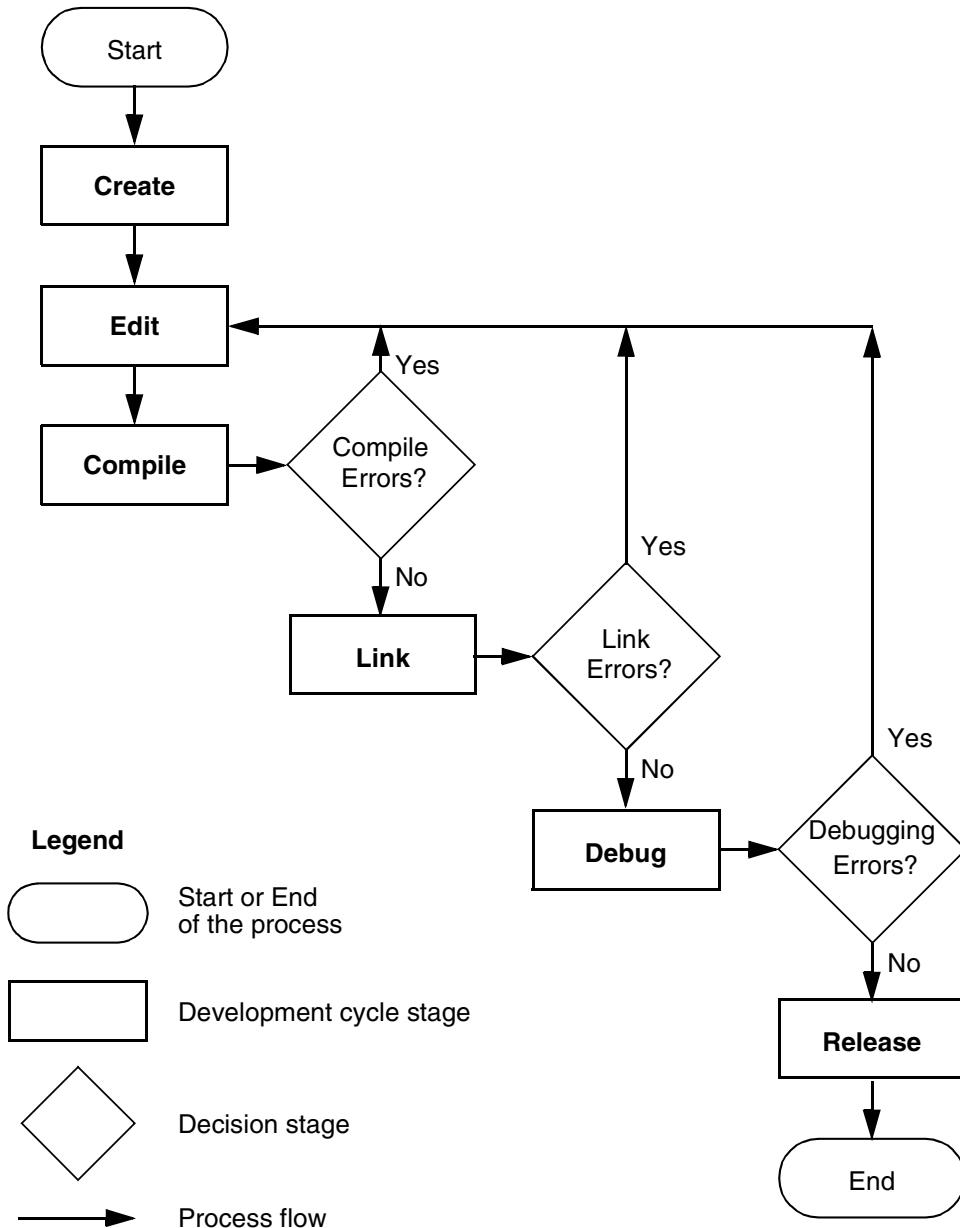
A software developer follows a general development process:

- Begin with an idea for new software.
- Implement the new idea in source code.
- Have the IDE compile source code into machine code.
- Have the IDE link machine code and form an executable file.
- Correct errors (debug).
- Compile, link, and release a final executable file.

The stages of the development cycle correspond to one or more chapters in this manual.

[Figure 2.1 on page 22](#) depicts the development cycle as a flowchart. [Table 2.1 on page 23](#) details the different stages and their corresponding sections in this manual.

**Figure 2.1 The Development Cycle diagram**



**Table 2.1 Stage descriptions and related sections in the IDE User's Guide**

Stage	Description	Related Sections
<b>Create</b>	Create the initial project, source files, and build targets that implement an idea for new software.	<ul style="list-style-type: none"> <li>• <a href="#">“Projects” on page 27</a></li> <li>• <a href="#">“Preferences and Target Settings” on page 325</a></li> <li>• <a href="#">“Menus” on page 449</a></li> </ul>
<b>Edit</b>	Transform the idea into working source code, organize interface elements, and correct errors.	<ul style="list-style-type: none"> <li>• <a href="#">“Editor” on page 89</a></li> <li>• <a href="#">“Browser” on page 145</a></li> <li>• <a href="#">“RAD (Rapid Application Development)” on page 275</a></li> </ul>
<b>Compile</b>	Compile the source code into machine format that operates on the target host.	<a href="#">“Compilers and Linkers” on page 265</a>
<b>Link</b>	Link the separate compiled modules into a single binary executable file.	<a href="#">“Compilers and Linkers” on page 265</a>
<b>Debug</b>	Find and resolve all coding and logic errors that prevent the program from operating as designed.	<a href="#">“Debugger” on page 189</a>
<b>Release</b>	Release for public use.	Beyond the scope of this manual.

## CodeWarrior IDE Advantages

Software developers take advantage of CodeWarrior IDE features during software development:

- Cross-platform development

Develop software to run on multiple operating systems, or use multiple hosts to develop the same software project. The IDE runs on popular operating systems, including Windows, Macintosh, Solaris, and Linux. The IDE uses virtually the same graphical user interface (GUI) across all hosts.

- Multiple-language support

Choose from multiple programming languages when developing software. The IDE supports high-level languages, such as C, C++, and the Java programming language, as well as in-line assemblers for most processors.

- Consistent development environment

Port software to new processors without having to learn new tools or lose an existing code base. The IDE supports many common desktop and embedded processor families, including x86, PowerPC, MIPS, and many others.

- 
- Plug-in tool support

Extend the capabilities of the IDE by adding a plug-in tool that supports new services. The IDE currently supports plug-ins for compilers, linkers, pre-linkers, post-linkers, preference panels, version controls, and other tools. Plug-ins make it possible for the CodeWarrior IDE to process different languages and support different processor families.

## IDE Tools Overview

The CodeWarrior IDE is a tool suite that provides sophisticated tools for software development. This section explains the standard tools available in the IDE:

- a project manager
- an editor
- a search engine
- a source browser
- a build system
- a debugger
- Rapid Application Development (RAD) tools

[Table 2.2 on page 25](#) explains the purpose of these tools and lists corresponding CodeWarrior IDE features.

**Table 2.2 IDE tools and features**

Tool	Purpose	CodeWarrior IDE Features
<b>Project Manager</b>	Manipulate items associated with a project	<ul style="list-style-type: none"> <li>Handles top-level file management for the software developer</li> <li>Organizes project items by major group, such as files and targets</li> <li>Tracks state information (such as file-modification dates)</li> <li>Determines build order and inclusion of specific files in each build</li> <li>Coordinates with plug-ins to provide version-control services</li> </ul>
<b>Editor</b>	Create and modify source code	<ul style="list-style-type: none"> <li>Uses color to differentiate programming-language keywords</li> <li>Allows definition of custom keywords for additional color schemes</li> <li>Automatically verifies parenthesis, brace, and bracket balance</li> <li>Allows use of menus for navigation to any function or into the header files used by the program</li> </ul>
<b>Search Engine</b>	Find and replace text	<ul style="list-style-type: none"> <li>Finds a specific text string</li> <li>Replaces found text with substitute text</li> <li>Allows use of regular expressions</li> <li>Provides file-comparison and differencing functionality</li> </ul>
<b>Source Browser</b>	Manage and view program symbols	<ul style="list-style-type: none"> <li>Maintains a symbolics database for the program. Sample symbols include names and values of variables and functions.</li> <li>Uses the symbolics database to assist code navigation</li> <li>Links every symbol to other locations in the code related to that symbol</li> <li>Processes both object-oriented and procedural languages</li> </ul>
<b>Build System</b>	Convert source code into an executable file	<ul style="list-style-type: none"> <li>Uses the compiler to generate object code from source code</li> <li>Uses the linker to generate a final executable file from object code</li> </ul>

**Table 2.2 IDE tools and features (*continued*)**

Tool	Purpose	CodeWarrior IDE Features
Debugger	Resolve errors	<ul style="list-style-type: none"><li>• Uses the symbolics database to provide source-level debugging</li><li>• Supports symbol formats such as CodeView, DWARF (Debug With Arbitrary Records Format), and SYM (SYMbolic information format)</li></ul>
RAD tools	Graphically develop a user interface for the project	<ul style="list-style-type: none"><li>• Allows onscreen editing of a user interface as it appears to the end user</li><li>• Organizes user-interface elements into sets of onscreen buttons</li><li>• Handles generation of code to implement the user interface</li></ul>



# Projects

---

This section contains these chapters:

- [Working with Projects](#)
- [Project Window](#)
- [Working with Files](#)
- [Dockable Windows](#)
- [Workspaces](#)
- [Creating Console Applications](#)



# Working with Projects

---

This chapter explains how to work with projects in the CodeWarrior™ IDE. Projects organize several file types associated with a computer program:

- Text files—files that contain any kind of text. Sample text files include Read Me files and source files.
- Source files—files that contain source code only. Sample source files include C++ files and Java files.
- Library files—files that contain special code designed to work together with a particular programming language or operating environment.
- Generated files—files created by the IDE while building or debugging the project.

This chapter contains these sections:

- [“About Projects” on page 29](#)
- [“Managing Projects” on page 32](#)
- [“Advanced Projects” on page 39](#)

## About Projects

The IDE uses build targets and a Project Manager to organize source code and support files. This section explains both components.

## Project Manager

The IDE gathers source, library, resource, and other files into a *project*. The Project Manager manipulates the information stored in the project.

[Figure 3.1 on page 30](#) diagrams Project Manager interactions with IDE tools. [Table 3.1 on page 30](#) explains the interactions.

Figure 3.1 Project Manager

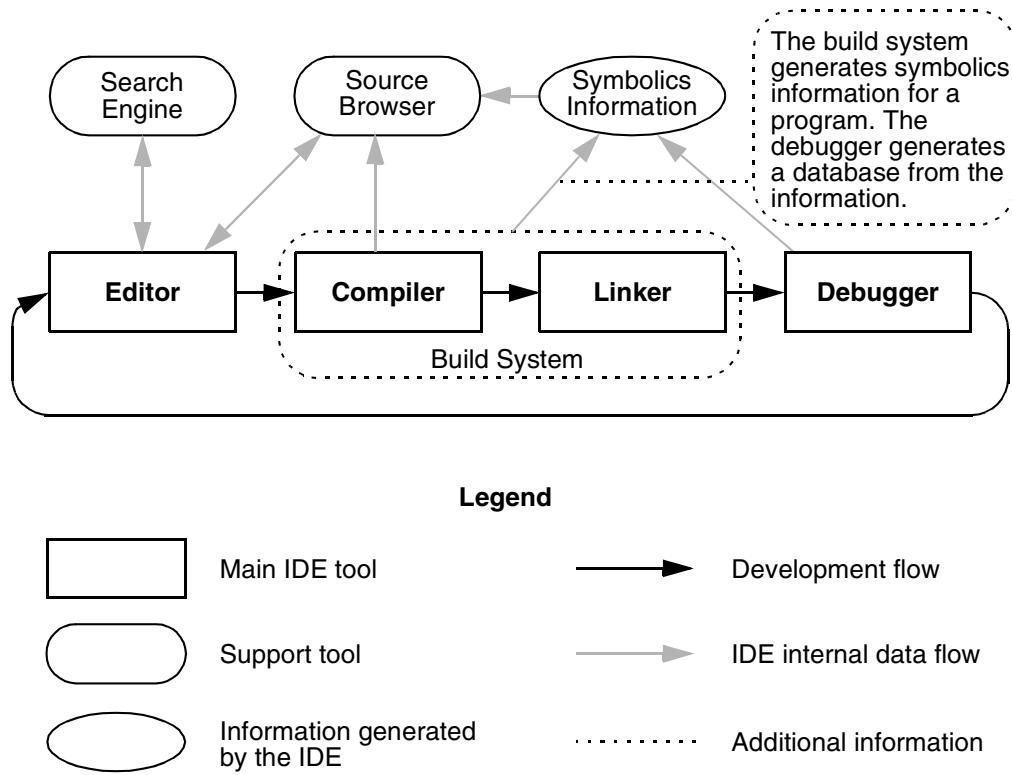


Table 3.1 Project Manager interactions

IDE Tool	Project Manager Interactions
Editor	<ul style="list-style-type: none"><li>Coordinating internal data flow among editor windows, the search engine, and the source browser</li><li>Matching find-and-replace results between related header files and source files</li><li>Associating functions and variables with their corresponding source code</li></ul>
Compiler	<ul style="list-style-type: none"><li>Synchronizing with source code a symbolics database of program functions, variables, and values</li><li>Coordinating internal data flow between the symbolics database and the source browser</li><li>Determining the files to include in the build process</li></ul>

**Table 3.1 Project Manager interactions (*continued*)**

IDE Tool	Project Manager Interactions
Linker	<ul style="list-style-type: none"><li>• Sending compiled object code to the linker for conversion to executable code</li><li>• Setting the link order for processing compiled object code</li></ul>
Debugger	<ul style="list-style-type: none"><li>• Matching debugging data to source code</li><li>• Updating the symbolics database to reflect changing values during a debugging session</li></ul>

## Build Targets

For any given build, the project manager tracks:

- files and libraries
- link order
- dependencies
- compiler, linker, and other settings

The IDE stores this information in a *build target*. As the project changes, the project manager automatically updates the build target. The project manager also coordinates program builds, using the build-target information to call the appropriate tools in the correct order with the specified settings.

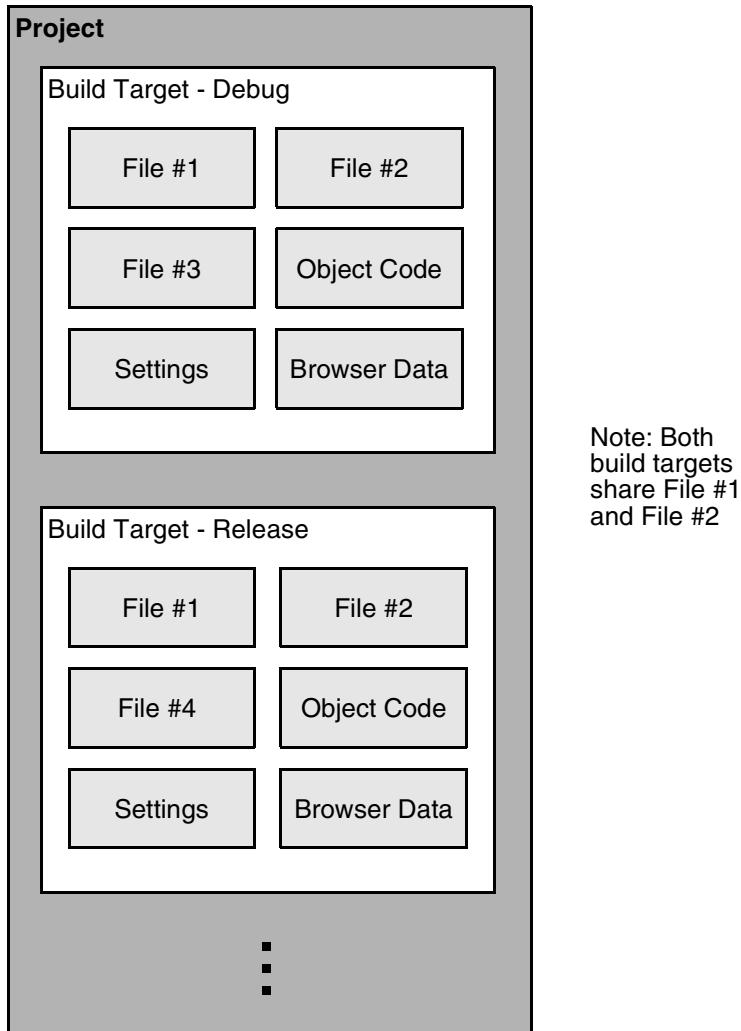
For example, the project manager directs the build system to compile only those source files that rely on the information in a modified file.

Note that all of this operation happens automatically. The software developer does not need to remember makefile syntax or semantics, and never has to debug makefile syntax errors. The IDE simplifies the process, making it easier to develop software.

The project manager also supports multiple build targets within the same project file. Each build target can have its own unique settings, and even use different source and library files. For example, it is common to have both debug and release build targets in a project.

[Figure 3.2 on page 32](#) shows a sample project with debug and release build targets.

**Figure 3.2 Project with multiple build targets**



## Managing Projects

Use these tasks to manage projects:

- Create a new project

- Open an existing project
- Save an existing project
- Close an open project
- Inspect an open project
- Print an open project

---

## Creating New Projects using Project Stationery

Use the project stationery provided with the IDE to quickly create new projects. The stationery contains everything needed for a minimal, ready-to-run project. Use project stationery as a foundation upon which to add features for each new programming endeavor.

1. Choose **File > New**.
2. Click the **Project** tab and select a project type.
3. Enter a project name (include the `.mcp` extension) in the **Project Name** field and set the **Location** for the new project.
4. Click **OK** in the **New** window.
5. Select the appropriate project stationery from the **New Project** window.
6. Click **OK** in the **New Project** window.

The IDE uses the selected project stationery as a template to create a new project.

---

## Creating New Projects from Makefiles

Use the Makefile Importer wizard provided with the Windows, Solaris, and Linux IDE to convert most Visual C `nmake` or GNU `make` files into projects. The wizard performs these tasks:

- Parses the makefile to determine source files and build targets
- Creates a project
- Adds the source files and build targets determined during parsing

- Matches makefile information, such as output name, output directory, and access paths, with the newly created build targets.
  - Selects a project linker
1. Choose **File > New**.
  2. Click the **Project** tab.
  3. Select **Makefile Importer Wizard**.
  4. Enter a project name (include the .mcp extension) in the **Project Name** field and set the **Location** for the new project.
  5. Click **OK** in the **New** window.
  6. Enter the path to the makefile in the **Makefile location** field or click **Browse** to navigate to the makefile.
  7. Choose the tool set used for makefile conversion and linker selection.
    - **Tool Set Used In Makefile**—Choose the tool set whose build rules form the basis of the makefile.
    - **Metrowerks Tool Set**—Choose the linker tool set to use with the generated project.
  8. Select the desired diagnostic settings.
    - **Log Targets Bypassed**—Select to log information about makefile build targets that the IDE fails to convert to project build targets.
    - **Log Build Rules Discarded**—Select to log information about makefile rules that the IDE discards during conversion.
    - **Log All Statements Bypassed**—Select to log targets bypassed, build rules discarded, and other makefile items that the IDE fails to convert.
  9. Click **Finish**, then **Generate**.

The Makefile Importer wizard performs the conversion process and displays additional information.

## Creating Empty Projects

Unlike project stationery, empty projects do not contain a pre-configured collection of template source files, library files, or build targets. Empty projects allow advanced software engineers to custom-build new projects from scratch.

<b>NOTE</b>	Avoid creating empty projects. Instead, modify a project created with project stationery. Project stationery pre-configures complicated settings to quickly get started.
-------------	--

1. Choose **File > New**.
2. Click the **Project** tab and select **Empty Project**.
3. Enter a project name (include the **.mcp** extension) in the **Project Name** field and set the **Location** for the new project.
4. Click **OK** in the **New** window.

The IDE creates an empty project. Add files and libraries, create build targets, and choose the appropriate target settings to complete the new project.

---

## Opening Projects

Use the IDE to open previously saved projects. CodeWarrior projects normally end in the *Metrowerks CodeWarrior Project* extension of **.mcp**. Open projects to add, remove, or modify files to enhance the capabilities of the final executable file.

1. Choose **File > Open**.
2. Find and select the project to open.
3. Click **Open**.

The IDE opens the project and displays its Project window.

**NOTE** The IDE prompts you for confirmation to update projects created in older CodeWarrior versions.

---

## **Opening Projects Created on Other Hosts**

CodeWarrior projects whose names end in .mcp are cross-platform. However, the object code stored inside each project folder is not cross-platform. Use these procedure to properly open the project on a different host computer.

1. If not present, add the .mcp file-name extension to the project name.
2. Copy the project folder from the original host to the new host.
3. Delete the Data folder inside the newly copied project folder.
4. Open the newly copied project on the new host IDE.
5. Recompile the project to generate new object code.

---

## **Saving Projects**

The IDE automatically saves projects and updates project information after performing these actions:

- Closing the project
- Applying or saving a preference or target-setting option
- Adding, deleting, or compiling a file
- Editing group information
- Removing or compacting object code
- Quitting the IDE

---

## **Inspecting Project Files**

Use the **Project Inspector** command to review and configure source-file attributes and target information in the Project Inspector window.

1. Select a file in the Project window.
2. Open the **Project Inspector** window, as explained in [Table 3.2](#).

**Table 3.2 Opening the Project Inspector window**

On this host...	Do this...
Windows	Select <b>View &gt; Project Inspector</b> .
Macintosh	Select <b>Window &gt; Project Inspector</b> .
Solaris	Select <b>Window &gt; Project Inspector</b> .
Linux	Select <b>Window &gt; Project Inspector</b> .

3. Examine the source-file attributes and target settings.
  - Click the **Attributes** tab to view the file attributes.
  - Click the **Targets** tab to view the build targets that use the file.

---

## Printing Projects

The Project Manager can print a complete listing of the **Files, Designs, Link Order**, or **Targets** tab currently displayed in the Project window.

1. Select the Project window.
2. Click the **Files, Designs, Link Order**, or **Targets** tab.
3. Choose **File > Print**.
4. Set the print options in the print dialog.
5. Print the Project window contents, as explained in [Table 3.3](#).

**Table 3.3 Printing the Project window contents**

On this host...	Do this...
Windows	Click <b>OK</b> .
Macintosh	Click <b>Print</b> .

**Table 3.3 Printing the Project window contents (*continued*)**

On this host...	Do this...
Solaris	Click <b>Print</b> .
Linux	Click <b>Print</b> .

The IDE prints the contents of the selected tab in the Project window.

---

## Choosing a Default Project

The IDE allows multiple open projects at the same time. However, a given source file can belong to more than one open project, making it ambiguous as to which project a source-file operation applies.

To resolve ambiguity, choose the default project to which the IDE applies operations.

1. If only one project is open, it automatically becomes the default project.
2. If more than one project is open, choose **Project > Set Default Project** to select the desired default project.

In ambiguous situations, the IDE applies operations to the selected default project.

---

## Exporting Projects to XML Files

The IDE can export a project to an Extensible Markup Language (XML) file. Use this capability to store projects in text-oriented environments, such as a version control system.

1. Bring forward the project to export.
2. Choose **File > Export Project**.
3. Name the exported XML file and save it in the desired location.

The IDE converts the project to an XML file.

## Importing Projects Saved as XML Files

The IDE can import a project previously saved in Extensible Markup Language (XML) format. Use this capability to recreate projects stored in text-oriented environments, such as a version control system.

1. Choose **File > Import Project**.
2. Create a new folder in which to save the converted project and all of its generated files.
3. Find the XML file that you want to import.
4. Save the XML file in the newly created folder.

The IDE converts the XML file to a project.

---

## Closing Projects

Use the **Close** command to close a CodeWarrior project file at the end of a programming session. The IDE automatically saves changes to a closed project.

1. Select the Project window to close.
2. Close the project.
  - Choose **File > Close**.
  - Click the close box in the Project window.

The IDE closes the project.

# Advanced Projects

Advanced projects deal with these topics:

- Custom project stationery—modified project stationery tailored to advanced programming needs.
- Subprojects—projects within projects.
- Strategies—obtaining the maximum benefit from advanced projects.

## Custom Project Stationery

Use custom project stationery to develop streamlined templates to meet advanced programming needs:

- Pre-configure new project stationery to include often-used files, libraries, and source code
- Configure build targets and options to any desired state
- Set up a reusable template to use for creating projects

---

<b>NOTE</b>	Custom project stationery requires in-depth knowledge about project structure and operation. Before creating custom stationery, be sure to fully understand existing project stationery included with the CodeWarrior product.
-------------	--

---

## Creating Custom Project Stationery

Use custom project stationery to develop a convenient template for creating new projects. An efficient way to develop custom stationery is to modify existing project stationery and save it under a new name in the **Stationery** or **Project Stationery** folder.

1. Follow the usual process for creating a project from project stationery.  
See "[Creating New Projects using Project Stationery](#)" on page 33 for more information.
2. Choose **File > Save A Copy As**.
3. Find the **Project Stationery** folder in the CodeWarrior installation.
4. Create a folder inside the **Project Stationery** folder to store the newly created project.
5. **Save** the project to its new folder. Use a descriptive project name with the `.mcp` extension.
6. Customize the newly saved project so that it becomes a template for creating other projects:

- Add source files to the project. Save these files in the same folder as the project itself.
  - Add build targets for building the project with frequently used settings.
  - Configure other project preferences as desired.
7. Close the customized project to save it.
  8. Open the customized project folder inside the **Project Stationery** folder.
  9. Find and delete the `_Data` folder.

The IDE now treats the customized project as project stationery. The descriptive name appears in the **Project** tab of the **New** window.

## Subprojects

A subproject is a project nested inside a parent project. Subprojects organize source code for the IDE to build prior to building the parent project. For example, the IDE builds subprojects for an application's plug-ins before building the parent project for the application itself.

---

### Adding Subprojects to a Project

Use a subproject to organize a separate set of source files and build targets inside a parent project.

1. Open the parent project in which to add a subproject. The parent Project window opens.
2. Click the **Files** tab in the Project window.
3. If the parent project has more than one build target, use the build-target list box in the Project window toolbar to choose the desired build target.
4. Add a separate project to the Project window:
  - Drag and drop the `.mcp` file of the separate project into the Project window, or
  - Choose **Project > Add Files** to add the `.mcp` file of the separate project.

The IDE treats the added project as a subproject. The subproject appears in the **Files** view of the parent Project window.

## Opening Subprojects

The IDE can open a subproject from the parent Project window. Use this feature to more conveniently open the subproject.

1. Double-click the subproject in the **Files** view of the parent Project window.
2. The IDE opens the subproject in its own Project window.

## Strategies

Projects can organize files into build targets or subprojects. Each of these structures has its own advantages. Choose the structure best suited to the programming need.

### Build Targets

Build targets organize collections of files inside a project. Build targets have these advantages:

- Using multiple build targets inside a single project allows access to all source code for that project.
- Build targets organize different collections of build settings for a single project.
- Each project accommodates up to 255 build targets.

### Subprojects

Subprojects incorporate separate, standalone projects into parent projects. Subprojects have these advantages:

- Subprojects separate distinct parts of a complex program, such as an application and its various plug-ins.
- Using subprojects streamlines a complicated build. For example, create a project that builds all plug-ins for an application. Add this project as a subproject of the main application. The IDE then builds all plug-ins before building the main application.
- Use subprojects to break down a complicated project that approaches the 255 build-target limit. Organize related build targets into different subprojects to improve build speed.





# Project Window

---

This chapter explains how to work with the Project window in the CodeWarrior™ IDE. The Project window provides these features:

- Organization—view and modify all files created for use with a computer program.
- Views—manipulate files arranged by type.
- Management—control the way the IDE handles files.

This chapter contains these sections:

- [“About the Project Window” on page 45](#)
- [“Project Window Views” on page 47](#)
- [“File, Group, Layout, and Target Management” on page 51](#)
- [“Build-Target Management” on page 56](#)

## About the Project Window

The Project window organizes files in a computer program. Use this window to control various aspects of each file. The window includes these items:

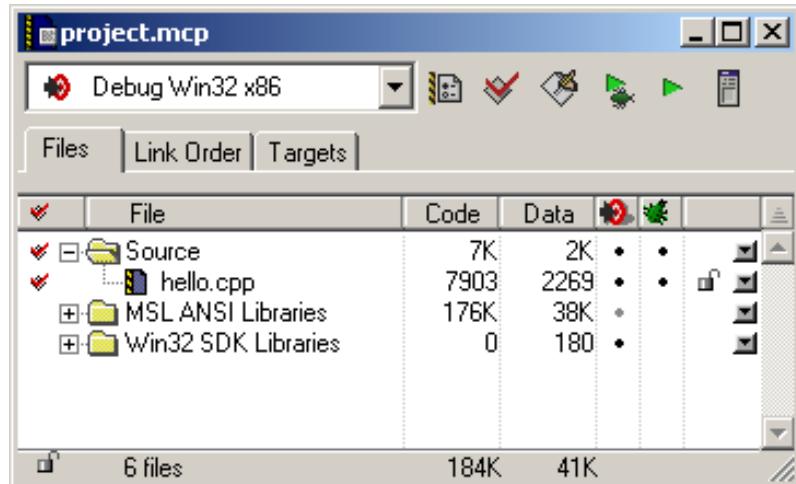
- Project window toolbar
- View tabs
- Columns

[Figure 4.1 on page 46](#) shows a sample Project window. [Table 4.1 on page 46](#) explains the items in the Project window.

## Project Window

*About the Project Window*

**Figure 4.1 Project window**



**Table 4.1 Project window—items**

Item	Icon	Explanation
Current Target		Choose the current build target with which to work.
Target Settings		Click to view and edit the settings for the current build target.
Synchronize Modification Dates		Click to check the modification dates of each project file and mark those files that need compilation.
Make		Click to compile and link all modified and manually selected (touched) project files.
Debug		Click to debug the current build target.
Run		Click to compile and link the current build target, then run the program.
Project Inspector		Click to view project information and edit file-specific information.

**Table 4.1 Project window—items (*continued*)**

Item	Icon	Explanation
Files		Click this tab to display a list of files in the project and their properties.
Designs		Click this tab to display the rapid application development (RAD) designs in the project. The Designs tab appears only in projects that support RAD development.
Link Order		Click this tab to display the link order of files in the current build target.
Frameworks		Click this tab to display available programming frameworks to link against. The Frameworks tab appears only in projects that support frameworks.
Targets		Click this tab to display a list of all project build targets, sub-projects, and target-linking information.

## Project Window Views

The Project window uses views to organize items:

- Files
- Designs (for projects supporting rapid application development)
- Link Order
- Targets
- Frameworks (for projects supporting code frameworks)

## Files View

The Files view presents information about individual files in a project. The Files view typically contain certain file types:

- Text files—files that contain any type of text. Sample text files include Read Me files and source files.
- Source files—files that contain source code only. Sample source files include C++ files and Java files.
- Library files—files that contain special code designed to work together with a particular programming language or operating environment.

[Table 4.2](#) explains the items in the Files view.

**Table 4.2 Files view—items**

Item	Icon	Explanation
Touch		Indicates the touch status of each file. Click in this column to toggle touching a file. Touching a file manually selects it for compilation during the next build. Click the Touch icon to sort files by touch status.
File		Displays a hierarchical view of the file and group names used by the project. Click the column title to sort files by name. Double-click a file to open it. Use the hierarchical controls to display and hide group contents.
Code		Displays the size, in bytes or kilobytes, of the compiled executable object code for files and groups. Click the column title to sort files by code size.
Data		Displays the size, in bytes or kilobytes, of non-executable data in the object code for files in the project. Click the column title to sort files by data size.
Target		Indicates whether each file belongs to the current build target. Click in this column to toggle inclusion status. Click the Target icon to sort files by inclusion status. The Target column appears only when the project has more than one build target.
Debug		Displays debugging status. Click in this column to toggle generation of debugging information for a file or group. Click the Debug icon to sort files by debugging status.
Checkout Status		Displays icons representing the current file status in a version-control system. The Checkout Status column appears only when the project uses a version-control system to manage files.
Interfaces		Click to display a list of files inside a group or a list of #include files inside a source file. Choose a file to open it.
Sort Order		Click to toggle sorting between ascending and descending order for the active column. The icon indicates the current sort order.

## **Viewing a File Path**

To distinguish between two files that have identical names but reside in different folders, examine the file path.

To view the complete file path of a file, perform the task explained in [Table 4.3](#).

**Table 4.3 Viewing a file path**

<b>On this host...</b>	<b>Do this...</b>
Windows	Right-click the filename and select <b>Open in Windows Explorer</b>
Macintosh	Control-click the filename and select <b>File Path</b> .
Solaris	Click and hold on the filename, then select <b>File Path</b> .
Linux	Click and hold on the filename, then select <b>File Path</b> .

The File Path submenu displays the path to the file.

## **Designs View**

The Designs view presents information about designs for projects that use Rapid Application Development (RAD). Layout files typically appear in the Designs view. These files describe the graphical user interface for a computer program.

[Table 4.4](#) explains the items in the Files view.

**Table 4.4 Designs view—items**

<b>Item</b>	<b>Explanation</b>
Object	Displays all design objects available in the current build target of the project.
Class	Displays the class type of each object.

## **Link Order View**

The Link Order view presents information about the order in which the IDE links project files. Manipulate the files in this view to change the link order. For example, if

file B depends on file A in order to function, move file B below file A in the Link Order view.

[Table 4.5](#) explains the items in the Link Order view.

**Table 4.5** Link Order view—items

Item	Explanation
Synchronize Modification Dates	To update the modification dates of the files stored in a project file, click the checkmark icon.  Use the Synchronize Modification Dates command to update files modified outside of the CodeWarrior IDE, perhaps by a third-party editor which cannot notify the CodeWarrior IDE of the changes.
Synchronize Status	To update version-control status information, click the Pencil icon.

## Targets View

The Targets view presents information about the build targets in a project. Use this view to create, manage, or remove build targets. Different build targets can store different IDE settings. For example, two build targets can handle the same project. One build target handles debugging the software, while the other build target handles building the software for final release.

[Table 4.6](#) explains the items in the Targets view.

**Table 4.6** Targets view—items

Item	Explanation
Targets	Displays all build targets and subprojects that the IDE processes to create a binary file. These icons denote build-target status: <ul style="list-style-type: none"><li>•  active build target</li><li>•  inactive build target</li><li>•  active build target with RAD designs</li><li>•  inactive build target with RAD designs</li></ul>
Link	Indicates the dependencies between build targets and subprojects.

# File, Group, Layout, and Target Management

Use these tasks to manage files, groups, layouts, and targets:

- Create an item.
- Delete an item.
- Move an item.
- Rename an item.
- Touch an item.
- Manage items.
- Set default items.
- Configure item settings.

---

## Removing Files/Groups/Layouts/Targets

The **Remove** command deletes files, groups, layouts, and build targets from the Project window. Removing files from the **Files** tab removes them from the project itself and from all build targets that use the files. Removing a file from the **Link Order**, **Segments**, or **Overlays** tab only removes the file from the current build target.

### Removing files/groups/layouts/targets from a project

1. Click the **Files**, **Designs**, or **Targets** tab in the Project window.
2. Select the item to remove.
3. Remove the selected item from the project, as explained in [Table 4.7](#).

**Table 4.7 Removing a selected item from a project**

On this host...	Do this...
Windows	Select <b>Edit &gt; Delete</b> .
Macintosh	Select <b>Edit &gt; Clear</b> .

## Project Window

*File, Group, Layout, and Target Management*

---

**Table 4.7 Removing a selected item from a project (continued)**

On this host...	Do this...
Solaris	Select <b>Edit &gt; Clear</b> .
Linux	Select <b>Edit &gt; Clear</b> .

The IDE removes the selected item from the project. For deleted files, the IDE updates all build targets that formerly used the file. For deleted build targets, the IDE deletes build-target information and leaves files intact.

## Removing files from a build target

1. Click the **Link Order, Segments**, or **Overlays** tab in the Project window.
2. Select the item to remove.
3. Remove the selected item from the active build target, as explained in [Table 4.8](#).

**Table 4.8 Removing a selected item from the active build target**

On this host...	Do this...
Windows	Select <b>Edit &gt; Delete</b> .
Macintosh	Select <b>Edit &gt; Clear</b> .
Solaris	Select <b>Edit &gt; Clear</b> .
Linux	Select <b>Edit &gt; Clear</b> .

The IDE removes the file from the build target, but leaves the file itself intact. The file can be re-assigned to other build targets in the project.

---

## Moving Files/Groups/Layouts/Targets

Reposition files, groups, layouts, or build targets within the **Files**, **Design**, **Link Order**, or **Targets** views with the pointer.

1. Select one or more files, groups, layouts, or build targets to move with the pointer.

2. Drag the selected items to a new position in the current view, using the focus bar as a guide.
3. Release the mouse button.

The IDE repositions the selected files, groups, layouts, or build targets to the new location.

---

<b>NOTE</b>	In the <b>Link Order</b> view, repositioning files changes the link order used by the <b>Make</b> command to build the final executable file.
-------------	---

---

## **Renaming Files/Groups/Targets**

The **Rename** command renames files, groups, or build targets in the project.

### **Rename files**

1. Open the file to rename.
2. Choose **File > Save As**.
3. Type a new filename in the **Name** text box.
4. Click **Save**.

The IDE saves the file under the new name. The new filename appears in the Project window. Subsequent modifications affect the renamed file, leaving the original file intact.

### **Rename one or more groups**

1. Click the **Files** tab in the Project window.
2. Select the group(s) to rename.
3. Press the **Enter** key.
4. Type a new name into the **Enter Group Name** text box of the **Rename Group** window.

## Project Window

*File, Group, Layout, and Target Management*

---

5. Click **OK**.

The IDE renames the group. For selections of more than one group, the **Rename Group** window appears for each group.

## Rename build targets

1. Click the **Targets** tab in the Project window.
2. Choose **Edit > targetname Settings**.
3. Select **Target Settings** in the **Target Settings Panels** list.
4. Type a new name in the **Target Name** text box.
5. Click **Save**.

The Project window displays the new build-target name.

---

## Touching Files and Groups

The **Touch** command manually selects source files or groups for compilation during the next **Bring Up To Date**, **Make**, **Run**, or **Debug** operation. A red check mark in the **Touch** column of the Project window indicates a touched file.

1. Click the **Files** tab in the Project window.
2. Touch a source file or group for compilation.
  - Click the **Touch** column next to the file or group name.
  - OR
  - Choose **Touch** from the **Interface** menu for the file or group.

A red check mark appears in the Touch column next to the file or group name.

## Touch all project files for recompiling

1. Perform the task explained in [Table 4.9 on page 55](#).

**Table 4.9 Touching all project files for recompiling**

On this host...	Do this...
Windows	Alt-click the Touch column.
Macintosh	Option-click the Touch column.
Solaris	Alt-click the Touch column.
Linux	Alt-click the Touch column.

2. Red check marks appear next to all files and groups.

---

## Untouching Files and Groups

The **Untouch** command manually excludes source files or groups from compilation during the next **Bring Up To Date**, **Make**, **Run**, or **Debug** operation.

1. Click the **Files** tab in the Project window.
2. Untouch a source file or group to remove it from the compilation list.
  - Click the red check mark in the **Touch** column next to the file or group name.  
OR
  - Choose **Untouch** from the **Interface** menu for the file or group.

The red check mark disappears from the **Touch** column next to the file or group name.

## Untouch all project files

1. Perform the task explained in [Table 4.10](#).

**Table 4.10 Untouching all project files**

On this host...	Do this...
Windows	Alt-click a red checkmark in the Touch column.
Macintosh	Option-click a red checkmark in the Touch column.
Solaris	Alt-click a red checkmark in the Touch column.
Linux	Alt-click a red checkmark in the Touch column.

2. The red checkmarks next to all files and groups disappear.

## Build-Target Management

These tasks help you manage build targets:

- Create a build target.
- Remove a build target.
- Set the default build target.
- Rename a build target.
- Configure build-target settings.

---

### Creating Build Targets

The **Create Target** command adds new build targets to a project.

1. Open the **Project** window.
2. Click the **Targets** tab in the Project window.
3. Choose **Project > Create Target**.
4. Type a name in the **Name** text box of the **New Target** window.
5. Select the **Empty target** or **Clone Existing Target** radio button as desired.
  - **Empty Target**—create a new build target from scratch.
  - **Clone Existing Target**—duplicate an existing build target in the **New Target** window.
6. Click **OK**.

The IDE adds the new build target to the project.

---

### Removing Build Targets from a Project

The **Remove** command deletes unneeded build targets from the Project window.

1. Click the **Targets** tab in the Project window.
2. Select the item to remove.
3. Remove the selected item from the active build target, as explained in [Table 4.11](#).

**Table 4.11 Removing the selected item from the active build target**

On this host...	Do this...
Windows	Select <b>Edit &gt; Delete</b> .
Macintosh	Select <b>Edit &gt; Clear</b> .
Solaris	Select <b>Edit &gt; Clear</b> .
Linux	Select <b>Edit &gt; Clear</b> .

The IDE removes the file from the build target, but leaves the file itself intact. The file can be re-assigned to other build targets in the project.

---

## Setting the Default Build Target

The CodeWarrior Project Manager can handle up to 255 build targets in a single project. One build target must be defined as the default target when more than one project is open. The default target is the target affected by project commands, such as **Make** and **Run**.

## The Project menu

1. Choose **Project > Set Default Target > buildtarget**.
2. A checkmark indicates the default target.

## Using the Project window toolbar

1. Enable the **Project** window.
  2. Choose the build-target name from the **Current Target** pop-up menu.
- The build-target name appears in the pop-up menu.

## The Targets view

1. Enable the **Project** window.
2. Click the **Targets** tab.
3. Click the desired build-target icon.

The icon changes to indicate that the build target is now the default.

---

## Renaming Build Targets

The **Rename** command renames build targets in a project.

1. Click the **Targets** tab in the Project window.
2. Choose **Edit > *targetname* Settings**.
3. Select **Target Settings** in the **Target Settings Panels** list.
4. Type a new name in the **Target Name** text box.
5. Save the new name, as explained in [Table 4.12](#).

**Table 4.12 Saving a new name for a build target**

On this host...	Do this...
Windows	Click <b>OK</b> .
Macintosh	Click <b>Save</b> .
Solaris	Click <b>Save</b> .
Linux	Click <b>Save</b> .

The new build-target name appears in the Project window.

---

## Configuring Build Target Settings

The **Target Settings** panel options determine:

- The compiler used to process the project and produce object code
- The linker used to combine object code and produce a binary file
- The pre-linker and post-linker options that further process the object code
- The name assigned to a build target

Follow these steps to configure build-target settings.

1. Choose **Edit > targetname Settings**.
2. Select **Target Settings** from the **Target Setting Panels** list.
3. Specify target options as desired.
4. Save the new options, as explained in [Table 4.13](#).

**Table 4.13 Saving the build-target settings**

<b>On this host...</b>	<b>Do this...</b>
Windows	Click <b>OK</b> .
Macintosh	Click <b>Save</b> .
Solaris	Click <b>Save</b> .
Linux	Click <b>Save</b> .

---

**NOTE**      The panels available in the **Target Settings Panels** list update to reflect the choices in the **Target Settings** panel.

---



# Working with Files

---

This chapter explains how to work with files in the CodeWarrior™ IDE. Most computer programs use these file types:

- Text files—files that contain any type of text. Example text files include Read Me files and source files.
- Source files—files that contain source code only. Example source files include C++ files and Java files.

## Managing Files

These tasks manage files:

- Create a new file.
- Open an existing file.
- Save a file.
- Close a file.
- Print a file.
- Revert a file to a previously saved state.

---

### Creating Text Files (Windows)

The **New** command opens a window from which you create new text files. You can use new text files as source files in a project or as plain-text files.

1. Select **File > New**.  
The **New** window appears.
2. Click the **File** tab in the New window.
3. Select **Text File** in the list.

4. Type a filename into the **File name** text box.
5. Click **Set** to specify the location to save the new file.
6. Click **OK**.

The IDE creates the new text file and displays its contents in a new editor window.

---

**TIP** Use the **Customize IDE Commands** window to add the **New Text File** menu command to the **File** menu. Adding this menu command reduces the process of creating a new text file to one step: select **File > New Text File**. See “[Customizing the IDE](#)” on page 327 for more information about using the Customize IDE Commands window.

---

---

## **Creating Text Files (Macintosh, Solaris, Linux)**

The **New Text File** command creates new text files. You can use new text files as source files in a project or as plain-text files.

Select **File > New Text File** to create a new text file. The IDE creates the new text file and displays its contents in a new editor window.

---

## **Opening Source Files**

The **Open** command opens one or more editable source files at a time. Each open file appears in its own editor window.

---

**NOTE** The CodeWarrior editor cannot open files that prohibit editing. For example, the editor cannot open library files.

---

## **From the File menu**

1. Choose **File > Open**.
2. Windows: Use the **Files of type** pop-up menu to select **All Files**.

3. Select a file.

4. Click **Open**.

The IDE displays the file in an editor window.

## From the Project window

1. Perform one of these:

- Double-click a filename in the **Files** tab of the Project window, or
- Select an interface filename from the Interface menu.

2. The IDE finds, opens, and displays the selected source file in an editor window.

## From an editor window

1. Select an interface filename from the Interface menu.

2. The IDE selects, opens, and displays the source file in an editor window.

---

<b>NOTE</b>	The menu does not show files that lack source code or are not yet compiled.
-------------	---

---

## Using Find and Open Files

1. Select text in an editor window containing the name of an interface file.

2. Choose **File > Find and Open File**.

The IDE finds, opens, and displays in an editor window the source file matching the text selection.

## To open a recent file or project

1. Choose **File > Open Recent > *recentfilename | recentprojectname***.

2. The IDE finds and opens the selected source file or project.

## Saving Files

Use the **Save** command to save source files to ensure their continued existence between development sessions.

1. Choose **File > Save**.

---

**NOTE** If the file has no title, a save dialog appears. Type a filename and specify a location for the file, then click **Save**.

---

2. The IDE saves the file.

---

## Saving All Modified Files

Use the **Save All** command to save the contents of all modified files. This command is useful for saving all files at the same time, rather than saving each file individually.

1. Save all currently opened and modified files, as explained in [Table 5.1](#).

**Table 5.1 Saving all currently opened and modified files**

On this host...	Do this...
Windows	Select <b>File &gt; Save All</b> .
Macintosh	While pressing Option, select <b>File &gt; Save All</b> .
Solaris	While pressing Alt, select <b>File &gt; Save All</b> .
Linux	While pressing Alt, select <b>File &gt; Save All</b> .

2. The IDE saves the files.

## Saving File Copies

Use the **Save a Copy As** command to save a back-up copy of a project or file before modifying the original. Working on a copy of the original file provides a way to return to the original copy should modifications fail.

1. Choose **File > Save A Copy As**.
2. Type a new filename in the **Name** text box.
3. Click **Save**.

The IDE creates a copy of the file under the new name, leaving the original file unchanged.

---

## Closing Files

The **Close** command closes open source files when you finish working on them. Close editor windows to close a file.

1. Select an editor window to close.
2. Close the file window.
  - Choose **File > Close**, or
  - Click the close box.

---

<b>NOTE</b>	The IDE displays an alert if the file is modified. The alert asks whether to save changes to the file.
-------------	--

---

The IDE closes the file window.

---

## Closing All Files

The **Close All** command closes all currently open files. This command is useful for closing all files at the same time, rather than closing each file individually.

1. Close all currently open files, as explained in [Table 5.2](#).

**Table 5.2 Closing all currently open files**

On this host...	Do this...
Windows	Select <b>Window &gt; Close All</b> or <b>Window &gt; Close All Editor Windows</b> .
Macintosh	While pressing Option, select <b>File &gt; Close All</b> .
Solaris	While pressing Alt, select <b>File &gt; Close All</b> .
Linux	While pressing Alt, select <b>File &gt; Close All</b> .

2. The IDE closes the files.

---

## Printing Source Files

The **Print** command prints the entire contents of a selected file window.

1. Activate the desired editor window to print.
2. Choose **File > Print**.
3. Set print options in the **Print** dialog.
4. Print the file, as explained in [Table 5.3](#).

**Table 5.3 Printing a source file**

On this host...	Do this...
Windows	Click <b>OK</b> .
Macintosh	Click <b>Print</b> .
Solaris	Click <b>Print</b> .
Linux	Click <b>Print</b> .

The IDE prints the selected file.

<b>NOTE</b>	Use the same process to print the contents of a window, such as a Project window.
-------------	---

---

## Printing Source-File Selections

The **Print** command prints the currently selected contents in an editor window.

1. Activate the desired editor window to print.
2. Select the portion of text to print.
3. Choose **File > Print**.
4. Set print options in the **Print** dialog.
5. Print the file, as explained in [Table 5.4](#).

**Table 5.4** Printing a source-file selection

On this host...	Do this...
Windows	Click <b>OK</b> .
Macintosh	Click <b>Print</b> .
Solaris	Click <b>Print</b> .
Linux	Click <b>Print</b> .

The IDE prints the selected text in the file.

---

## Reverting Files

Use the **Revert** command to replace the current file with its previously saved version.

1. Choose **File > Revert**.
2. Click **OK** in the **Revert changes to file** dialog.

The IDE replaces the file with its previous version. The replaced file appears in the editor window.

## **Working with Files**

### *Managing Files*

---

# Dockable Windows

---

This chapter explains how to work with dockable windows in the Windows-hosted CodeWarrior™ IDE. Use dockable windows to do these tasks:

- Organize—attach, or *dock*, various windows to the edges of the screen for quick access.
- Group—dock windows of the same type to create a single window with multiple tabs, where each tab represents one of the original docked windows.

---

<b>NOTE</b>	The dockable windows feature is available in Multiple Document Interface (MDI) mode only. This feature is not available in Floating Document Interface (FDI) mode. Toggle the <a href="#">Use Multiple Document Interface</a> option in the <a href="#">IDE Extras</a> preference panel to change between these two modes.
-------------	--

---

This chapter contains these sections:

- [“About Dockable Windows” on page 69](#)
- [“Working with Dockable Windows” on page 71](#)
- [“Dock Bars” on page 76](#)

## About Dockable Windows

You can dock certain windows to the edges of the main frame window of the IDE. [Table 6.1 on page 70](#) explains possible states for dockable windows. [Figure 6.1 on page 70](#) shows the different window states.

In MDI mode, the IDE occupies a main window frame, or *client area*. IDE windows normally appear within this client area as you work. These windows are called *child windows* of the IDE’s client area.

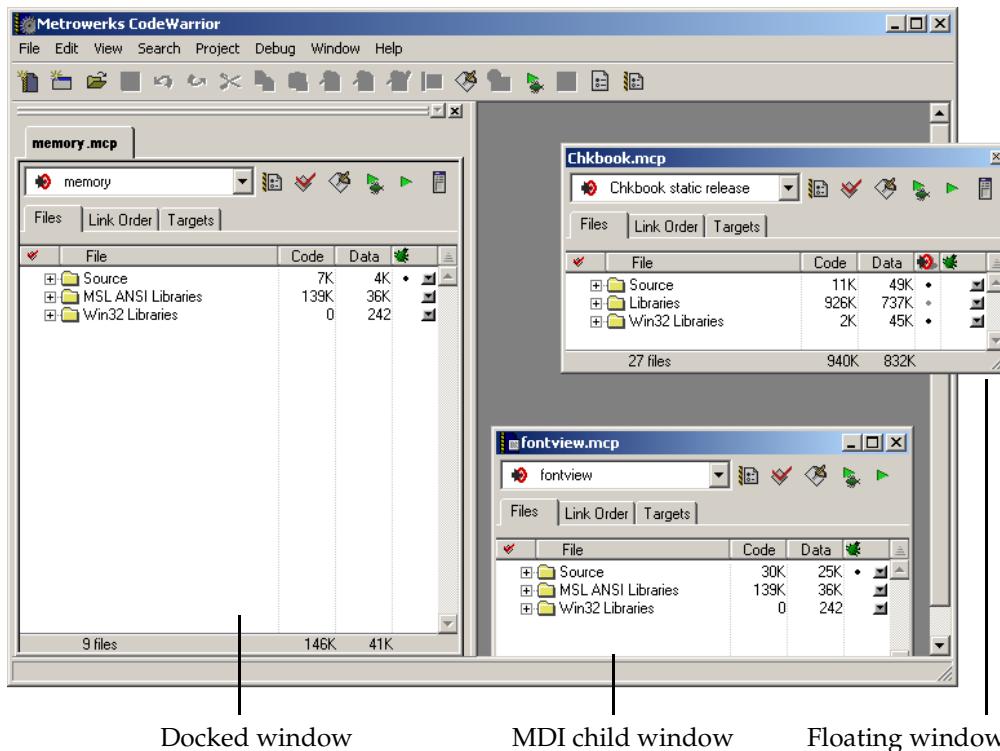
## Dockable Windows

[About Dockable Windows](#)

**Table 6.1 Window states**

State	Characteristics
Docked	<ul style="list-style-type: none"><li>Attached to the left, right, top, or bottom edge of the client area</li><li>restricted to the client area</li><li>resizable</li><li>has a dock bar instead of a title bar</li></ul>
Floating	<ul style="list-style-type: none"><li>Rests above all docked windows and MDI child windows</li><li>movable outside the client area, like a floating palette</li><li>has a thin title bar</li><li>does not have Minimize or Maximize buttons</li></ul>
MDI Child	<ul style="list-style-type: none"><li>Normal child window of the client area, when running in MDI mode</li><li>restricted to the client area</li></ul>

**Figure 6.1 Window states**



[Table 6.2](#) explains the difference between dockable windows and non-dockable windows. In this table, the term *non-modal* refers to a window that does not require your attention before allowing the IDE to proceed with other operations.

**Table 6.2 Differences between dockable and non-dockable windows**

Window Type	Required Criteria	Sample Windows
Dockable	All of these: <ul style="list-style-type: none"><li>• non-modal</li><li>• resizable</li><li>• maximizable</li></ul>	<ul style="list-style-type: none"><li>• Thread</li><li>• Project</li><li>• Component Catalog</li></ul>
Non-dockable	Any of these: <ul style="list-style-type: none"><li>• modal</li><li>• non-resizable</li><li>• non-maximizable</li></ul>	<ul style="list-style-type: none"><li>• IDE Preferences</li><li>• Find</li><li>• About Box</li></ul>

---

**NOTE** The default setting for project windows is to dock to an edge of the client area. You can undock these windows.

Compound windows that have more than one pane dock as a group. You cannot separately dock individual panes from these windows. For example, you can dock the Thread Window, but you cannot dock the Stack Crawl pane separately from the Thread Window.

---

## Working with Dockable Windows

You can dock windows in one of two ways:

- dragging a floating window to a docking position
- using a contextual menu to dock a window

You can resize docked windows and undock them to floating windows or MDI child windows.

This section explains how to perform tasks with dockable windows.

---

## Docking a Window By Using a Contextual Menu

Use a contextual menu to dock a floating window or MDI child window to one of the four edges of the client area.

1. Right-click the window title bar.  
A contextual menu appears.
2. Choose **Docked** from the contextual menu.

---

**NOTE**

The **Docked** command appears in the contextual menu for dockable windows only.

---

The window docks to an edge of the client area. You can resize the docked window or move it to a different edge of the client area.

---

## Docking a Window By Using Drag and Drop

You can drag a docked window or a floating window to one of the four edges of the client area to dock it.

1. Drag the window to one edge of the client area.  
Drag a floating window by its title bar. Drag a docked window by its dock bar.
2. A window outline appears near the client-area edge, showing the final position after you release the window.  
Use the outline as a visual cue that the IDE will dock the window. If an outline does not appear, you cannot dock the window.
3. Release the window to dock it to the edge.  
The window appears in the position indicated by the window outline.

## Docking Windows of the Same Kind

You can dock two or more windows of the same kind inside a single docked window. In this arrangement, tabs inside the single docked window represent each of the original docked windows. You can undock each tab individually from the single docked window.

1. Dock the first of two or more windows of the same kind to an edge of the client area.
2. Dock the second window to the same edge as the first window.

Use the window outline that appears as a visual cue that the IDE will dock the second window to the same edge as the first window.

3. Dock subsequent windows to the same edge as the first window.
- Each additional docked window appears as a tab inside the first docked window. Click a tab to view its contents. The frontmost tab appears in bold font.

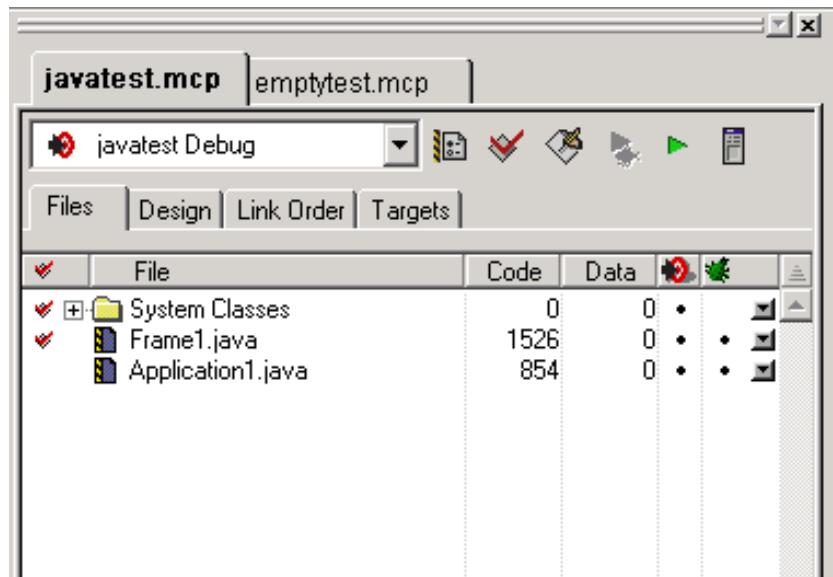
[Figure 6.2](#) shows two projects represented as tabs in a single docked window.

## Dockable Windows

### Working with Dockable Windows

---

**Figure 6.2 Two projects in a single docked window**



---

## Undocking a Window

Use a contextual menu to undock a window from an edge of the client area to a floating window or MDI child window.

1. Right-click the tab inside the docked window that represents the window you want to undock.

A contextual menu appears.

**Figure 6.3 Contextual menu**



2. Choose **Floating** or **MDI Child** from the contextual menu.
  - Floating—undock the window so that it becomes a floating window
  - MDI child—undock the window so that it becomes an MDI child window of the client area

The window undocks and becomes the chosen window type.

Alternately, double-click the tab to undock the corresponding window to a floating window.

---

## Floating a Window

Use a contextual menu to float a docked window or MDI child window.

1. Right-click the tab in the docked window or the title bar of the MDI child window.

A contextual menu appears.
2. Choose **Floating** from the contextual menu.

---

<b>NOTE</b>	The <b>Floating</b> command appears in the contextual menu for floatable windows only.
-------------	--

---

The window becomes a floating window (that you can drag outside the client area).

Alternately, double-click the tab in a docked window to float its corresponding window.

---

## Unfloating a Window

Use a contextual menu to dock a floating window or make it an MDI child window.

1. Right-click the title bar of the floating window.

A contextual menu appears.
2. Choose **Docked** or **MDI Child** from the contextual menu.
  - Docked—dock the floating window

## Dockable Windows

### Dock Bars

---

- MDI child—unfloat the window so that it becomes an MDI child window

The window unfloats and becomes the chosen window type.

Alternately, drag the floating window to an edge of the client area to dock it.

---

## Making a Window an MDI Child

Use a contextual menu to make a docked window or floating window an MDI child window.

1. Right-click the tab in the docked window or the title bar of the floating window.

A contextual menu appears.

2. Choose **MDI Child** from the contextual menu.

The docked window or floating window becomes an MDI child window.

---

## Suppressing Dockable Windows

Suppress dockable windows to drag a window to any location onscreen without docking it to an edge of the client area.

1. Hold down the Ctrl key while dragging or floating an MDI child window.

The thin window outline that normally indicates docked-window placement becomes a heavy window outline. Use this heavy outline as a visual cue that the IDE suppresses dockable windows.

2. Release the window at its final position.

The window appears in the position indicated by the heavy window outline.

3. Release the Ctrl key.

## Dock Bars

A docked window has a dock bar instead of a title bar. Use the dock bar to perform these tasks:

- move the docked window to a different edge of the client area

- collapse or expand view of the docked window
- close the docked window

[Figure 6.4](#) shows a dock bar.

**Figure 6.4 Dock Bar**



---

## **Collapsing a Docked Window**

If two or more distinct docked windows occupy the same edge of the client area, you can collapse one docked window to view more contents of the other docked windows.

1. Dock two or more windows to the same edge of the client area.  
The windows' contents must appear in separate docked windows, not as tabs in a single docked window.
2. Click the collapse button  on the dock bar of the docked window that you want to collapse.
3. The docked window collapses to hide its contents.

---

## **Expanding a Docked Window**

If you previously collapsed a docked window, you can expand it and view its contents once again.

1. Click the expand button on the dock bar: 
2. The docked window expands to restore its original view.

---

## **Moving a Docked Window**

Use the gripper in a docked window's dock bar to move the docked window to a different edge of the client area.

1. Drag the docked window by the gripper in its dock bar: 
2. Release the docked window at its new position.

---

## Closing a Docked Window

Close a docked window directly from its dock bar.

1. Click the close button on the dock bar: 

2. The docked window closes.

Re-opening the window later restores its docked position.

## **Dockable Windows**

### *Dock Bars*

---

# Workspaces

---

This chapter explains how to work with workspaces in the CodeWarrior™ IDE. Use workspaces to do these tasks:

- Organize—save the state of all windows onscreen for later reuse
- Migrate across computers—transfer your workspace from one computer to another

This chapter contains these sections:

- [“About Workspaces” on page 81](#)
- [“Using Workspaces” on page 81](#)

## About Workspaces

A *workspace* stores information about the current state of the IDE. This information consists of the size, location, and the docked state (Windows) of IDE windows. If you save a workspace during an active debugging session, the workspace also stores information about the state of debugging windows.

The IDE can use a *default workspace*, or it can use a workspace that you create. The IDE works with one workspace at a time. You can save and re-apply a workspace from one IDE session to the next.

## Using Workspaces

You use menu commands to perform these workspace tasks:

- save a new workspace
- open an existing workspace
- close the current workspace

## Using the Default Workspace

Use the default workspace to preserve IDE state from one session to the next. The IDE saves and restores the default workspace automatically.

1. Choose **Edit > Preferences**.

The IDE Preferences window opens.

2. Select **IDE Extras** in the **IDE Preference Panels** list.

The IDE Extras preference panel appears.

3. Configure the [Use default workspace](#) option.

- Selected—the IDE saves its state at the time you quit, then restores that state the next time you launch the IDE
- Cleared—the IDE always launches with the same default state: no windows visible

---

## Saving a Workspace

Save a workspace to store information about the current state of onscreen windows, recent items, and debugging.

1. Arrange your workspace.

Move windows to your favorite positions and start or finish a debugging session.

2. Choose **File > Save Workspace**.

A **Save** dialog box appears.

3. Enter a name for the current workspace

---

<b>NOTE</b>	Add the extension .cww to the end of the workspace name, for example, <code>myworkspace.cww</code> . This extension helps you readily identify the workspace file. Furthermore, the Windows-hosted IDE requires this extension to recognize the file as a CodeWarrior workspace.
-------------	--

---

4. Save the workspace to a location on your hard disk.

The IDE now uses your saved workspace. In subsequent programming sessions, you can open the workspace and apply its settings to the IDE.

---

## **Opening a Workspace**

Open a workspace to apply its settings to the IDE.

1. Choose **File > Open Workspace**.

An **Open** dialog box appears.

2. Open the workspace.

Use this dialog box to navigate your hard disk and select a workspace file. These files end in the .cww extension.

The IDE opens the selected workspace and applies its settings.

---

## **Saving a Copy of a Workspace**

Save a copy of a current workspace to save an existing workspace under a different name.

1. Open an existing workspace.

2. Choose **File > Save Workspace As**.

A **Save As** dialog box appears.

3. Enter a name for the copy of the current workspace

---

<b>NOTE</b>	Add the extension .cww to the end of the workspace name, for example, <code>myworkspace.cww</code> . This extension helps you readily identify the workspace file. Furthermore, the Windows-hosted IDE requires this extension to recognize the file as a CodeWarrior workspace.
-------------	--

---

4. Save the workspace to a location on your hard disk.
-

The IDE saves a copy of the current workspace under the name you specified.

---

## Closing a Workspace

Close the current workspace after you finish working with it.

1. Choose **File > Close Workspace**.
2. The IDE closes the current workspace.

---

<b>NOTE</b>	You cannot close the default workspace, however, the <b>IDE Extras</b> preference panel contains an option that determines whether the IDE uses the default workspace.
-------------	--

---

You can now open a different workspace or quit the IDE.

---

## Opening a Recent Workspace

You can have the IDE display recently used workspaces in the **Open Recent** submenu. The **IDE Extras** preference panel contains an option that determines the number of recent workspaces that the IDE displays.

1. Choose **File > Open Recent**.  
A submenu appears. This submenu lists recently used workspaces. A checkmark appears next to the active workspace.
2. Choose a recent workspace from the Open Recent submenu.  
The IDE applies the workspace that you select.

# Creating Console Applications

---

This chapter explains how to work with console applications in the CodeWarrior<sup>TM</sup> IDE. Console applications provide these benefits to novice programmers:

- Simplicity—console applications are computer programs that use a simple text-mode interface. The simplicity of console-mode applications free novice programmers to learn a programming language without having to learn graphical user interface programming at the same time.
- Foundation—understanding console applications provides the basis for more advanced computer programming. Advanced programmers readily understand console applications.

Read this chapter to learn more about typical tasks for working with console applications.

This chapter contains these sections:

- [About Console Applications](#)
- [Creating Console Applications](#)

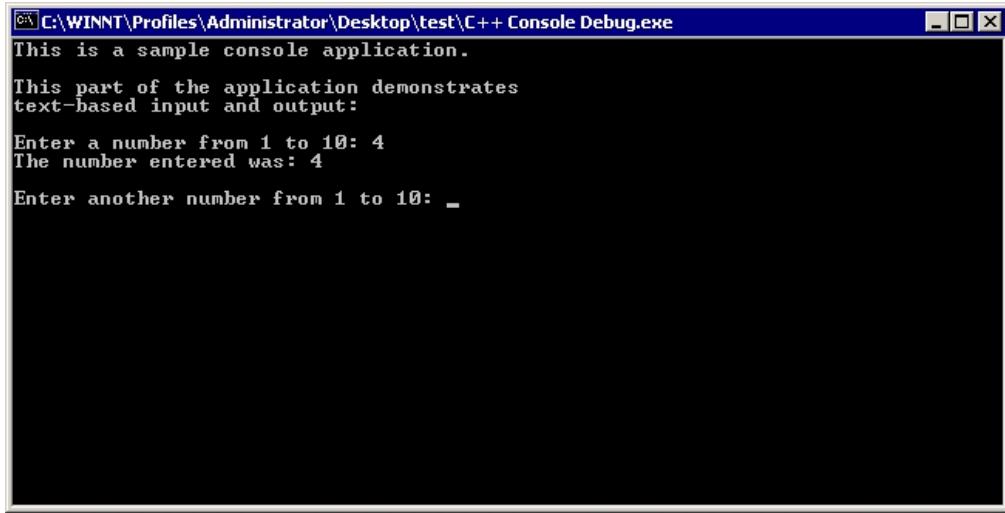
## About Console Applications

A console application is a simple, text-based computer program. Console applications do not usually employ a graphical user interface (GUI). Instead, the applications rely on plain-text input and output to complete tasks.

Console applications are ideal for novice programmers. The applications are easier to program because they lack a GUI. If problems arise, the programmer can use text-based feedback together with the debugger to correct problems.

[Figure 8.1 on page 86](#) shows output from a sample console application.

**Figure 8.1 Console application**



## Creating Console Applications

Create a console application to begin working with a text-based computer program. The CodeWarrior IDE provides pre-configured project stationery for creating console applications. Project stationery simplifies the project-creation process. This section explains how to create a console application.

---

### Creating a Console Application

Use the **New** command to create a new project. The project stores information about the files in the console application.

1. Choose **File > New**.
2. Click the **Project** tab.
3. Select a project stationery file.

For example, select **Win32 C Stationery** to create a C console application.

4. Enter a project name in the **Project name** field and add the .mcp extension.

For example, name the project `test.mcp`.

5. Click **Set**.

Save the project in the desired location.

6. Click **OK**.

The **New Project** window appears.

7. Select a specific stationery file.

For example, expand **Win32 Console App** and select **C Console App**.

8. Click **OK**.

The IDE creates a console application from the selected stationery. The Project window for the console application appears.

9. Expand the **Sources** group.

This group contains placeholder source files.

10. Remove placeholder source files.

For example, select `main.c` and choose **Edit > Delete**.

11. Create a new source file, as explained in [Table 8.1](#).

**Table 8.1 Creating a new source file**

On this host...	Do this...
Windows	Press Ctrl-N.
Macintosh	Press Command-N.
Solaris	Press Meta-N.
Linux	Press Ctrl-N.

12. Enter source code.

For example, enter this source code shown in [Listing 8.1 on page 88](#).

**Listing 8.1 Sample source code**

---

```
/* A minimal Win32 console application */
#include <stdio.h>
int main( void )
{
    printf("Hello World!");
    return 0;
}
```

---

13. Save the source file, as explained in [Table 8.2](#).

**Table 8.2 Saving the source file**

On this host...	Do this...
Windows	Press Ctrl-S.
Macintosh	Press Command-S.
Solaris	Press Meta-S.
Linux	Press Ctrl-S.

Enter a name for the source code. For example, enter `Hello.c`. Then click **Save**.

14. Choose **Project > Add Hello.c to Project**.

The **Add Files** window appears.

15. Add the file to all build targets in the project.

Select all checkboxes to add the file to all build targets, then click **OK**.

16. Drag the source file inside the **Sources** group.

17. Choose **Project > Run**.

The IDE compiles, links, then runs the console application.



# Editor

---

This section contains these chapters:

- [The CodeWarrior Editor](#)
- [Editing Source Code](#)
- [Navigating Source Code](#)
- [Finding and Replacing Text](#)



# The CodeWarrior Editor

---

This chapter explains how to work with the editor in the CodeWarrior™ IDE. Use the editor to perform these tasks:

- Manage text files—the editor includes common word-processing features for creating and editing text files. Sample text files include Read Me files and release notes.
- Manage source files—the editor includes additional features for creating and editing source files. The IDE processes source files to produce a program.

This chapter contains these sections:

- [“Editor Window” on page 91](#)
- [“Editor Toolbar” on page 93](#)
- [“Other Editor Window Components” on page 97](#)

## Editor Window

Use the editor window to create and manage text files or source files. The window contains these major parts:

- Editor toolbar
- Text-editing area
- Line and column indicator
- Pane splitter controls

[Figure 9.1 on page 92](#) shows the editor window. [Table 9.1 on page 92](#) explains the items in the editor window.

## The CodeWarrior Editor

### Editor Window

Figure 9.1 Editor window

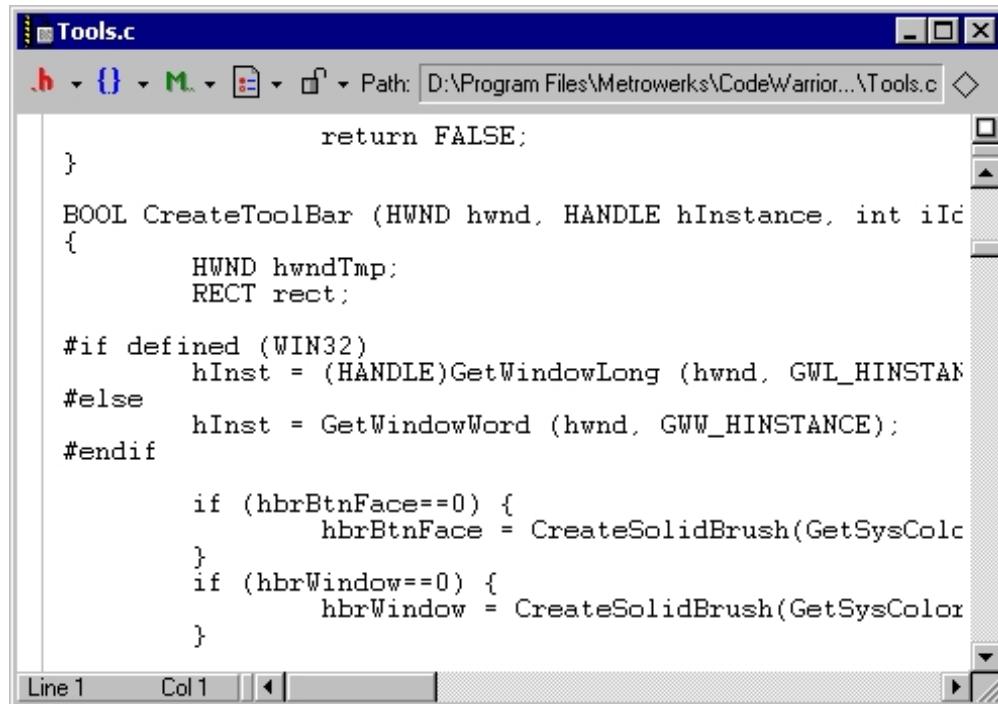


Table 9.1 Editor window—items

Item	Icon	Explanation
<a href="#">Interfaces Menu</a>		Displays a list of referenced interface files or header files for the source file.
<a href="#">Functions Menu</a>		Displays a list of functions defined in the source file.
<a href="#">Markers Menu</a>		Displays a list of markers defined in the file.
<a href="#">Document Settings Menu</a>		Displays file-format options and a syntax-coloring toggle.

**Table 9.1 Editor window—items (*continued*)**

Item	Icon	Explanation
<a href="#">Version Control System Menu</a>		Displays a list of available Version Control System (VCS) commands. Choose a command to apply to the source file.
<a href="#">Path Caption</a>		Displays the complete path to the file.
<a href="#">File Modification Icon</a>	 	This icon indicates an unchanged file since the last save. This icon indicates a file with modifications not yet saved.
<a href="#">Breakpoints Column</a>		Displays breakpoints for the file.
<a href="#">Text Editing Area</a>		Shows the text or source-code content of the file.
<a href="#">Line and Column Indicator</a>	Line 1 Col 1	Displays the current line and column number of the text-insertion cursor
<a href="#">Pane Splitter Controls</a>		Drag to split the window into panes.

## Editor Toolbar

Use the editor toolbar to complete these tasks:

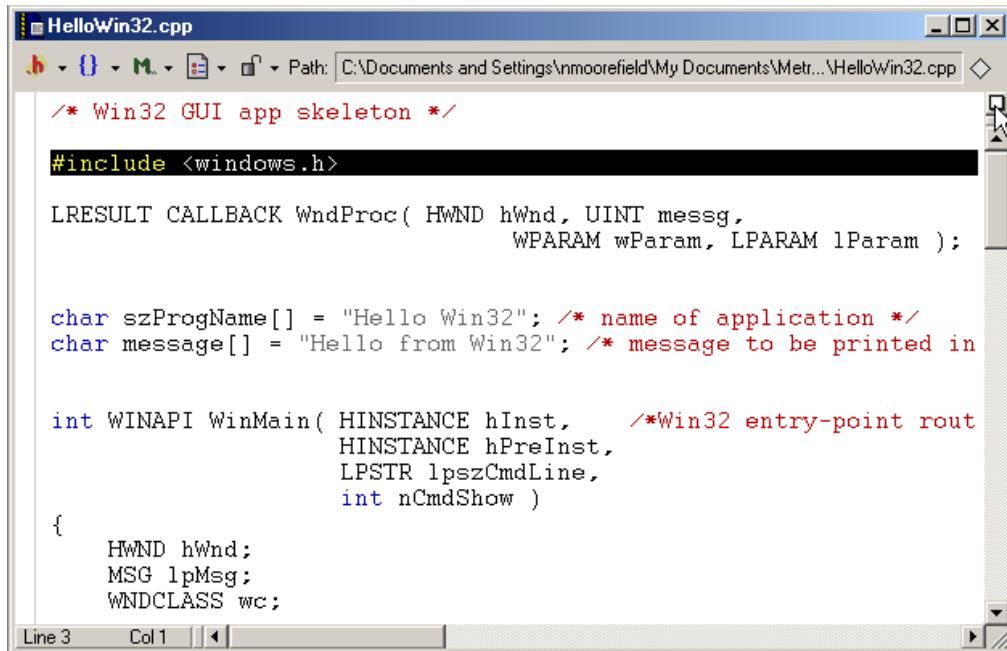
- Open interface and header files
- Find function definitions
- Set and clear markers
- Modify file formats
- Control syntax coloring
- Execute version-control operations
- Determine a file's save state

This section explains how to expand and collapse the toolbar, and how to perform each toolbar task.

## Expanding and Collapsing the Editor Window Toolbar

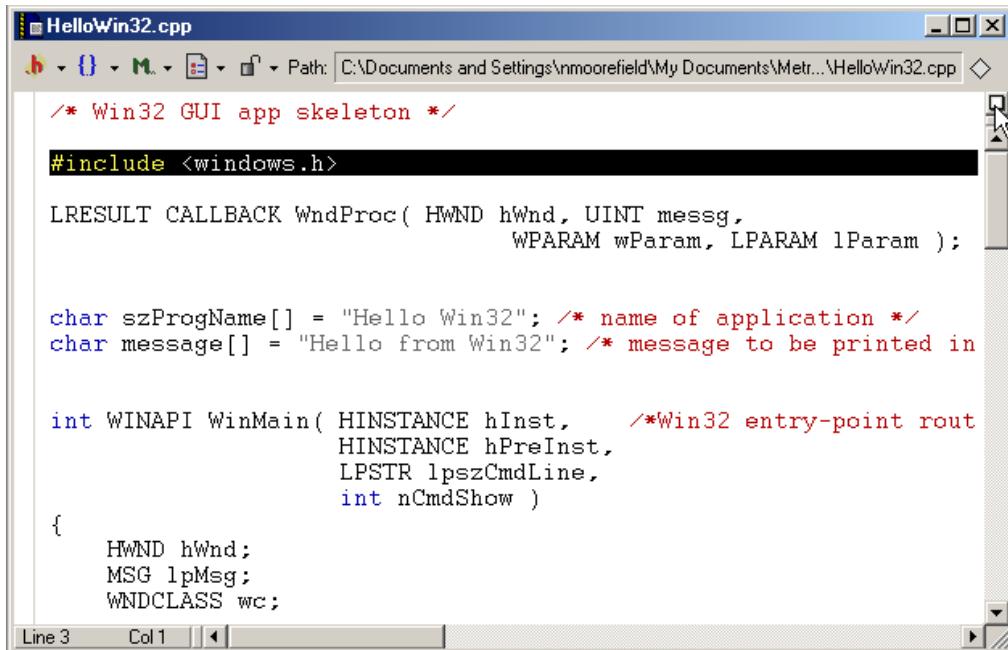
- To expand the editor window toolbar, click this icon in the right-hand top corner of the editor window. [Figure 9.2 on page 94](#) shows an editor window with an expanded toolbar.

**Figure 9.2 Editor window toolbar (expanded)**



- To collapse the Editor Window Toolbar, click this icon in the right-hand top corner of the Editor window. [Figure 9.3 on page 95](#) shows an editor window with a collapsed toolbar.

Figure 9.3 Editor window toolbar (collapsed)



## Interfaces Menu

The Interfaces menu lists the source files included in the current source file.

See "[Finding Interface Files](#)" on page 116 for information on navigating source code with the Interfaces menu.

## Functions Menu

The Functions menu lists the functions (routines) defined in the current file.

See "[Locating Functions](#)" on page 116 for information on navigating source code with the Functions pop-up.

## Markers Menu

The Marker menu lists the markers placed in the current file. Use markers to scroll to specific items in source code and find code segments by intuitive names.

See “[Using Markers](#)” on page 119 for information on navigating source code with Markers.

## Document Settings Menu

The Document Settings menu shows whether the IDE applies syntax coloring to the window text, as well as the format in which the IDE saves the file.

---

### Using the Document Settings Menu

Use the **Document Settings** pop-up in editor windows to toggle syntax coloring on or off for the current file, and set the EOL (end-of-line) format for saving a text file.

The EOL formats are:

- Macintosh: <CR>
- DOS: <CR><LF>
- UNIX: <LF>

### To toggle syntax coloring

- Choose **Document Settings > Syntax Coloring**.

The editor window updates to display the new syntax color setting.

### To specify the EOL format for the file

- Choose the EOL format for the file.

The IDE applies the specified EOL format to the file the next time it gets saved.

## Version Control System Menu

The version control system pop-up in editor windows lists options provided by a version control system (VCS) compatible with the IDE. Use a VCS to manage multiple versions of files. VCS packages are available separately for use with the IDE.

## Using the Version Control System Menu

Use the **Version Control System (VCS)** pop-up in editor windows to access version control commands related to the editor window's file. If a version control system is not enabled for a project, the only item on the VCS menu is "No Version Control Available."

- Choose **VCS > VCScommand**.

The IDE executes the VCS command.

# Other Editor Window Components

Use other editor window components to perform these tasks:

- Determine the path to a file.
- Determine the modification status of a file.
- Set or clear breakpoints.
- Edit text or source code.
- Find the text-insertion point.

This section explains these additional editor window components.

## Path Caption

The Path caption shows the path to the active file. The directory delimiters follow host conventions. For example, slashes separate directories for a path on a Windows computer.

## File Modification Icon

The File Modification icon indicates the save status of the file:

- The  icon indicates an unchanged file since the last **Save**.
- The  icon indicates a file with modifications not yet saved.

## Breakpoints Column

The Breakpoints column shows breakpoints defined in the current file. Each marker in the column indicates the line of source code at which the debugger suspends program execution.

See [“Using Breakpoints” on page 231](#) for information on using breakpoints.

## Text Editing Area

The text editing area behaves the same way as it does in a word processor. Enter text or source code, perform edits, and copy or paste selections.

See [“Editing Source Code” on page 101](#) for information on using special editor features.

## Line and Column Indicator

The Line and Column indicator shows the current position of the text-insertion point. Click the indicator to specify a line to scroll into view.

See [“Going Back and Forward” on page 118](#) for information on navigating source code with the Line and Column indicator.

## Pane Splitter Controls

Use the pane splitter controls to perform these tasks:

- Add panes to editor windows.
- Adjust pane size.
- Remove panes from editor windows.

This section explains how to perform each task.

---

### Adding Panes to an Editor Window

Use the **Pane Splitter** controls to add additional view panes in an editor window and view two or more sections of a source file at the same time.

1. Click and drag a **Pane Splitter control** to add a view pane.

2. The IDE adds a new view pane to the editor window.

---

## Resizing Panes in an Editor Window

Use the **Pane Resize** controls to resize the panes in an editor window.

1. Click and drag a vertical or horizontal **Pane Resize** control.
2. The IDE resizes the selected view pane.

---

## Removing Panes from an Editor Window

Use the **Pane Resize** controls to remove additional view panes from an editor window.

1. Remove an editor window pane.
  - Double-click the **Pane Resize** control to remove the pane.  
OR
  - Click and drag the **Pane Resize** control to the left or top edge of the editor window.
2. The IDE removes the view pane from the editor window.

## **The CodeWarrior Editor**

### *Other Editor Window Components*

---

# Editing Source Code

---

This chapter explains how to edit source code in the CodeWarrior™ IDE. The IDE provides these features to help you edit source code:

- Select and indent text—the editor can select text by line, routine, or rectangular selection. The editor also handles text indentation.
- Balance punctuation—the editor can find matching pairs of parentheses, brackets, and braces. Most programming languages, such as C++, produce syntax errors for punctuation that lacks a counterpart.
- Complete code—the IDE can suggest ways to complete the symbols you enter in a source file

Read this chapter to learn more about editing source code.

This chapter contains these sections:

- [“Text Manipulation” on page 101](#)
- [“Punctuation Balancing” on page 105](#)
- [“Code Completion” on page 107](#)

## Text Manipulation

Manipulate text files to manage their contents from the IDE. Use these tasks to manipulate text files:

- Select text
- Overstrike text
- Use virtual space
- Indent text

This section explains how to perform each task.

## Selecting Text in Editor Windows

The editor lets you select text in several ways while you edit source files.

---

**NOTE** Enable the **Left margin click selects line** option in the **Editor Settings** preference panel to use the right-pointing arrow cursor.

---

### Lines

Follow these steps to select a line of text:

- Triple-click anywhere on a line, or
- Click the right-pointing cursor in the left margin of the line.

These actions select the line of text.

### Multiple lines

Follow these steps to select multiple lines of text:

- Drag the cursor over the contiguous range of text and release, or
- Position the cursor at the beginning of a selection range, then Shift-click the end of the selection range to select all text between the two points, or
- Drag the right-pointing cursor to select lines of text.

These actions select the lines of text.

### Rectangular text selections

[Table 10.1](#) explains how to select rectangular portions of text.

**Table 10.1 Selecting a rectangular portion of text**

On this host...	Do this...
Windows	Alt-drag the cursor over the portion of text.
Macintosh	Command-drag the cursor over the portion of text.

**Table 10.1 Selecting a rectangular portion of text (*continued*)**

On this host...	Do this...
Solaris	Alt-drag the cursor over the portion of text.
Linux	Alt-drag the cursor over the portion of text.

## Entire routines

Follow these steps to select an entire routine:

1. Hold down the **Shift** key.
2. Choose a function name from the **Function** list menu.

This action selects the function.

---

## Overstriking Text (Windows)

Use the Overstrike command to toggle between text insertion and text overwriting mode when entering text.

1. Press the **Ins** key to toggle overstrike mode.
2. Overstrike mode toggles.

---

## Using Virtual Space

**Use the Virtual Space feature** to place the cursor anywhere in the white space of a line of source code and enter text at that position.

For example, consider the line of C++ code shown in [Listing 10.1](#).

---

### Listing 10.1 Sample C++ source code

---

```
void aFunction (const char * inMessage)           virtualspace
```

---

Toggling virtual space changes the cursor behavior:

- enabled—clicking in the *virtualspace* places the cursor at the location that you clicked. You can enter text at that location.

- disabled—clicking in the *virtualspace* places the cursor just after the last character on the line (in the example, just after the closing parenthesis). To place the cursor beyond this character, you must repeatedly press the space bar on your keyboard.

To use virtual space, follow these steps:

1. Select **Edit > Preferences**.

The **IDE Preferences** window opens.

2. Select **Editor Settings** in the IDE Preference Panels list.

The Editor Settings preference panel appears.

3. Configure the **Enable Virtual Space** option:

- select to use virtual space
- clear to not use virtual space

4. Click **Apply** or **Save** to save your changes to the preference panel.

5. Close the IDE Preferences window.

---

## Indenting and Unindenting Text Blocks

Use the **Shift Left** and **Shift Right** commands to shift a selected block of text to the left or right. You can indent or unindent one or more lines using these commands. The **Tab Size** option specifies the amount of indentation.

1. Select the text to be shifted.
2. Indent or unindent the selected text.
  - To unindent text: Choose **Edit > Shift-Left**.
  - To indent text: Choose **Edit > Shift-Right**.

The IDE shifts the text block.

## Symbol Editing Shortcuts

You can use the browser contextual menu to enhance source-code editing in the IDE. Use this menu to streamline text entry in editor windows. You can enter the first few letters of a function name, then use the browser contextual menu to complete the entry.

The IDE also provides these keyboard shortcuts with the browser enabled:

- **Find symbols with prefix**—find symbols matching the selected prefix
- **Find symbols with substring**—find symbols matching the selected substring
- **Get next symbol**—obtain the next symbol from the browser database
- **Get previous symbol**—obtain the previous symbol from the browser database

See the *IDE Quick Reference* card for more information about these keyboard shortcuts.

## Punctuation Balancing

Balance punctuation to ensure that each opening parenthesis, bracket, or brace has a corresponding closing counterpart. This section explains how to balance punctuation.

---

### Balancing Punctuation

Use the **Balance** option when editing source code to make sure that every parenthesis (( )), bracket ([ ]), and brace ({ }) has a mate.

1. Position the cursor between the suspect punctuation.
  2. Check for the matching punctuation.
    - Choose **Edit > Balance**.
- OR
- Double-click the parenthesis, bracket, or brace character to check for a matching character.

From a text insertion point, the editor searches forward until it finds a parenthesis, bracket, or brace, then it searches in the opposite direction until it finds the matching punctuation. When double-clicking on a parenthesis, bracket, or brace, the editor searches in the opposite direction until it finds the matching punctuation.

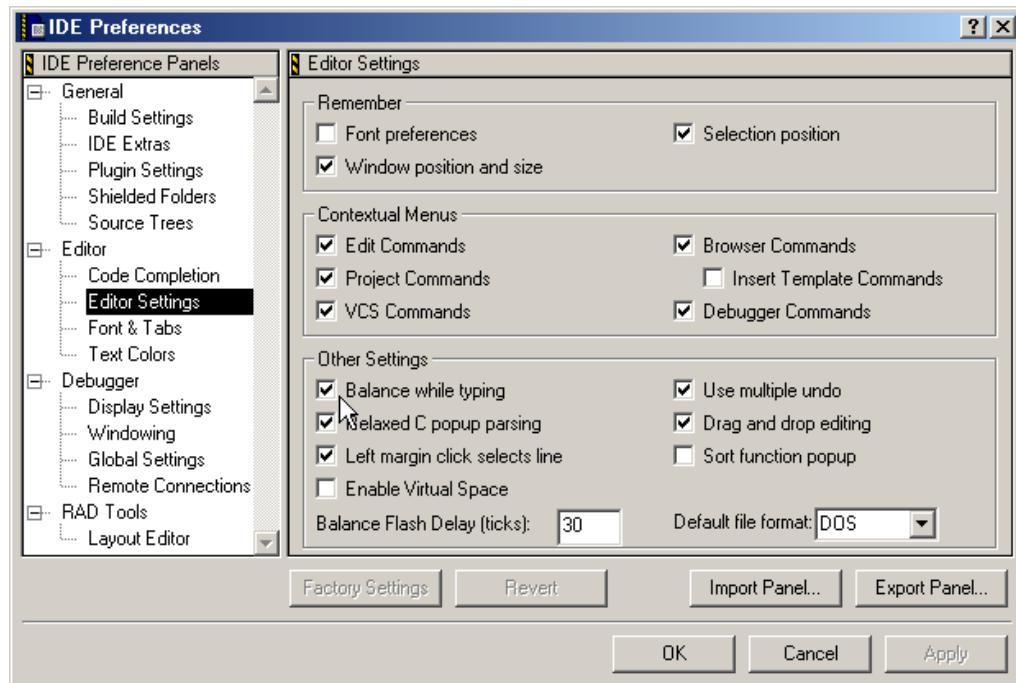
When it finds a match, it highlights the text between the matching characters. If the insertion point is not enclosed or if the punctuation is unbalanced, the computer beeps.

---

## Toggling Automatic Punctuation Balancing

Use the **Editor Settings** to enable or disable the punctuation balancing feature.

**Figure 10.1 Editor Settings (Balance While Typing)**



To toggle automatic punctuation balancing, follow these steps:

1. From the **Edit** menu in the Main Toolbar, select **Preferences**.  
This opens the **IDE Preferences** window.
2. In the **IDE Preference Panels** list, select **Editor**.
3. Choose **Editor Settings**.

4. In the **Other Settings** area of Editor Settings, select or clear the **Balance While Typing** checkbox.

## Code Completion

Use code completion to have the IDE automatically suggest ways to complete the symbols you enter in a source file. By using code completion, you avoid referring to other files to remember available symbols.

## Code Completion Configuration

You can activate, deactivate, and customize code-completion operation. Use these tasks to configure code completion:

- Activate automatic code completion
- Trigger code completion from the IDE menu bar
- Trigger code completion from the keyboard
- Deactivate automatic code completion

---

### Activating Automatic Code Completion

Activate automatic code completion to have the IDE display a Code Completion window that helps you complete the symbols you enter in source code. The **Code Completion** preference panel configures the Code Completion window behavior.

1. Choose **Edit > Preferences**.
- The **IDE Preferences** window appears.
2. Select the **Code Completion** preference panel in the **IDE Preference Panels** list.
- The **Code Completion** preference panel appears.
3. Select the [\*\*Automatic Invocation\*\*](#) option.

Selecting this option configures the IDE to automatically open the Code Completion window.

4. Enter a delay in the [Code Completion Delay](#) field.

This delay determines how long the IDE waits between the time you type a trigger character and the time the Code Completion window appears. If you perform any action during this delay time, the IDE cancels the Code Completion operation.

5. Save your preferences.

Click the **Save** or **Apply** button.

6. Choose **File > Close**.

The **IDE Preferences** window closes.

The Code Completion window now appears automatically to help you complete code in editor windows.

---

## Triggering Code Completion from the IDE Menu Bar

Trigger code completion from the main IDE menu bar to open the Code Completion window.

1. Bring forward an editor window.
2. Place the insertion point at the end of the source code that you want to complete.
3. Choose **Edit > Complete Code**.

The Code Completion window appears. Use it to complete the symbol at the insertion point.

---

## Triggering Code Completion from the Keyboard

Trigger code completion from the keyboard to bypass opening the Code Completion window.

1. Bring forward an editor window.
2. Place the insertion point at the end of the source code to complete.

3. Press the appropriate code-completion key binding.

[Table 10.2 on page 109](#) lists the default code-completion key bindings for each IDE host. Use the **Customize IDE Commands** window to change these key bindings.

**Table 10.2 Code Completion key bindings**

Host	Get Next Completion	Get Previous Completion	Complete Code
Windows	Alt-/	Alt-Shift-/	Alt-.
Macintosh	Control-/	Control-Shift-/	Control-.

---

## Deactivating Automatic Code Completion

Deactivate automatic code completion to prevent the IDE from displaying the Code Completion window as you edit source code. The **Code Completion** preference panel configures Code Completion window behavior.

You can still manually trigger code-completion functionality from the keyboard or from the main IDE menu bar.

---

**NOTE** To dismiss the Code Completion window after it automatically opens, press the **Esc** key or click outside the active editor window.

---

1. Choose **Edit > Preferences**.  
The **IDE Preferences** window appears.
2. Select the **Code Completion** preference panel in the **IDE Preference Panels** list.  
The **Code Completion** preference panel appears.
3. Clear the [Automatic Invocation](#) option.  
Clearing this option prevents the IDE from automatically opening the Code Completion window.
4. Save your preferences.  
Click the **Save** or **Apply** button.

## Editing Source Code

### Code Completion

---

5. Choose File > Close.

The IDE Preferences window closes.

The IDE now requires you to manually open the Code Completion window from the keyboard or from the main menu bar.

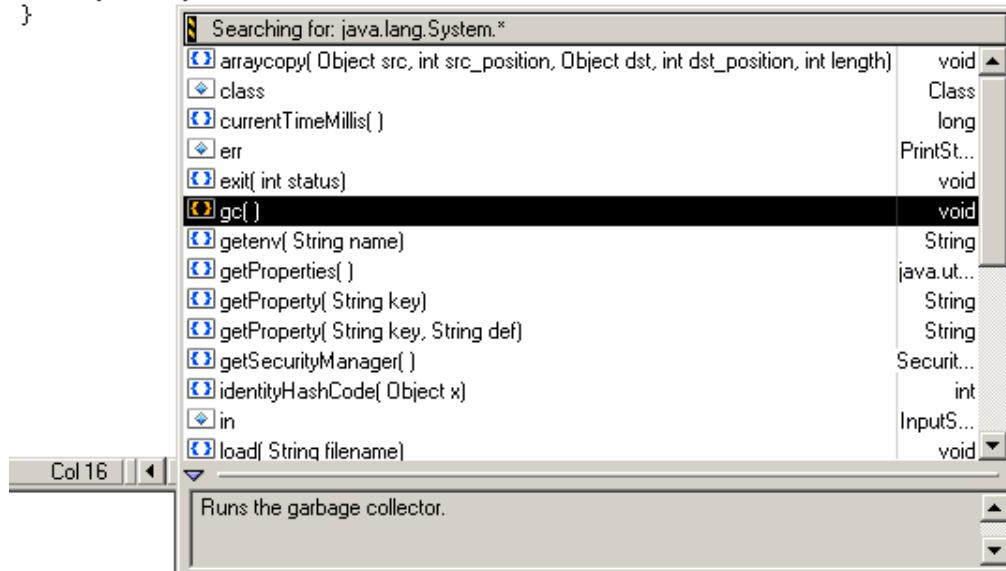
## Code Completion Window

The Code Completion window displays possible symbols based on the context of the insertion point.

[Figure 10.2](#) shows the Code Completion window. [Table 10.3 on page 111](#) explains the items in the Code Completion window. [Table 10.4 on page 111](#) explains the icons that appear in the Code Completion list.

**Figure 10.2** Code Completion window

```
public static void main(String args[]) {  
    System.out.println( "Hello World!" );  
    System.|
```



**Table 10.3 Code Completion window—items**

Item	Icon	Explanation
Code Completion list		Lists available variables and methods or functions along with their corresponding return types or parameters. This list changes based on the context of the insertion point in the active editor window. Icons help distinguish items in the list.
Disclosure Triangle		Click to toggle display of the Documentation pane for those programming languages that support it.
Resize Bar		Drag to resize the Code Completion list and the Documentation pane.
Documentation pane		Displays summary information or documentation for the selected item in the Code Completion list. This pane appears only for programming languages that support summary information or documentation.

**Table 10.4 Code Completion window—icons**

Icon	Code Type	Icon	Code Type
	Class		Method
	Function		Namespace
	Global Variable		None
	Language Keyword		Package
	Local Variable		Variable

---

## Navigating the Code Completion Window

Navigate the Code Completion window by mouse or keyboard. You can perform these tasks:

- Resize the window

## Editing Source Code

### Code Completion

---

- Navigate the window by keyboard
  - Refine the Code Completion list by keyboard
1. Bring forward an editor window.
  2. Place the insertion point at the end of the source code to complete.
  3. Choose **Edit > Complete Code**.
- The Code Completion window appears.
4. Use the mouse to resize the Code Completion window.

The new window size remains in effect until you refine the Code Completion list or close the Code Completion window. You refine the Code Completion list by typing additional characters in the active editor window.

5. Use the keyboard to navigate the Code Completion list.

[Table 10.5](#) explains how to navigate the Code Completion list by keyboard.

**Table 10.5 Navigating the Code Completion list by keyboard**

Key	Action
Up Arrow	Select the previous item
Down Arrow	Select the next item
Page Up	Scroll to the previous page
Page Down	Scroll to the next page

6. Use the keyboard to refine the Code Completion list.

The Code Completion list updates as you add or delete characters in the active editor window. Continue adding characters to narrow the list, or delete existing characters to broaden the list. Press the Backspace key to delete characters.

---

## Selecting an Item in the Code Completion Window

Select an item in the Code Completion window to have the IDE enter that item in the active editor window at the insertion point.

1. Bring forward an editor window.
2. Place the insertion point at the end of the source code to complete.
3. Choose **Edit > Complete Code**.

The Code Completion window appears. The Code Completion list in this window shows possible symbol matches for the text in the active editor window.

4. Select an item in the Code Completion list.

Use the mouse or the keyboard to navigate the Code Completion list and select an item.

5. Enter the item into the active editor window.

Press the **Return** or **Enter** keys on the keyboard or double-click the item to have the IDE insert that item into the editor window.

---

## Completing Code for Data Members

Complete code for data members for those programming languages that support it. For example, in the Java programming language you can complete code for data members through the `.` character.

1. Bring forward an editor window.
2. Place the insertion point at the end of the class to complete.
3. Type a data-member trigger character.

The character depends on the programming language you use. In the Java programming language, for example, type a period.

4. The Code Completion window appears.

Use this window to complete the data member for the class.

## **Completing Code for Parameter Lists**

Complete code for parameter lists for those programming languages that support it. For example, in the Java programming language you can complete code for parameter lists through the ( character.

1. Bring forward an editor window.
2. Place the insertion point at the end of the function or method to complete.
3. Type a parameter-list trigger character.  
The character depends on the programming language you use. In the Java programming language, for example, type an open parenthesis.

4. The Code Completion window appears.  
The upper portion of this window lists different (overloaded) versions of the function or method. The lower portion shows possible parameter lists for the selected function or method in the top portion. Use this window to complete the parameter list for the function or method.

# Navigating Source Code

---

This chapter explains how to navigate source code in the CodeWarrior™ IDE. Navigate source code to accomplish these tasks:

- Find specific items—the editor finds interface files, functions, and lines of source code.
- Go to a specific line—the editor can scroll to a specific line of source code.
- Use markers—the editor allows labelling of specific items of text. These labels, or markers, provide intuitive navigation of text.

Read this chapter to learn more about typical tasks for navigating source code.

This chapter contains these sections:

- [“Finding Interface Files, Functions, and Lines” on page 115](#)
- [“Going Back and Forward” on page 118](#)
- [“Using Markers” on page 119](#)
- [“Symbol Definitions” on page 121](#)
- [“Reference Templates \(Macintosh\)” on page 123](#)

## Finding Interface Files, Functions, and Lines

Find interface files, functions, and lines of source code to expedite programming. You can find these types of items:

- interface files
- functions
- lines of source code

This section explains how to perform each task.

---

## Finding Interface Files

Find interface (header) files to view files referenced by the current source code. Some programming languages, such as C++, use interface files in conjunction with source code. Interface files typically define functions or objects used in the source code. Interface files also separate function or object declarations from implementations. This section explains how to find interface files.

---

### Using the Interface Menu

Use the Interface menu in editor windows to open interface or header files referenced by the current file. The project file must be open for the Interface menu to operate.

1. Click the Interface menu.
2. Select the filename of the interface file that you want to open.

If found, the file is opened in an editor window. If not found, an alert sounds.

---

**NOTE**

Only source code interface files can be opened. Libraries and pre-compiled header files can not be opened.

---

## Locating Functions

Find functions to expedite source-code editing. Most source files contain several functions that divide a complicated task into a series of simpler tasks. The editor allows scrolling to individual functions within the current source file. This section explains how to find functions.

---

### Using the Functions Menu

Use the Functions menu in editor windows to quickly navigate to specific functions or routines in the current source file.

1. Click the Functions menu.
2. Select the routine name to view.

The editor scrolls to display the selected routine.

---

## Alphabetizing Functions Menu with the Mouse and Keyboard

The default behavior of the Functions menu is to list functions in order of appearance in the source file. You can use the mouse and keyboard to list functions in alphabetical order.

[Table 11.1 Alphabetizing the Functions list](#) explains how to use the mouse and keyboard to alphabetize functions in the Functions menu.

**Table 11.1 Alphabetizing the Functions list**

On this host...	Do this...
Windows	Ctrl-click the Functions menu.
Macintosh	Option-click the Functions menu.
Solaris	Alt-click the Functions menu.
Linux	Alt-click the Functions menu.

---

## Alphabetizing Functions Menu Order

The default behavior of the Functions menu is to list functions in order of appearance in the source file. You can select the [Sort function popup](#) option in the **Editor Settings** preference panel to list functions in alphabetical order.

1. Open the **IDE Preferences** window.  
Choose **Edit > IDE Preferences**.
2. Select the **Editor Settings** preference panel.
3. Select the [Sort function popup](#) option.
4. Save your modifications to the **Editor Settings** panel, as explained in [Table 11.2 on page 118](#).

**Table 11.2 Saving changes to the IDE Preferences window**

<b>On this host...</b>	<b>Do this...</b>
Windows	Click <b>Apply</b> , then <b>OK</b> .
Macintosh	Click <b>Save</b> , then close the IDE Preferences window.
Solaris	Click <b>Save</b> , then close the IDE Preferences window.
Linux	Click <b>Save</b> , then close the IDE Preferences window.

## Going Back and Forward

Go back and forward in source files to edit existing code. Most source files contain more than one screen of code. The editor always counts the number of lines in the source files. Go to a particular line to scroll a particular item into view.

---

### Going to a Particular Line

Use the **Goto Line** command to navigate to a specific source line in an editor window if you know its number. Lines are numbered consecutively, with the first line designated as line 1. The **Line Number** control at the bottom of the editor window shows the number of the line where the text insertion point is positioned.

1. Open the Line Number window.
  - Click the **Line and Column Indicator** control.  
OR
  - Choose **Search > Go To Line**.
2. Type a line number in the **Line Number** text box.
3. Click **OK**.

---

**NOTE** If a line number does not exist, the insertion point jumps to the last line of the source file.

---

# Using Markers

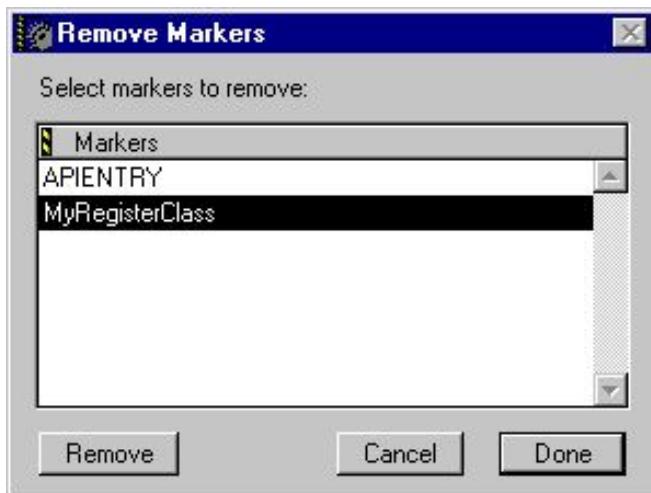
Markers behave like labels in the editor, identifying specific parts of source code. Use these tasks to work with markers:

- Add markers to a source file
- Navigate to a marker
- Remove some or all markers from a source file

## Remove Markers Window

Use the **Remove Markers** window to manage the use of destination markers in source files. [Figure 11.1](#) shows the Remove Markers window. [Table 11.3](#) explains the items in the window.

**Figure 11.1** Remove Marker window



**Table 11.3** Remove Markers window—items

Item	Explanation
Markers list	Displays a list of all markers in the current source file.
Remove button	Click to remove all selected markers.

**Table 11.3 Remove Markers window—items (continued)**

Item	Explanation
Cancel button	Click to close the Remove Markers window without applying changes.
Done button	Click to close the Remove Markers window and apply changes.

---

## **Adding Markers to a Source File**

Use the **Add Marker** command to add a marker to a file to identify specific line locations by name.

1. Position the cursor on a line.
2. Choose **Marker > Add Marker**.
3. Type a name for the new marker.
4. Click **Add**.

The IDE adds the marker to the file.

---

## **Navigating to a Marker**

Once you add a marker, you can use the Marker menu to return to it later.

1. Select the marker name from the Marker menu.
2. The editor window scrolls to display the selected marker.

---

## **Removing a Marker from a Source File**

Use the **Remove Marker** command to remove one or more markers from a source file.

1. Choose **Marker > Remove Markers**.
2. Select the marker name to remove from the list.

3. Click **Remove**.

The IDE removes the selected marker.

---

## Removing All Markers from a Source File

Use the **Remove Marker** command to remove one or more markers from a source file.

1. Choose **Marker > Remove Markers**.

2. Select all markers in the **Markers** list, as explained in [Table 11.4](#).

**Table 11.4 Selecting all markers in the Markers list**

On this host...	Do this...
Windows	Shift-click each marker name in the list.
Macintosh	Select <b>Edit &gt; Select All</b> .
Solaris	Select <b>Edit &gt; Select All</b> .
Linux	Select <b>Edit &gt; Select All</b> .

3. Click **Remove**.

The IDE removes all markers.

# Symbol Definitions

You can find a symbol definition in your project's source code. For the Mac OS, you can also look up a symbol definition using the online documentation viewer in the **IDE Extras** selection in the **IDE Preferences** panel.

Supported online reference viewers include HTMLHelp (Windows) and QuickHelp (Mac OS), as well as older online help systems such as QuickView (Mac OS) and THINK Reference (Mac OS).

---

**TIP**

You can also use the browser to look up symbol definitions.

---

**Figure 11.2 Find Definition**



---

## Looking Up Symbol Definitions

To look up the definition of a selected symbol, follow these steps:

1. From the **Search** menu in the Main Toolbar, choose **Find Definition**.
2. Enter the symbol definition.
3. Click **OK**.

CodeWarrior searches all of the files in your project for the symbol definition.

If CodeWarrior finds a definition, it opens an editor window and highlights the definition for you to examine.

---

**TIP** To return to your original location after viewing a symbol definition, press Shift-Ctrl B (Windows) or Shift-Command B (Mac OS). This key binding is equivalent to the **Go Back** menu command.

---

**Mac OS, Solaris, and Linux** You can also use the **Find Reference** and **Find Definition & Reference** commands to look up symbol definitions. After you select a symbol and choose the Find Reference command, CodeWarrior searches the online documentation for the symbol definition. After you select a symbol and choose the Find Definition & Reference command, the IDE searches both the project files and the online documentation for the symbol definition. If CodeWarrior does not find a definition or reference, it notifies you with a beep.

# Reference Templates (Macintosh)

If you look up a routine (such as an operating system call) in the QuickView or THINK Reference online viewers, you can paste the template for the call into the active editor window at the text-insertion point. If you know the name of the call that you want to add to your source code, but are not familiar with the call parameters, this technique is useful.

[Listing 11.1](#) shows a sample routine template.

## **Listing 11.1 Sample routine template**

---

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

---

## **Inserting a Reference Template**

To insert a reference template into your code, follow these steps:

1. From the online viewer window, type the routine name that you want to insert.
2. Select the name you just typed.
3. Choose **Insert Reference Template** from the **Edit** menu.

The IDE searches for the routine in either QuickView (Mac OS) or THINK Reference (Mac OS), starting the required application if it is not already running. If the IDE finds the routine, the IDE copies the template to the active editor window and replaces the text you selected with the template.

## **Navigating Source Code**

*Reference Templates (Macintosh)*

---

# Finding and Replacing Text

---

This chapter explains how to work with the find-and-replace features in the CodeWarrior™ IDE. Use these features to perform these tasks:

- Find text—the IDE provides several ways to find text in a single file or in a collection of files.
- Replace text—the IDE substitutes found text with replacement text. Replace one or more occurrences of found text.

This chapter contains these sections:

- [“About Find and Replace” on page 125](#)
- [“Single-File Find” on page 126](#)
- [“Single-File Find and Replace” on page 127](#)
- [“Multiple-File Find and Replace” on page 129](#)
- [“Search Results Window” on page 136](#)
- [“Text-Selection Find” on page 137](#)
- [“Regular-Expression Find” on page 140](#)

## About Find and Replace

The IDE provides find-and-replace operations that streamline repetitive text modifications and enhance common word-processing features:

- Search multiple files—change recurring text within more than one file. For example, use this feature to change a function name throughout the files that comprise a project.
- Search specific files—perhaps a required change only affects certain files in the project. Create a file set that contains these files. The IDE improves efficiency by searching only the specified files in the file set.

The IDE provides different windows for find operations, replace operations, and find-and-replace operations.

## Single-File Find

Use the **Find** window to search for text within a single file:

- The **Find** operation returns a single instance of matching text.
- The **Find All** operation returns all instances of matching text.

[Figure 12.1](#) shows the Find window. [Table 12.1](#) explains the items in the Find window.

**Figure 12.1** Find window



**Table 12.1** Find window—items

Item	Explanation
Find text box	Enter the search string.
Find list box	Select a previously searched string to insert into the Find text box.
Find	Click to start a search operation using the string in the Find text box.
Find All	Click to search for all matches in the active editor window.
Cancel	Click to close the Find window without performing a search.
Match whole word	Select to search for whole-word matches only, ignoring matches within words.
Case sensitive	Select to search for matches of the search string according to text case.
Regular expression	Select to interpret find and replace strings as regular expressions.
Search selection only	Select to search only the currently selected text and not the entire file.

---

Table 12.1 Find window—items (*continued*)

Item	Explanation
Stop at end of file	Select to stop a search at the end of a file and not wrap around to the beginning of the file.
Up	Click to perform a search operation back from the current selection point.
Down	Click to perform a search operation forward of the current selection point.

---

## Searching Text in a Single File

Use the **Find** command to search text in the active editor window.

1. Choose **Search > Find**.

---

**NOTE** (Mac OS, Solaris, and Linux) Use the **Customize IDE Commands** window to activate the **Find** menu command.

---

2. Type search text into **Find** text box.
3. Set search options.
4. Start the **Find** or **Find All** operation.
  - Click **Find** to find a single match.
  - Click **Find All** to find all matches.

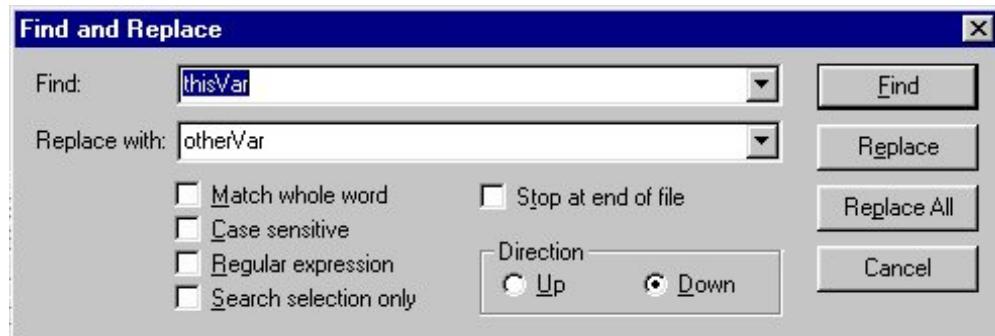
The IDE searches the current file and displays matching text or beeps if no matching text occurs.

## Single-File Find and Replace

Use the single-file **Find and Replace** window to perform these tasks:

- Search a single file.
- Replace found text in a single file.

**Figure 12.2 Find and Replace window**



**Table 12.2 Find and Replace window—items**

Item	Explanation
Find text box	Enter the search string.
Find list box	Select a previously searched string to insert into the Find text box.
Find	Click to start a search operation using the string in the Find text box.
Replace with text box	Enter the replacement string.
Replace with list box	Select a previously used replacement string to insert into the Replace text box.
Replace	Click to replace the current editor window selection with the replacement string.
Match whole word	Select to search for whole-word matches only, ignoring matches within words.
Cancel	Click to close the Find and Replace window without performing a search.
Case sensitive	Select to search for matches of the search string according to text case.
Regular expression	Select to interpret find and replace strings as regular expressions.
Search selection only	Select to search only the currently selected text and not the entire file.
Stop at end of file	Select to stop a search at the end of a file and not wrap around to the beginning of the file.
Up	Click to perform a search operation back from the current selection point.
Down	Click to perform a search operation forward of the current selection point.

## Replacing Text in a Single File

Use the **Replace** command to search text in source files and replace matching text on a case-by-case basis.

1. Open the **Find and Replace** window, as explained in [Table 12.3](#).

**Table 12.3 Opening the Find and Replace window**

On this host...	Do this...
Window	Select <b>Search &gt; Replace</b> .
Macintosh	Select <b>Search &gt; Find and Replace</b> .
Solaris	Select <b>Search &gt; Find and Replace</b> .
Linux	Select <b>Search &gt; Find and Replace</b> .

2. Type search text into the **Find** text box.
3. Type replacement text into the **Replace with** text box.
4. Set replacement options.
5. Start the **Replace** or **Replace All** operation.
  - Click **Replace** to replace a single match.
  - Click **Replace All** to replace all matches.

The IDE searches the current file and replaces matching text or beeps if no matching text occurs.

## Multiple-File Find and Replace

Use the multiple-file **Find in Files** window to perform these tasks:

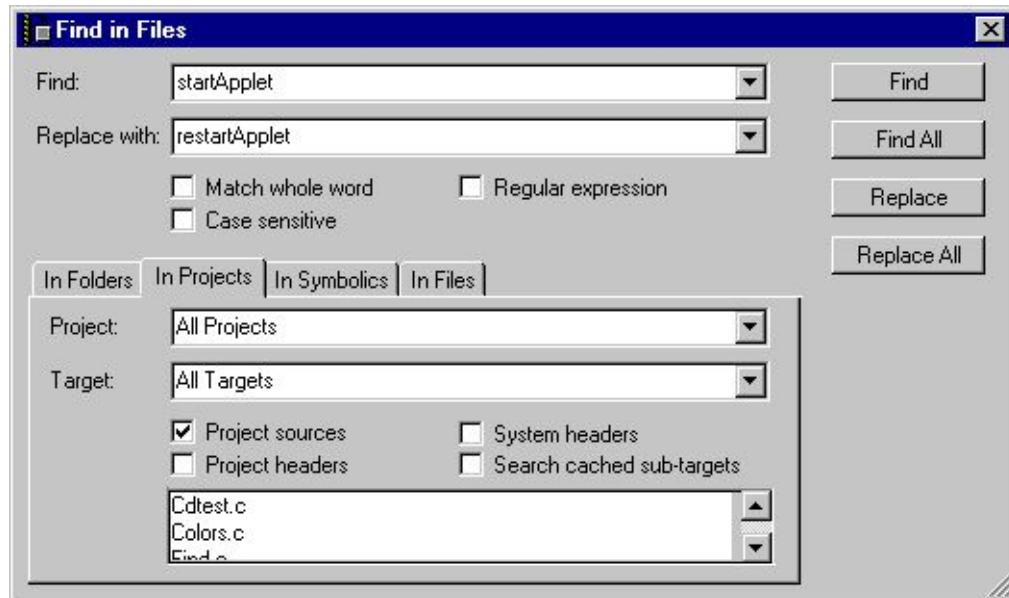
- Search several files.
- Replace found text in multiple files.
- Replace found text in files within a specific build target.

## Finding and Replacing Text

### Multiple-File Find and Replace

[Figure 12.3](#) shows the Find in Files window. [Table 12.4](#) explains the items in the window.

**Figure 12.3** Find in Files window



**Table 12.4** Find in Files window—items

Item	Explanation
Find text box	Enter the search string.
Find list box	Select a previously searched string to insert into the Find text box.
Find	Click to start a search operation using the string in the Find text box.
Find All	Click to search for all matches to the find string.
Replace with text box	Enter the replacement string.
Replace with list box	Select a previously used replacement string to insert into the Replace text box.
Replace	Click to replace the current editor window selection with the replacement string.
Replace All	Click to replace all matches in the file with the replacement string.

**Table 12.4 Find in Files window—items (*continued*)**

Item	Explanation
Match whole word	Select to search for whole-word matches only, ignoring matches within words.
Case sensitive	Select to search for matches of the search string according to text case.
Regular expression	Select to interpret find and replace strings as regular expressions.
In Folders	Click to search specific folders in the host computer's file system.
In Projects	Click to search active projects and build targets.
In Symbolics	Click to search files containing symbolics (debugging and browsing) information generated by the IDE.
In Files	Click to search files contained in custom file sets.

## Searching Text Across Multiple Folders

Use the **In Folders** tab to search text within folders on the host computer.

1. Choose **Search > Find in Files**.
2. Type search text into the **Find** text box.
3. Set general search options.
4. Click the **In Folders** tab.
5. Select the folder to search.
  - Enter in the **Search in** field the path to the folder to search.
  - Choose a recently searched folder from the **Search in** list box.
  - Click **Choose** to navigate the computer's file system and select the folder to search.
6. If desired, select **Search sub-folders** to search all folders inside the selected folder.
7. Select file types to search, as explained in [Table 12.5](#). Separate items in the file-type list with commas.

**Table 12.5 Selecting file types to search**

On this host...	Do this...
Windows	Edit the <b>By type</b> list.
Macintosh	Edit the <b>Ends with</b> list.
Solaris	Edit the <b>Ends with</b> list.
Linux	Edit the <b>Ends with</b> list.

8. Start the **Find** or **Find All** operation.
  - Click **Find** to find a single match.
  - Click **Find All** to find all matches.

The IDE searches the specified folders and displays matching text or beeps if no matching text occurs.

---

## Searching Text Across Multiple Projects

Use the **In Projects** tab to search text within active projects.

1. Choose **Search > Find in Files**.
2. Type search text into the **Find** text box.
3. Set general search options.
4. Click the **In Projects** tab.
5. Choose the projects to search from the **Project** list box.
  - Choose **All Open Projects** to search all active projects.
  - Choose by name a specific project to search.
6. Choose the build targets to search from the **Target** list box.
  - Choose **All Targets** to search all build targets in the selected projects.
  - Choose **Default Targets** (available after choosing **All Open Projects**) to search the default build target of each project.

- Choose by name a specific build target (available after choosing a specific project) to search.
7. Set additional search options for the selected projects and build targets.
- Select **Project sources** to search all source files.
  - Select **Project headers** to search all header files.
  - Select **System headers** to search all system header files.
  - Select **Search cached sub-targets** to search targets in the selected build targets stored for frequent use by the IDE.

---

**NOTE** Choose **Project > Make** to update the project data to correctly list source files, header files, and sub-targets.

---

8. Edit the file list at the bottom of the **In Projects** tab as desired. The list reflects the previously selected search options.

---

**TIP** Select, remove, and open files directly from the file list.

---

9. Start the **Find** or **Find All** operation.
- Click **Find** to find a single match.
  - Click **Find All** to find all matches.

The IDE searches the specified projects and displays matching text or beeps if no matching text occurs.

---

## Searching Text Across Multiple Symbolics Files

Use the **In Symbolics** tab to search text within project files that contain symbolics information. To generate symbolics information for most files:

- Activate the browser for the project and choose **Project > Make**.
- Choose **Project > Debug** to debug the project.

---

**NOTE** The IDE does not generate symbolics information for some files, such as libraries.

---

1. Choose **Search > Find in Files**.
2. Type search text into the **Find** text box.
3. Set general search options.
4. Click the **In Symbolics** tab.
5. Choose the symbolics to search from the **Symbolics** list box.
  - Choose **All Symbolics** to search all symbolics files in the project.
  - Choose by name a specific symbolics file to search.

---

**NOTE** Choose **Project > Make** to update the project data to correctly list symbolics files.

---

6. Edit the file list at the bottom of the **In Symbolics** tab as desired. The list reflects the previously selected search options.

---

**TIP** Select, remove, and open files directly from the symbolics file list.

---

7. Start the **Find** or **Find All** operation.
  - Click **Find** to find a single match.
  - Click **Find All** to find all matches.

The IDE searches the specified symbolics files and displays matching text or beeps if no matching text occurs.

---

## Searching Text Across Multiple Files

Use the **In Files** tab to search text in a file set. A file set lists specific files to search.

1. Choose **Search > Find in Files**.
2. Type search text into the **Find** text box.
3. Set general search options.

4. Click the **In Files** tab.
5. Select a file set from the **File Set** options. The list at the bottom of the **In Files** tab shows the files to search.
  - Choose **Open Editor Files** to search currently open editor files.
  - Choose **New File Set** to create a set of files to search.
6. Edit the file list at the bottom of the **In Files** tab as desired.
  - Click **Add Files** to navigate the computer's file system and select a file to add to the current list.
  - Click **Clear List** to remove all items from the current list.
  - Click **Save This Set** to save the list as a new file set.
  - Click **Remove a Set** to remove an existing file set.

---

**TIP**      Select, remove, and open files directly from the file list. Also drag and drop files or folders to the list to modify it.

---

7. Start the **Find** or **Find All** operation.
  - Click **Find** to find a single match.
  - Click **Find All** to find all matches.

The IDE searches the specified files and displays matching text or beeps if no matching text occurs.

---

## Replacing Text Across Multiple Items

Use the **Replace** and **Replace All** commands to search text and replace matching text.

1. Choose **Search > Find in Files**.
2. Type search text into the **Find** text box.
3. Type replacement text into the **Replace** text box.
4. Set general search options.

## Finding and Replacing Text

### Search Results Window

5. Set additional search options in the **In Folders**, **In Projects**, **In Symbolics**, or **In Files** tabs.
6. Start the **Replace** or **Replace All** operation.
  - Click **Replace** to replace a single match.
  - Click **Replace All** to replace all matches.

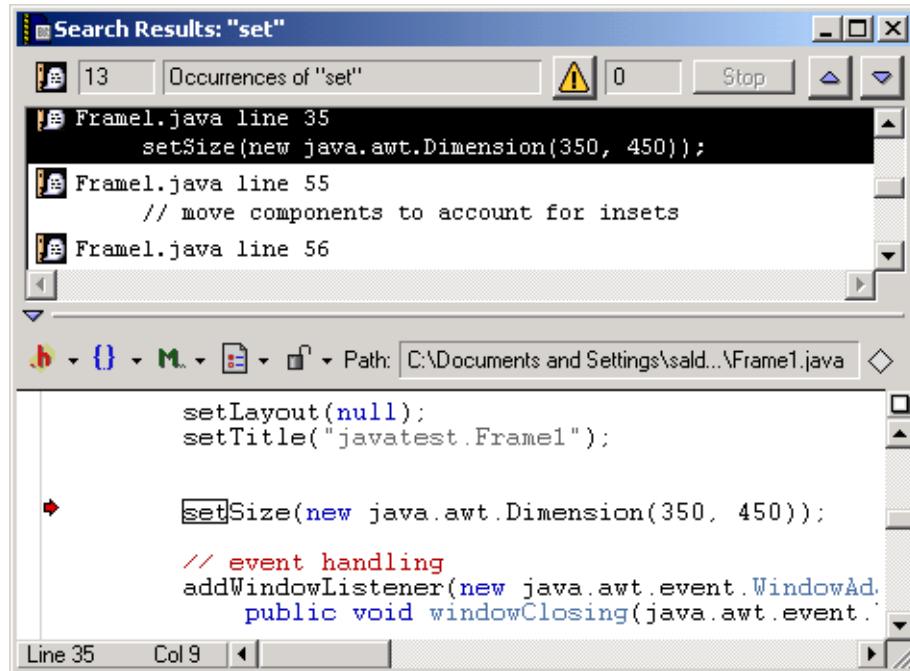
The IDE searches the specified items and replaces matching text or beeps if no matching text occurs.

## Search Results Window

Use the **Search Results** window to explore matches found by the IDE during multiple-file search operations. Also use the window to stop searches in progress.

[Figure 12.4](#) shows the Search Results window. [Table 12.6 on page 137](#) explains the items in the window.

**Figure 12.4** Search Results window



**Table 12.6 Search Results window—items**

Item	Icon	Explanation
Messages caption		Shows the total number of search results.
Search Information caption		Shows the search criteria.
Warnings button		Click to display compiler and linker warnings in the Results pane.
Stop button		Click to stop the search in progress.
Previous Message button		Click to select the previous search result.
Next Message button		Click to select the next search result.
Results pane		Lists individual search results.
Source Code disclosure triangle		Click to show or hide the Source Code pane.
Pane resize bar		Drag to resize the Results and Source Code panes.
Source Code pane		Shows the source code corresponding to the selected item in the Results pane.

## Text-Selection Find

Search for text without having to use the **Find**, **Find and Replace**, or **Find in Files** windows. The IDE assumes that the most recent criteria defined in these windows also applies to the current search. Select text in the active editor window to define the search string.

---

### Using the Find Previous Command

When searching for text, you can use the **Find Previous** command to have the IDE find a previous match.

Follow these steps to use the Find Previous command:

## Finding and Replacing Text

### *Text-Selection Find*

---

1. Select **Edit > Commands & Key Bindings**.

The **Customize IDE Commands** window opens.

2. Click the **Commands** tab in the Customize IDE Commands window.

The window shows a customizable list of IDE commands.

3. Expand the **Search** item in the Commands list (the left-hand pane).

4. Select the **Find Previous** item in the expanded list.

If necessary, scroll in order to see the Find Previous item.

5. Information about the Find Previous item appears in the right-hand pane.

6. Select the **Appears in Menus** checkbox in the right-hand pane.

The Find Previous command will appear in the Search menu in the main IDE menu bar.

7. Click **Save** to confirm your changes.

8. Close the **Customize IDE Commands** window.

You can now select the Find Previous command in the Search menu. You can also use the key binding associated with Find Previous.

---

## Searching with a Text Selection

Use the **Find Selection** command to search the active editor window for selected text.

1. Select the text to use as the search string.

2. Choose **Search > Find Selection**.

3. If desired, find preceding or succeeding matches.

- Choose **Search > Find Previous** to find a preceding match.
- Choose **Search > Find Next** to find a subsequent match.

The IDE searches the active editor window and scrolls to matching text or beeps if no matching text occurs.

<b>NOTE</b>	(Macintosh) Hold down the Shift key in order to select <b>Search &gt; Find Previous</b> .
-------------	---

---

On the Windows-hosted IDE, use the **Customize IDE Commands** window to activate the **Find Previous** menu command. For more information, see the task **Using the Find Previous Command**.

---

## Searching with a Text Selection Across Multiple Windows

Use the **Enter Find String** command to change the current search string, then search through multiple editor windows.

1. Select the text to use as the search string.
2. Choose **Search > Enter Find String**.  
The selected text replaces the search string defined in the **Find**, **Find and Replace**, and **Find in Files** windows.
3. Bring forward the editor window to search.
4. Choose **Search > Find Selection**.
5. If desired, find preceding or succeeding matches.
  - Choose **Search > Find Previous** to find a preceding match.
  - Choose **Search > Find Next** to find a subsequent match.

The IDE searches the active editor window and scrolls to matching text or beeps if no matching text occurs.

<b>NOTE</b>	(Macintosh) Hold down the Shift key in order to select <b>Search &gt; Find Previous</b> .
-------------	---

---

On the Windows-hosted IDE, use the **Customize IDE Commands** window to activate the **Find Previous** menu command. For more information, see the task **Using the Find Previous Command**.

---

# Regular-Expression Find

Use regular expressions to search text according to sophisticated text-matching rules. A *regular expression* is a text string used as a mask for matching text in a file. To use regular expressions, select **Regular expression** in the **Find**, **Find and Replace**, or **Find in Files** windows. Certain characters become operators with special meanings in a regular expression.

---

**TIP** For an in-depth description of regular expressions, refer to the World Wide Web for information on `regexp(3)` by Henry Spencer, and *Mastering Regular Expressions* by Jeffrey E.F. Friedl, published by O'Reilly & Associates, Inc.

---

[Table 12.7](#) explains the regular-expression operators that the IDE recognizes.

**Table 12.7 Regular-expression operators recognized by the IDE**

Operator	Name	Explanation
.	match any	Matches any single printing or non-printing character except newline and null.
*	match zero or more	Replaces the smallest/preceding regular expression with a sub-expression.
+	match one or more	Repeats the preceding regular expression at least once and then as many times as necessary to match the pattern.
?	match zero or one	Repeats the preceding regular expression once or not at all.
\n	back reference	Refers to a specified group (a unit expression enclosed in parentheses) in the find string. The digit <i>n</i> identifies the <i>n</i> th group, from left to right, with a number from 1 to 9.
	alternation	Matches one of a choice of regular expressions. If this operator appears between two regular expressions, the IDE matches the largest union of strings.
^	match beginning of line	Matches items from the beginning of a string or following a newline character. This operator also represents a list operator when enclosed within brackets.
\$	match end of line	Matches items from the end of a string or preceding a newline character.

**Table 12.7 Regular-expression operators recognized by the IDE (continued)**

Operator	Name	Explanation
[ . . . ]	list	Defines a set of items to use as a match. The IDE does not allow empty lists.
( . . . )	group	Defines an expression to be treated as a single unit elsewhere in the regular expression.
-	range	Specifies a range. The range starts with the character preceding the operator and ends with the character following the operator.

## Matching Any Character

Use the `.` operator to match any character except a newline character. The operator effectively becomes a placeholder for a single character in the search string.

[Table 12.8](#) shows examples of using regular expressions to match any character.

**Table 12.8 Examples of matching any character**

This regular expression...	...matches this text...	...in this text sample:
<code>var.</code>	<code>var1</code> <code>var2</code>	<code>cout &lt;&lt; var1;</code> <code>cout &lt;&lt; var2;</code>
<code>c.t</code>	<code>cut</code> <code>cot</code>	<code>cin &gt;&gt; cutF;</code> <code>cin &gt;&gt; cotG;</code>

## Repeating Expressions

Use the `*` and `+` operators to indicate recurring characters in the search string. The IDE matches the largest possible string with the regular expression.

[Table 12.9](#) shows examples using repeating text in regular expressions.

**Table 12.9 Examples of repeating expressions**

This regular expression...	...matches this text...	...in this text sample:
s*ion	ion ssion	information the session
s+ion	sion ssion	confusion the session

## Grouping Expressions

Use the ( and ) operators to group regular expressions into a single unit in the search string. The IDE applies operators to the unit.

[Table 12.10](#) shows examples of grouping text in regular expressions.

**Table 12.10 Examples of grouping expressions**

This regular expression...	...matches this text...	...in this text sample:
ris	ris	surprise
r( i)s	r is	the value of theVar is

## Choosing One Character from Many

Use the [ and ] operators to define a list of possible character matches. The IDE matches the search string with text that contains any character in the list. Precede the list with ^ to match text with any character *not* in the list. Any operator immediately following the [ operator becomes a literal character.

Use the - operator to define a range of possible character matches. The IDE matches the search string with text that contains any character in the defined range. The - operator becomes a literal character when placed at the beginning or end of a list.

[Table 12.11](#) shows examples of using regular expressions to choose one character from many characters.

**Table 12.11 Examples of choosing one character from many**

This regular expression...	...matches this text...	...in this text sample:
[bls]ag	sag bag lag	the sagging bag lagged
[[aeiou][0-9]]	[2 u9	cout << a[2] << u9;
[^bls]ag	rag	the sagging rag lagged
[-ab]V	aV -V	aVal-Val;

## Matching Line Beginnings and Endings

Use the ^ operator to identify a search string to match only with the beginning of a line. Similarly, use the \$ operator to identify a search string to match only with the end of a line.

[Table 12.12](#) shows examples of using regular expressions to match line beginnings and endings.

**Table 12.12 Examples of matching line beginnings and endings**

This regular expression...	...matches this text...	...in this text sample:
^([ \t]*cout)	cout cout	cout << "no tab"; cout << "tab";
(1*;) \$	1; ;	a-ct; a = battLvl; b-ct;

## Using the Find String in the Replace String

Use the & operator to incorporate matching text into a replacement string. The IDE substitutes the matching text for the & operator. Use \& to indicate a literal ampersand in the replacement string.

[Table 12.13](#) shows examples of using the find string in the replace string of regular expressions.

## Finding and Replacing Text

### Regular-Expression Find

---

**Table 12.13 Examples of using the find string in the replace string**

Find string	Replace string	Matching text	After replacement
var[0-9]	my_&	var1	my_var1
tgt	\&target	tgt	&target

## Remembering Sub-expressions

Use the \n construct to recall sub-expressions from the find string in the replacement string. The digit n ranges from 1 to 9 and represents the nth sub-expression in the find string, counting from left to right. Enclose each sub-expression in parentheses.

These tables show a sample replacement operation that recalls sub-expressions. [Table 12.14](#) shows sample find-string, replacement-string, and sub-expression definitions. [Table 12.15](#) shows the result of applying the sample definitions to some text.

**Table 12.14 Sub-expression definitions**

Find string	\#define[ \t]+(.+)[ \t]+([0-9]+);
Replace string	const int \1 = \2;
\1	(.+)
\2	([0-9]+)

**Table 12.15 Remembering sub-expressions**

Sample text	\1 matches this text	\2 matches this text	After replacement
#define var1 10;	var1	10	const int var1 = 10;
#define a 100;	a	100	const int a = 100;

# Browser

---

This section contains these chapters:

- [Using the Browser](#)
- [Using Class Browser Windows](#)
- [Using Other Browser Windows](#)
- [Using Browser Wizards](#)



# Using the Browser

---

This chapter explains how to work with the browser in the CodeWarrior® IDE. Use the browser to perform these tasks:

- Generate a browser database—the browser stores collected symbol information in a browser database for the project. You can generate browser data from the compiler or the language parser.
- Collect symbol information—symbols include functions, variables, and objects. Enable the browser to collect information about the symbols in a project.

Read this chapter to learn more about typical tasks for working with the browser.

This chapter contains these sections:

- [“Browser Database” on page 147](#)
- [“Browser Symbols” on page 151](#)

## Browser Database

The browser database contains information about the symbols in a program, which include—depending on the program language—global variables, functions, classes, and type declarations, among others.

Some IDE windows require that the project contain a browser database. For example, the **Class Hierarchy** window only displays information for a project that contains a browser database. This section explains how to configure a project to generate its browser database.

---

<b>NOTE</b>	Generating a browser database increases the project’s size. To minimize the project’s size, generate the browser database only for the targets you frequently use.
-------------	--

---

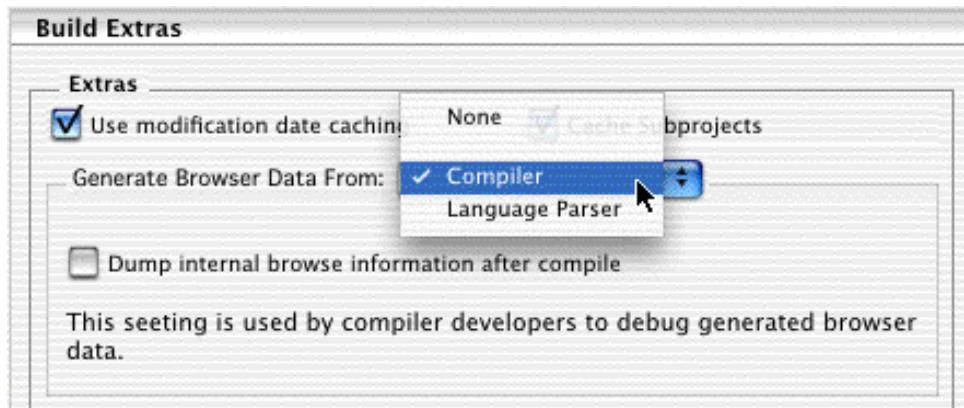
## Browser Data

Browser data contains symbolic and relationship information about the project code. The browser uses this data to access the code information.

Use the **Generate Browser Data From** menu (Figure 13.1) in the **Build Extras** target settings panel to enable and disable browser data generation. This drop-down menu provides these options, which determine where the IDE generates browser data:

- **None**—The IDE does not generate browser data. Use **None** to *disable* browser data. Select None to generate faster compiles (with no browser features).
- **Compiler**—The Compiler generates the browser data. While it compiles more slowly, the compiler generates the most accurate browser data.
- **Language Parser**—The Code Completion plug-in associated with the project's programming language generates the browser data.

**Figure 13.1** Generate Browser Data From menu



## Generating Browser Data

You can select an option in the **Generate Browser Data From** drop-down menu to establish where the IDE generates browser data for a project file.

To generate browser data, follow these steps:

1. Choose **Edit > Target Settings**.
2. From the **Target Settings Panels** list, select **Build Extras**.

3. Choose Compiler or Language Parser from the Generate Browser Data From menu.

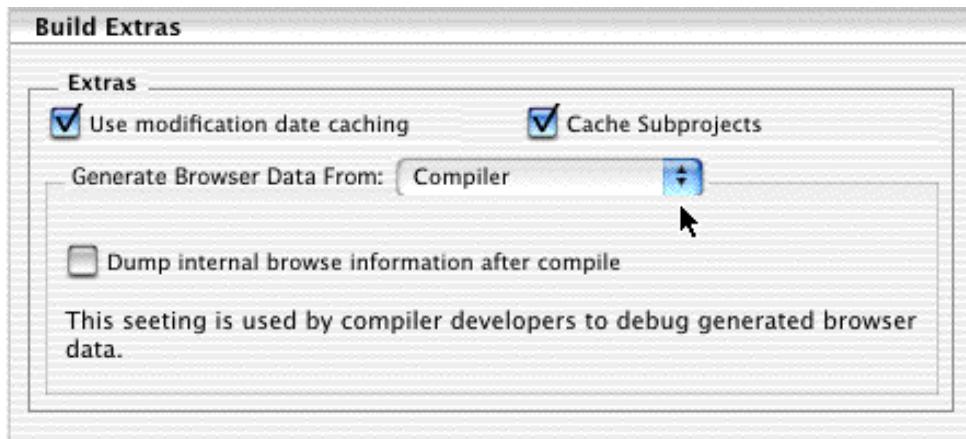
- a. **Compiler**—The compiler generates browser data ([Figure 13.2](#)).

---

**NOTE** Some compilers do not generate browser data.

---

**Figure 13.2 Generate browser data from compiler**



If you enable **Dump internal browse information after compile**, the generated browser data does not appear in a log window after you compile a file.

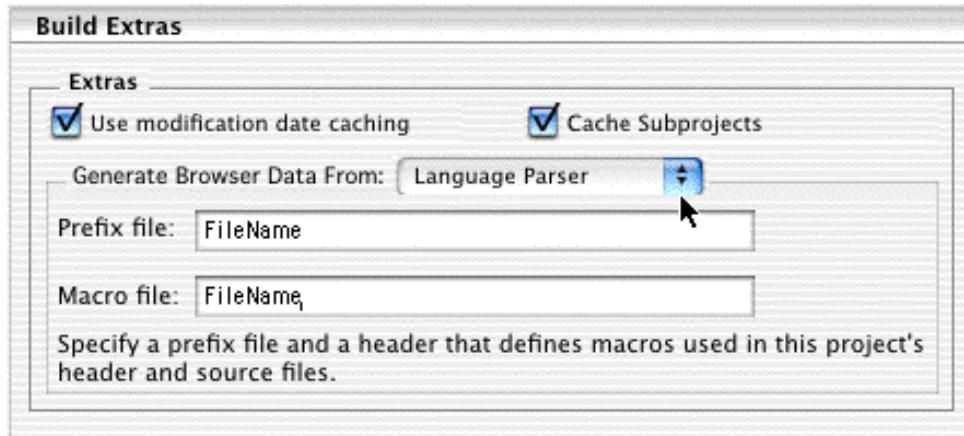
- **Language Parser**—The **Code Completion** plug-in associated with the project's programming language generates the browser data. Browser data and the #include pop-up window update as you edit.

---

**NOTE** Choose Language Parser for C/C++ code completion.

---

**Figure 13.3 Generate browser data from language parser**



Applicable only to C/C++ Code Completion: the **Prefix** and **Macro** files ([Figure 13.3](#)).

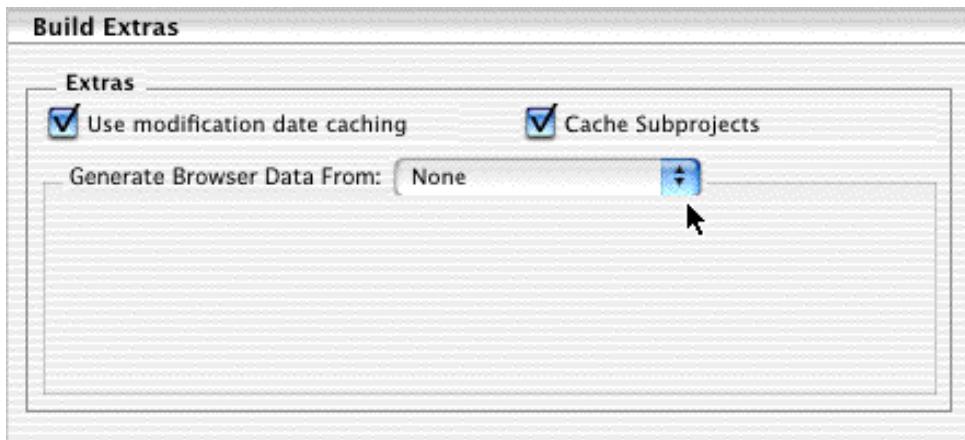
- **Prefix** file—Similar to that used in the **C/C++ Language Settings** panel, the Prefix file contains header files that help the C/C++ Code Completion plug-in parse code. The Prefix file should only include text files (not pre-compiled header files).
  - **Macro** file—Contains C/C++ macro files that help the Code Completion plug-in resolve any #ifdefs found in the source code or in the header files.
4. If you selected **Compiler**, choose **Project > Bring Up To Date** or **Make**.  
The IDE generates browser data for the project.  
If you selected **Language Parser**, the IDE generates browser data in the background.

---

## Disabling Browser Data

Select **None** to disable browser data and stop the IDE from generating browser information for the project.

Figure 13.4 Generate browser data from none



1. Choose **Edit > Target Settings**.
2. Select **Build Extras** from the **Target Settings Panels** list.
3. In the **Generate Browser Data From** drop-down menu, select **None**.
4. Click **Save**.
5. Choose **Project > Make**.

The IDE stops generating browser information.

## Browser Symbols

Navigate browser symbols to open browser views, find symbol definitions, and examine inheritance.

You can navigate browser symbols in these ways:

- Use the Browser contextual menu to open various browser windows for a selected symbol.
- Double-click a symbol name in the Class Browser window to open the file that contains the declaration of that symbol.
- Use the class hierarchy windows to determine the ancestors or descendants of a selected symbol.

## Browser Contextual Menu

Use the IDE's browser contextual menu to enhance source-code editing in the IDE. Use this menu to streamline text entry in editor windows. You can enter the first few letters of a function name, then use the browser contextual menu to complete the entry.

---

### Using the Browser Contextual Menu

Use a contextual menu in the browser to enhance source-code editing.

1. Open the browser contextual menu, as explained in [Table 13.1](#).

**Table 13.1 Opening a browser contextual menu**

On this host...	Do this...
Windows	Right-click a symbol name.
Macintosh	Click and hold on a symbol name.
Solaris	Click and hold on a symbol name.
Linux	Click and hold on a symbol name.

2. Select a command from the contextual menu.

The IDE performs the selected command.

---

### Identifying Symbols in the Browser Database

As a shortcut, you can use browser coloring to help recognize if a symbol resides in the browser database. When you activate a browser, you can see browser-database symbols because they appear in the editor and browser windows according to the colors you select.

---

**TIP** The default color setting is identical for all eight types of browser-database symbols. You can choose a different color for each symbol type.

---

To change the browser symbol colors the editor uses, follow these steps:

1. Choose **Edit > Preferences**.
2. Select the **Text Colors** panel from the **IDE Preference Panels** list.
3. Select the **Activate Syntax Coloring** option.
4. Select the **Activate Browser Coloring** option.
5. Click the color swatch next to the symbol name to set that symbol's color.
6. Click **Save**.

## **Using the Browser**

### *Browser Symbols*

---

# Using Class Browser Windows

---

This chapter explains how to work with the Class Browser windows in the CodeWarrior™ IDE. Use the Class Browser to perform these tasks:

- View browser data—the class browser collects information about the elements of a computer program. Such elements include functions, variables, and classes. The class browser displays these elements in organized lists.
- Show data relationships—the class browser shows the relationships between classes, data members, and methods. The class browser also updates the display to reflect changes in class scope.

Read this chapter to learn more about typical tasks for working with Class Browser windows.

This chapter contains these sections:

- [“Class Browser window” on page 155](#)
- [“Classes pane” on page 161](#)
- [“Member Functions pane” on page 163](#)
- [“Data Members pane” on page 164](#)
- [“Source pane” on page 165](#)
- [“Status Area” on page 165](#)

## Class Browser window

Use the Class Browser window to view information about the elements of a computer program. This section explains how to use the Class Browser window to view browser data, assuming knowledge about activating the browser.

[Figure 14.1 on page 156](#) shows the Class Browser window. [Table 14.1 on page 156](#) explains the items in the window. [Table 14.2 on page 158](#) explains the options in the Browser Access Filters list box.

## Using Class Browser Windows

### Class Browser window

Figure 14.1 Class Browser window

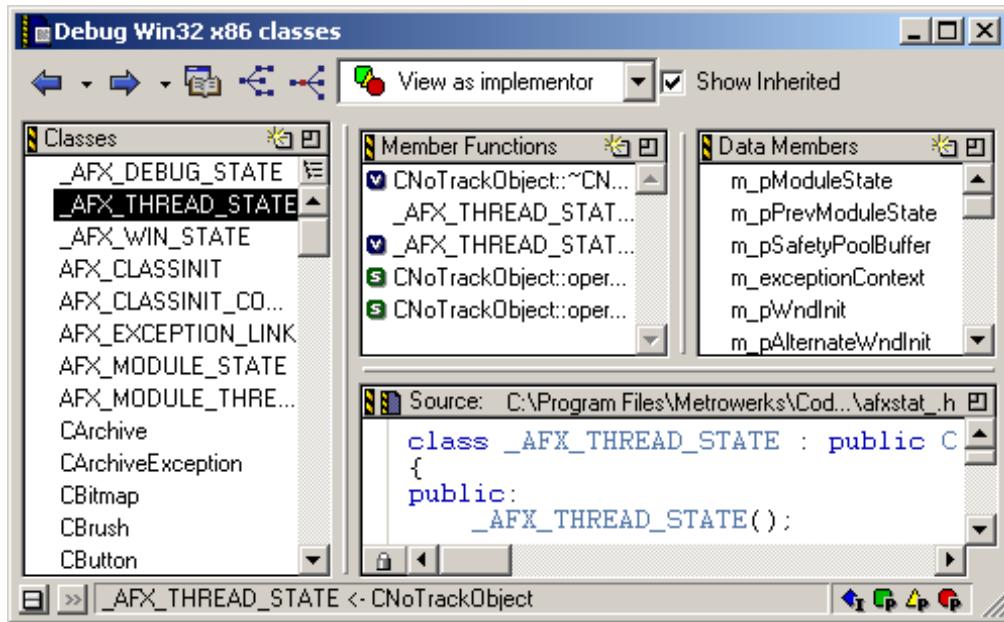


Table 14.1 Class Browser window—items

Item	Icon	Explanation
Go Back button		Click to return to the preceding browser view.
Go Forward button		Click to move to the succeeding browser view.
Browser Contents button		Click to open the Browser Contents window.
Class Hierarchy button		Click to open the Multi-class Hierarchy window.
Single Class Hierarchy Window button		Click to open the Single-class Hierarchy window for the selected class.

**Table 14.1 Class Browser window—items (*continued*)**

Item	Icon	Explanation
Browser Access Filters list box		Select filters for displaying items in class-browser panes.
Show Inherited	<input checked="" type="checkbox"/> Show Inherited	Select to show inherited items in the <a href="#">Member Functions pane</a> and <a href="#">Data Members pane</a> . Clear to hide inherited items from these panes.
<a href="#">Classes pane</a>		Lists all classes in the project browser database.
<a href="#">Member Functions pane</a>		Lists all member functions defined in the currently selected class.
<a href="#">Data Members pane</a>		Lists all data members defined in the selected class.
<a href="#">Source pane</a>		Displays the source code for the currently selected item.
<a href="#">Status Area</a>		Displays various status messages and other information.
Display toggle buttons		Toggles the Classes display between alphabetical and hierarchical listings.
New Item button		Opens wizards to create new items (e.g., classes, data members, member functions).
Pane Expand box		Expands the pane to the width of the full window.
Pane Collapse Box		Collapses the pane to its original size.
Classes Pane button		Lists all classes in the project browser database.
Class Declaration button		Opens a window that shows declarations for all classes in the project.
Open File button		Opens the current source file in a new editor window.
VCS list pop-up		With a version control system enabled, choose the version-control command to execute on the displayed source file.

## Using Class Browser Windows

### Class Browser window

---

**Table 14.2 Browser access filters**

Filter	Icon	Show items with this access:		
		Public	Private	Protected
View as implementor		•	•	•
View as subclass		•		•
View as user		•		
Show public		•		
Show protected				•
Show private			•	

---

## Viewing Class Data from the Browser Contents Window

To view class data for a project in the **Browser Contents** window, follow these steps:

1. Open the **Browser Contents** window, as explained in [Table 14.3 on page 158](#).

**Table 14.3 Opening the Browser Contents window**

On this host...	Do this...
Windows	Select <b>View &gt; Browser Contents</b> .
Macintosh	Select <b>Window &gt; Browser Contents</b> .
Solaris	Select <b>Window &gt; Browser Contents</b> .
Linux	Select <b>Window &gt; Browser Contents</b> .

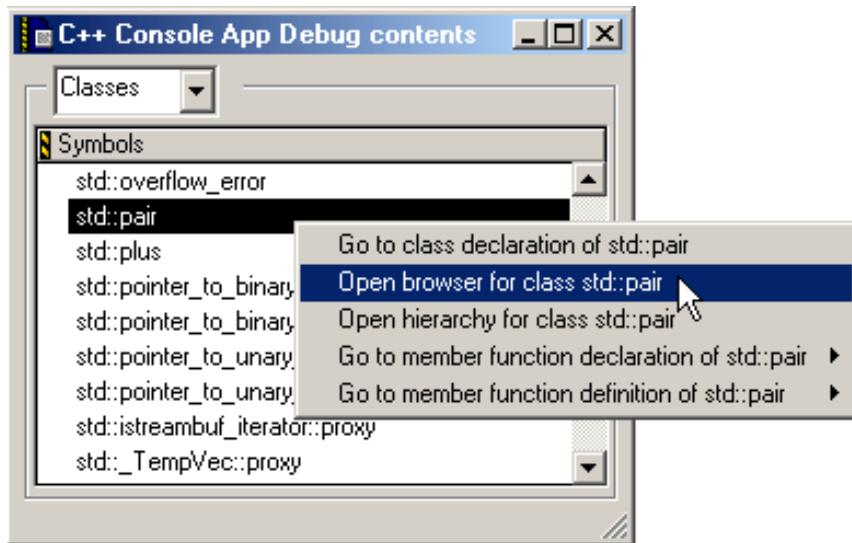
2. Select a class in the Browser Contents window.
3. Open a contextual menu for the selected class, as explained in [Table 14.4](#).

Table 14.4 Opening a contextual menu for the selected class

On this host...	Do this...
Windows	Right-click the selected class.
Macintosh	Control-click the selected class.
Solaris	Click and hold on the selected class.
Linux	Click and hold on the selected class.

A contextual menu like the one shown in [Figure 14.2](#) appears.

Figure 14.2 Browser Contents window—contextual menu



4. Select **Open browser for class *classname*** from the contextual menu.

The ***classname*** is the name of the class that you selected.

A Class Browser window appears.

---

## Viewing Class Data from Hierarchy Windows

To view class data from a hierarchy window, follow these steps:

## Using Class Browser Windows

### Class Browser window

---

1. Open a **Single-Hierarchy** or **Multi-Class Hierarchy** window:
    - a. Click the **Single Class Hierarchy Window** button  in the browser toolbar, or
    - b. Click the **Class Hierarchy** button 
  2. In the Single- or Multi-Class Hierarchy window, double-click a class name.
- A **Class Browser** window appears.

---

## Expanding Browser Panes

Click the **Pane Expand** box (just above the scroll bar in the upper right-hand corner of the pane) to expand the Classes, Function Members, Data Members, or Source panes in a **Browser** window.

1. Click the **Pane Expand** box  to expand a pane.

This pane expands to fill the **Browser** window.

2. Use the enlarged pane to view data.

Alternately, you can use the resize bar between the panes to enlarge each pane (although the pane does not fill the Browser window).

1. Rest the cursor over the resize bar.

The cursor icon changes to this: 

2. Hold down the mouse button.

3. Drag the resize bar to enlarge or shrink the pane.

---

## Collapsing Browser Panes

Click the **Pane Collapse** box (just above the scroll bar in the upper right-hand corner of the pane) to collapse the Classes, Function Members, Data Members, or Source panes in a **Browser** window.

1. Click the **Pane Collapse** box  to collapse a pane.

The chosen pane collapses to its original size.

2. You can now view other panes in a Browser window.

Alternately, you can use the resize bar between the panes to collapse each pane (although the pane does not fill the Browser window).

1. Rest the cursor over the resize bar.

The cursor icon changes to this: 

2. Hold down the mouse button.

3. Drag the resize bar to collapse the pane.

## Classes pane

Use the **Classes** pane to perform these tasks:

- Create a new class
- Toggle viewing of classes
- Sort classes

[Figure 14.1 on page 156](#) shows the Classes pane. [Table 14.5](#) explains the items in the pane.

**Table 14.5 Classes pane—items**

Item	Icon	Explanation
New Item		Click to create a new class using the New Class Wizard.
Sort Alphabetical		Click to sort the Classes list in alphabetical order.
Sort Hierarchical		Click to sort the Classes list in hierarchical order.

## Using Class Browser Windows

### Classes pane

---

## Creating a New Class

Use the **New Class** wizard to specify the name, declaration, and location for a new class. Click **Finish** in any screen to apply default values to any remaining parameters and complete the process. The New Class wizard creates the files that define the class.

1. From the Classes pane, click the **New Item** button .
2. Enter the **Name** and **Location** in the New Class window.
3. To create a more complex class, click **Next** (optional).  
Follow the on-screen directions to further define the class.
4. Click **Finish** to complete the New Class process.

---

## Showing the Classes Pane

Use the **Show Classes** button to expand the Classes pane.

1. Click the **Show Classes** button: 
2. The Classes pane appears in the **Class Browser** window.

---

## Hiding the Classes Pane

Use the **Hide Classes** button to collapse the Classes pane.

1. Click the **Hide Classes** button: 
2. The Classes pane disappears from the **Class Browser** window.

## Sorting the Classes List

Use the **Sort Alphabetical** and **Sort Hierarchical** commands to specify the sort order of classes in the Classes pane. The displayed icon always represents the alternate sort order. For example, when the Classes list appears in alphabetical order, the Sort Hierarchical icon is visible.

- Click the **Sort Alphabetical** icon .

The IDE sorts the Classes list in alphabetical order.

- Click the **Sort Hierarchical** icon .

The IDE sorts the Classes list in hierarchical order.

## Member Functions pane

Use the **Member Functions** pane to perform these tasks:

- Create a new member function
- Determine the inheritance type of a member function

**Table 14.6 Member Function and Data Member identifier icons**

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

---

## Creating a New Member Function

Use the **New Member Function** wizard to specify the name, return type, and parameters for a new member function. Click **Finish** in any screen to apply default values to any remaining parameters and complete the process.

## Using Class Browser Windows

### Data Members pane

---

1. Click the **New Item** button  in the **Member Functions** pane.
2. Enter the **Member Function Declarations** in the **New Member Function** window.
3. Click **Next**.
4. Enter **Member function file locations** and **Include Files** information.
5. Click **Finish**.
6. Review the settings summary, then click **Generate**.

The IDE adds the new member function to the class declaration.

## Data Members pane

Use the **Data Members** pane to create a new data member. This section explains how to create the data member.

Click the New Item button in the Data Members pane to open the New Data Member wizard. See [Table 14.6](#) for a complete list of identifier icons that appear in the Data Members pane.

---

### Creating a New Data Member

Use the **New Data Member** wizard to specify the name, type, and initializer for the new data member. Specify other options to further refine the data member. Click **Finish** in any screen to apply default values to any remaining parameters and complete the process.

1. From the **Data Members** pane, click the **New Item** button: 
2. Enter the **Data Member Declarations** in the **New Data Member** window.
3. Click **Next**.
4. Enter Data Member file locations and `#include` files information.

5. Click **Finish**.
6. Review the settings summary, then click **Generate**.

The IDE adds the new data member to the class declaration.

## Source pane

Use the **Source** pane to view the source code that corresponds to the selected class, member function, or data member. This section explains the items in the **Source** pane.

[Figure 14.1 on page 156](#) shows the Source pane. [Table 14.7](#) explains the items in the pane.

For information on editing source code, see [“Editing Source Code” on page 101](#).

**Table 14.7 Source pane—items**

Item	Icon	Explanation
Open File		Click to open the current source file in a new editor window.
VCS menu		Enable a version-control system in order to activate this menu. Use this menu to select and execute a version-control command on the source file.

## Status Area

Use the status area to perform these tasks:

- Toggle viewing of the **Classes** pane
- View class declarations
- View classes according to public, private, or protected access

[Figure 14.1 on page 156](#) shows the status area. [Table 14.8 on page 166](#) explains the items in the status area.

## Using Class Browser Windows

### Status Area

---

**Table 14.8 Status area—items**

Item	Icon	Explanation
Show Classes Pane		Click to display the Classes pane in the Class Browser window.
Hide Classes Pane		Click to hide the Classes pane in the Class Browser window.
Class Declaration		Click to show the declaration of the current class.
Access Filter Display		Displays the access state of the current class.

# Using Other Browser Windows

---

This chapter explains how to work with the Class Hierarchy windows in the CodeWarrior™ IDE. Use Class Hierarchy windows to perform these tasks:

- View hierarchical browser data—the class hierarchy window shows a graphical representation of hierarchical structure. Object-oriented languages, such as C++ and Java, allow hierarchical relationships between classes.
- Analyze inheritance structure—the class hierarchy window shows the inheritance structure of classes. This structure reveals the data-handling capabilities of a particular class.

Read this chapter to learn more about typical tasks for working with Class Hierarchy windows.

This chapter contains these sections:

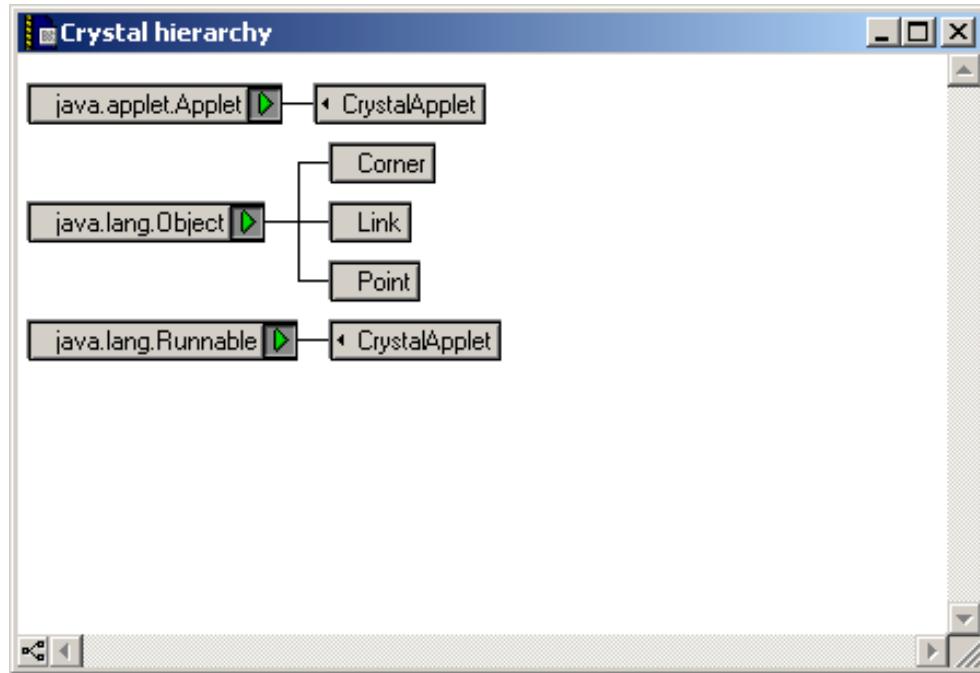
- [“Multiple-Class Hierarchy Window” on page 167](#)
- [“Single-Class Hierarchy Window” on page 170](#)
- [“Browser Contents window” on page 172](#)
- [“Symbols window” on page 174](#)

## Multiple-Class Hierarchy Window

Use the Multi-Class Hierarchy window to visually examine the structure of every class in the browser database. Each class name appears in a box, and lines connect boxes to indicate related classes. The left-most box is the base class, and subclasses appear to the right.

[Figure 15.1 on page 168](#) shows the Multi-Class Hierarchy window. [Table 15.1 on page 168](#) explains the items in the window.

**Figure 15.1 Multi-Class Hierarchy window**



**Table 15.1 Multi-class hierarchy window—items**

Item	Icon	Explanation
Hierarchy Control		Click to expand or collapse the subclasses displayed for a specific class.
Ancestor menu		Click and hold on a class or subclass box to display a menu. Select a class from this menu in order to display that class.
Line button		Click to toggle the lines that connect classes between diagonal and straight lines.

---

## Viewing Browser Data by Inheritance

Use a **Hierarchy** window to view data in graphical form and better understand class relationships. Use the expand and collapse arrows to enlarge or shrink the class views.

1. Activate the browser.
2. Update the browser database by using the **Bring Up To Date, Make, Run, or Debug** command.
3. Open a graphical **Hierarchy** window, as explained in [Table 15.2](#).

**Table 15.2 Opening the Hierarchy window**

On this host...	Do this...
Windows	Select <b>View &gt; Class Hierarchy</b> .
Macintosh	Select <b>Window &gt; Class Hierarchy</b> .
Solaris	Select <b>Window &gt; Class Hierarchy</b> .
Linux	Select <b>Window &gt; Class Hierarchy</b> .

---

## Printing Class Hierarchies

To print the contents of a **Class Hierarchy** window, save an image of the window contents, then print the image file from a graphics-processing application.

The IDE saves the image in a graphics-file format based on the host platform, as shown in [Table 15.3](#).

**Table 15.3 Graphics-file format for host platforms**

Host	Graphics-file Format
Windows	EMF (Enhanced Metafile)
Macintosh	PICT (Picture)
Solaris	PICT (Picture)
Linux	PICT (Picture)

1. Open the **Class Hierarchy** window.
2. Choose **File > Save a Copy As**.
3. Save the image to a file.

4. Open the image file in an graphics-processing application.
5. Print the image file.

The graphics-processing application prints the image of the class hierarchy.

---

## Changing Line Views in a Hierarchical Window

Use the **Diagonal Line** and **Straight Line** commands to change the appearance of the connecting lines between classes and subclasses in a hierarchical window display.

- Click the **Diagonal Line** icon .

The Hierarchical window display updates to use diagonal lines.

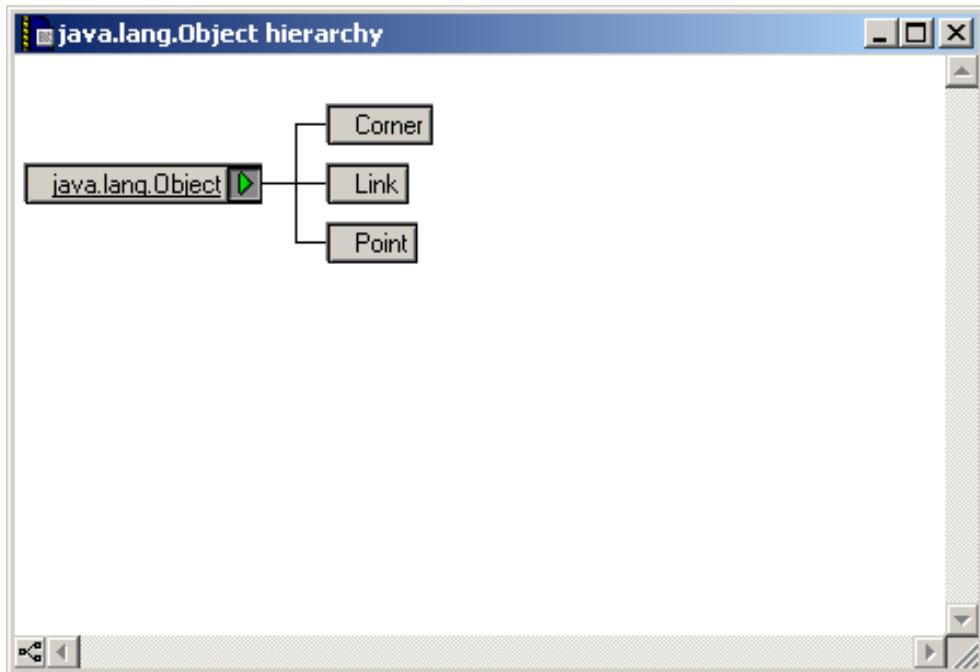
- Click the **Straight Line** icon .

The Hierarchical window display updates to use straight lines.

## Single-Class Hierarchy Window

Use the Single-Class Hierarchy window to examine the structure of a single class in the browser database. The Single-Class Hierarchy window operates identically to the Multi-Class Hierarchy window, but restricts the display to a single class.

Figure 15.2 Single-Class Hierarchy window



The Single-Class Hierarchy window contains the same components as the Multi-Class Hierarchy window.

## Opening a Single-Class Hierarchical window

Use one of these methods to open a Single-Class Hierarchical window:

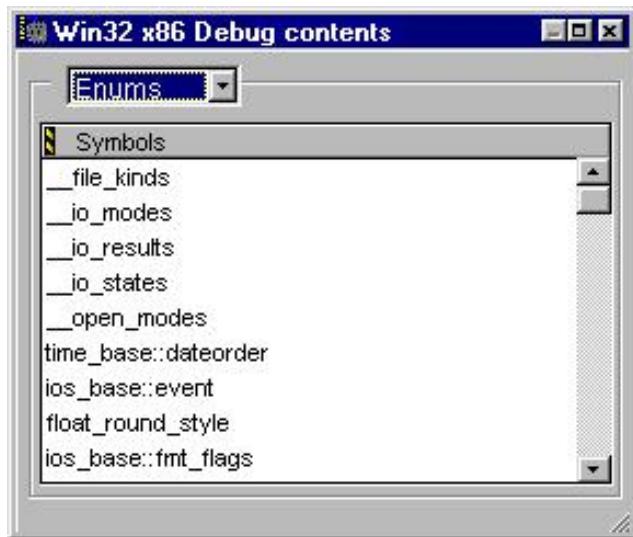
- Click the **Show Single-Class Hierarchy** icon  in a Browser toolbar.
  - Use the Browser Contextual menu in one of these windows:
    - New Class Browser window.
    - Browser Contents window.
    - Multi-Class Hierarchical window.
- A Single-Class Hierarchical window appears.

## Browser Contents window

Use the Browser Contents window to view browser data sorted by category into an alphabetical list. This section explains how to use the Browser Contents window to view browser data, assuming knowledge about activating the browser.

[Figure 15.3](#) shows the Browser Contents window. [Table 15.4](#) explains the items in the window.

**Figure 15.3** Browser Contents window



**Table 15.4** Browser Contents window—items

Item	Icon	Explanation
Symbols list box	Enums	Select the type of symbol to display in the Symbols list.
Symbols list		Double-click a symbol name to display the source file in a new editor window that defines the symbol.

## Viewing Browser Data by Contents

Use the **Browser Contents** window to display symbol information stored in the browser database, listed in alphabetical order. You can choose from these categories:

- classes
- constants
- enumerations
- functions
- global variables
- macros
- function templates
- type definitions

1. Activate the browser.
2. Use the **Bring Up To Date**, **Make**, **Run**, or **Debug** command to update the browser database.
3. Open the Browser Contents window, as explained in [Table 15.5](#).

**Table 15.5 Opening the Browser Contents window**

On this host...	Do this...
Windows	Select <b>View &gt; Browser Contents</b> .
Macintosh	Select <b>Window &gt; Browser Contents</b> .
Solaris	Select <b>Window &gt; Browser Contents</b> .
Linux	Select <b>Window &gt; Browser Contents</b> .

4. Select a category from the **Category** list pop-up.

The symbol information for the selected category appears in alphabetical order in the **Symbols** list.

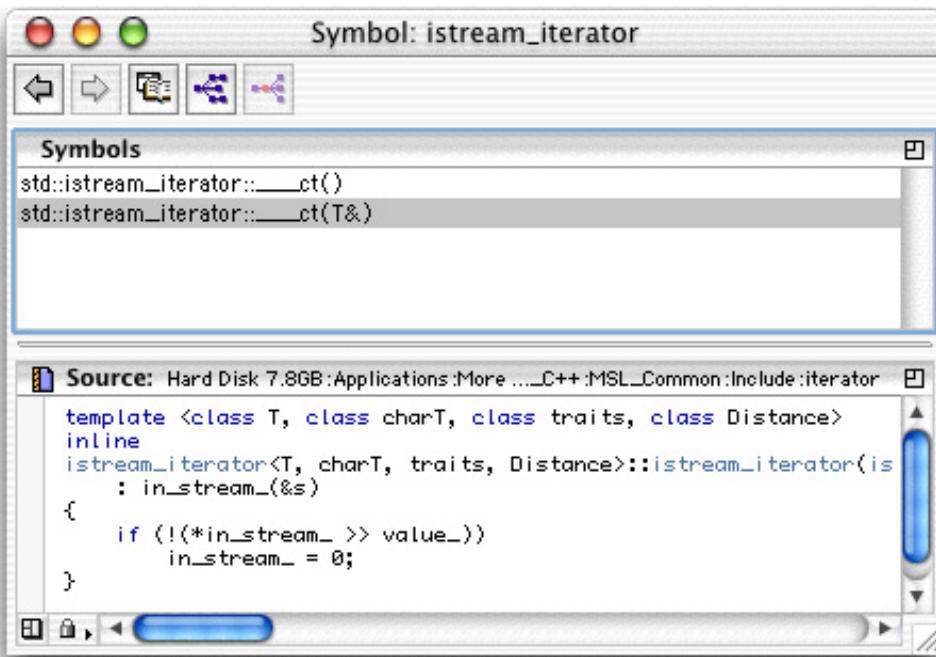
## Symbols window

The Symbols window displays information from project browser databases. With the browser enabled, the IDE generates a browser database for a project during the build process.

The Symbols window displays symbols that have multiple definitions in the browser database. For example, the window displays information about multiple versions of overridden functions in object-oriented code.

[Figure 15.4](#) shows the Symbols window. [Table 15.5 on page 173](#) explains the items in the window.

**Figure 15.4 Symbols window**



**Table 15.6 Symbols window—items**

Item	Explanation
<a href="#">Symbols toolbar</a>	Provides one-click access to common browser commands and class-filtering commands.
<a href="#">Symbols pane</a>	Displays a list of all symbols with multiple declarations.
<a href="#">Source pane</a>	Displays the source code for the currently selected item.

---

## Opening the Symbols Window

Use the **Symbols** window to list all implementations, whether overridden or not, of any symbol that has multiple definitions. You can access the Symbols window by using a contextual menu.

1. Open a contextual menu, as explained in [Table 15.7](#).

**Table 15.7 Opening the Symbols window**

On this host...	Do this...
Windows	Right-click the symbol name.
Macintosh	Control-click the symbol name.
Solaris	Click and hold on the symbol name.
Linux	Click and hold on the symbol name.

2. Select **Find all implementations of** from the contextual menu that appears.
3. The Symbols window opens.

## Symbols toolbar

Most of the Symbol toolbar items are identical to those in the [Class Browser window](#).

## Symbols pane

The **Symbols** pane lists symbols with multiple definitions in the browser database. Select a symbol from the list to view its definition in the **Source** pane.

## Source pane

The **Source** pane used in the Symbols window is identical to the one used by the [Class Browser window](#). See [“Source pane” on page 165](#) for more details.

# Using Browser Wizards

---

When you create a new class, member function, or data member in the IDE, you use browser wizards. These wizards provide the steps to help you complete the process.

This chapter provides information on these wizards:

- [“The New Class Wizard” on page 177](#)
- [“The New Member Function Wizard” on page 182](#)
- [“The New Data Member Wizard” on page 185](#)

---

**NOTE**

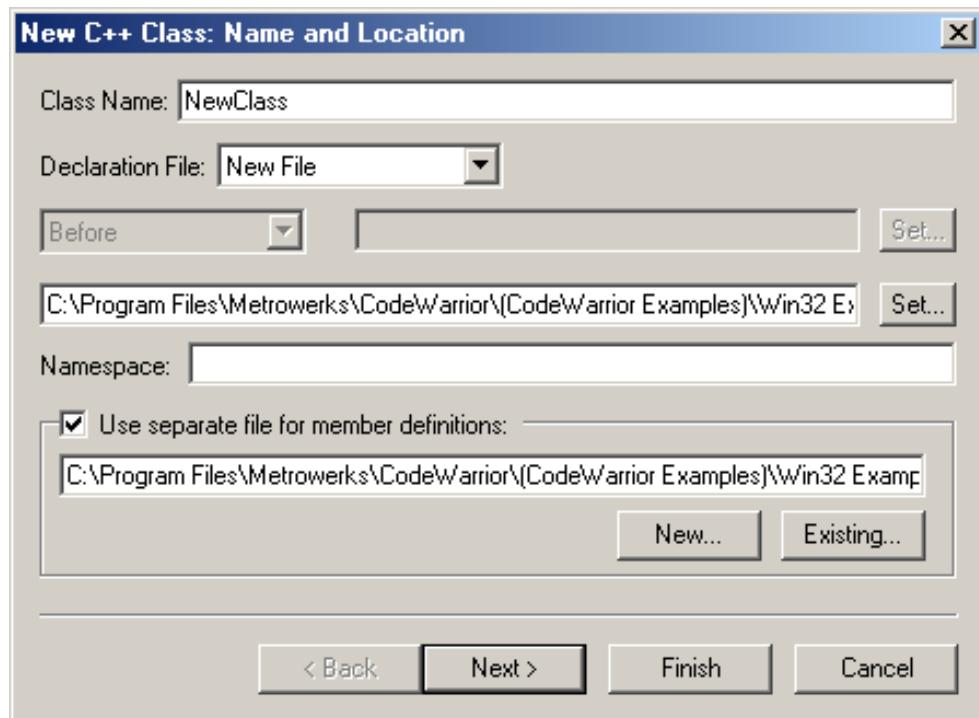
Most wizard pages contain default settings. To accept all current settings in the wizard, click **Finish** in any screen. The wizard displays a summary of all the current settings for the new project. Click **Generate** to accept the current settings and create the new item, or click **Cancel** to return to the wizard to modify settings.

---

## The New Class Wizard

Use the **New Class** wizard to specify the name, declaration, and location for a new class. Click **Finish** in any screen to apply default values to remaining parameters to complete the process. The **New Class** wizard creates the files that define the class.

**Figure 16.1** New Class wizard—Name and Location



---

## Using the New Class Wizard

To use the New Class Wizard, follow these steps:

1. Open the **Class Browser** window, as explained in [Table 16.1](#).

**Table 16.1** Opening the Class Browser window

On this host...	Do this...
Windows	Select <b>View &gt; Class Browser</b> .
Macintosh	Select <b>Window &gt; Class Browser</b> .

**Table 16.1 Opening the Class Browser window (*continued*)**

On this host...	Do this...
Solaris	Select <b>Window &gt; Class Browser</b> .
Linux	Select <b>Window &gt; Class Browser</b> .

2. Select **Browser > New Class**.

---

**NOTE** You can also click the New Item icon  in the Class Browser window to create a new class.

---

3. In the **New C++ Class** wizard, enter **Name and Location** information:
  - a. **Class Name**—Enter a name for the class in this field.
  - b. **Declaration File**—This menu lets you specify whether the file is a **New File**, which is a new declaration file, or **Relative to class**, which is a declaration that depends on an existing file in the project.

If you choose the **New File** option, type in the enabled field the path where you want to save the file. Alternatively, click **Set** next to the field to choose the path in which to save the file.

If you choose the **Relative to class** option, select **Before** or **After** to establish the order of the new class in relation to existing classes. In the field next to the Before and After drop-down selection, type the name of the class you want to relate to the new class. Alternatively, click **Set** next to this field, type the name of a class in the window that opens, and then click **Select**.

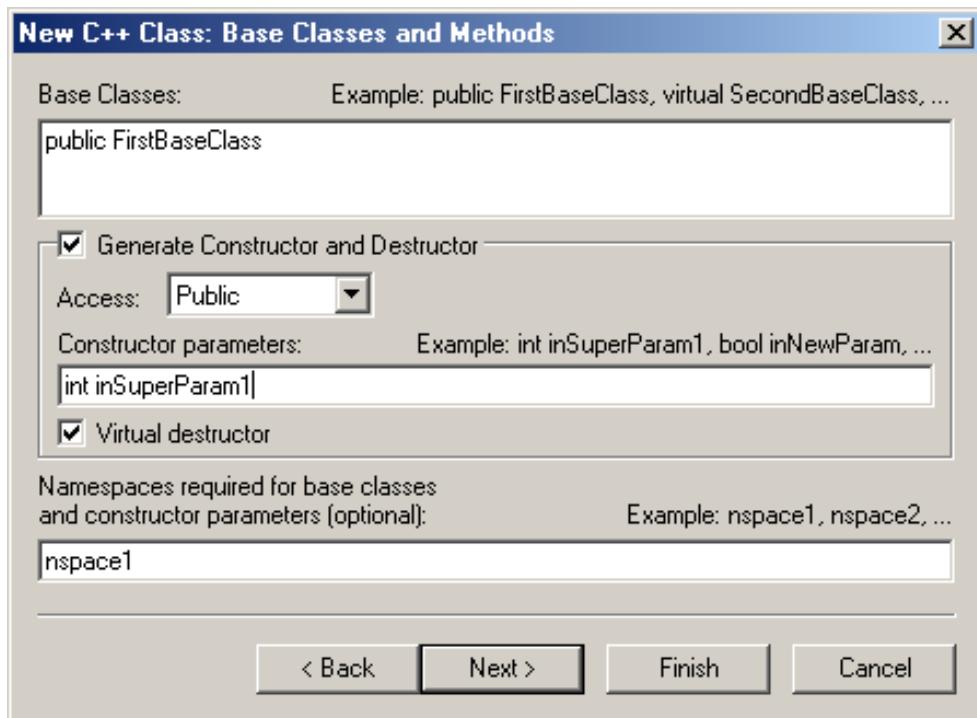
---

**NOTE** If you want to use a separate file to define the members of the new class, type the path to the separate file in the field below the **Use separate file for member definitions** checkbox. Alternatively, click **Existing** to use a standard dialog box to select the file. To create a new, separate file, click **New** and save the new file to a location on your hard disk.

---

4. Click **Next**.

Figure 16.2 New Class wizard—Base Class and Methods



5. Enter **Base Classes and Methods** information.

Enter a list of base classes for the new class:

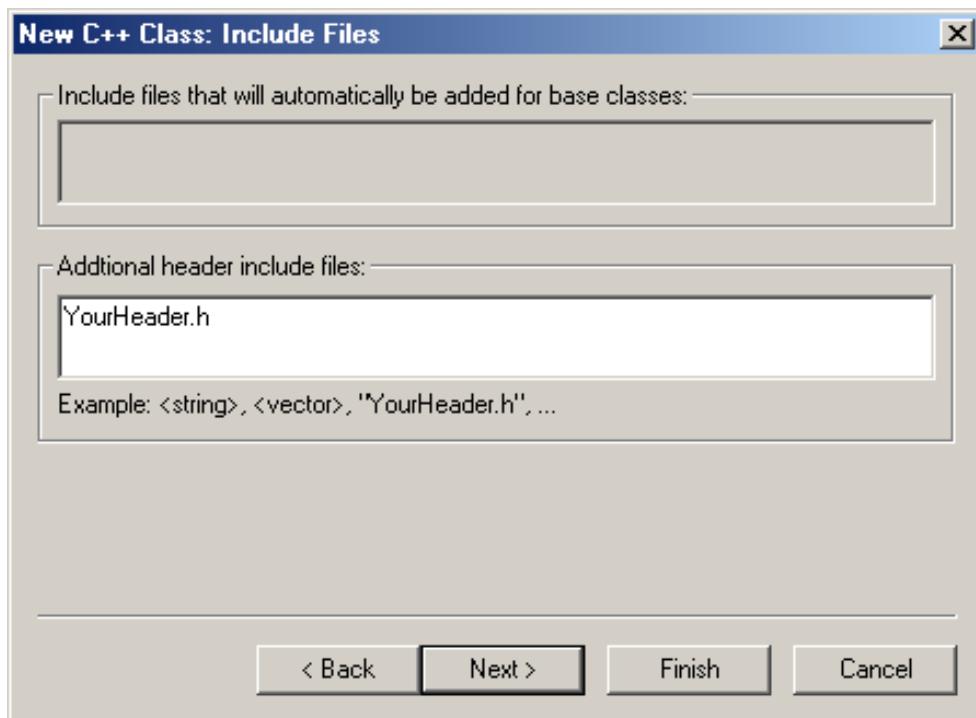
- a. **Access**—Choose an access type, **Public**, **Protected**, or **Private**, for the constructor and destructor from this drop-down menu.
- b. **Constructor parameters**—Enter a list of parameters for the constructor.
- c. **Virtual destructor**—Click this checkbox to create a virtual destructor for the new class.
- d. As an option, you can enter in the **Namespaces required for the base classes and constructor parameters** field the required namespaces for the base classes from the **Base Classes** field, and the constructor parameters in the **Generate Constructor and Destructor** section of the wizard.

Or,

If needed, you can specify the base classes and constructor parameters.

- 
6. Click Next.

**Figure 16.3 New Class wizard—Include Files**



7. Enter **Include Files** information.

Specify additional header #include files for the new class:

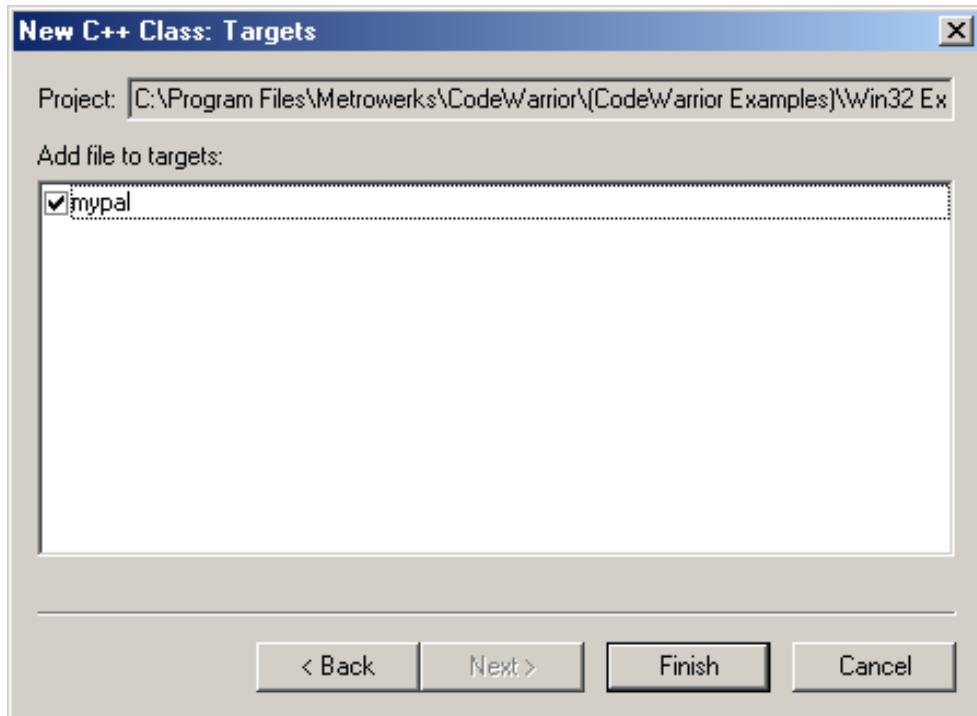
- a. **Include files that will automatically be added for base classes**—This field shows you a list of #include files that the IDE automatically adds to find the base classes.
- b. **Additional header include files**—Enter in this field a list of other include files for the new class in addition to those in the previous field. Separate each file in the list with a comma.

8. Click **Next**.

## Using Browser Wizards

### The New Member Function Wizard

Figure 16.4 New Class wizard—Targets



9. Enter **Targets** information:

Select the checkbox next to the build target's name in the list to add the class files to a specific build target.

10. Click **Finish**.

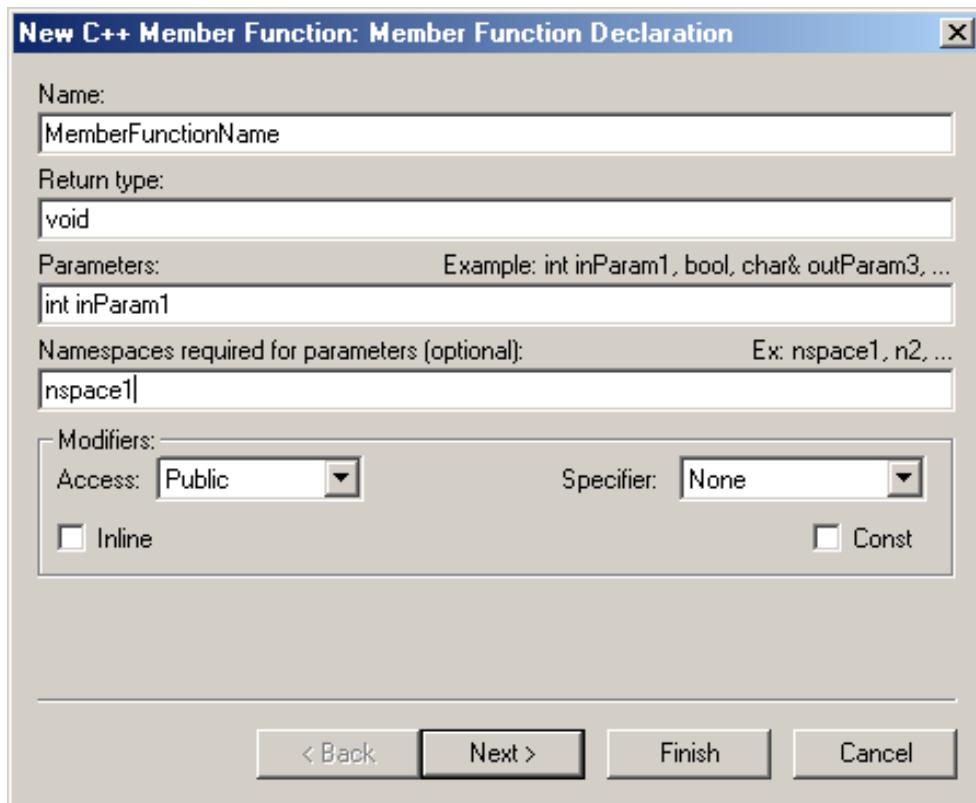
Review the settings summary.

11. Click **Generate**.

## The New Member Function Wizard

Use the **New Member Function** wizard to specify the name, return type, and parameters for a new member function. Enter additional information in the wizard fields to refine the function definition.

**Figure 16.5** New Member Function wizard



---

## Using the New Member Function Wizard

To use the New Member Function wizard, follow these steps:

1. Open the **Class Browser** window, as explained in [Table 16.2 on page 183](#).

**Table 16.2** Opening the Class Browser window

On this host...	Do this...
Windows	Select View > Class Browser.
Macintosh	Select Window > Class Browser.

## Using Browser Wizards

### The New Member Function Wizard

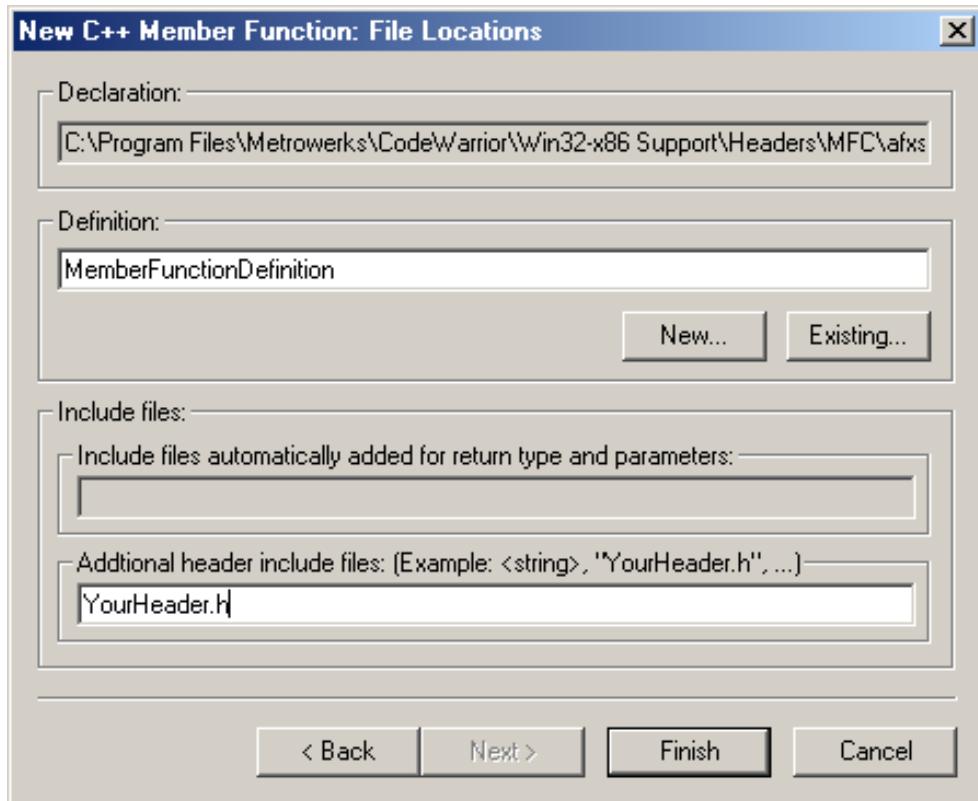
---

**Table 16.2 Opening the Class Browser window (*continued*)**

On this host...	Do this...
Solaris	Select <b>Window &gt; Class Browser</b> .
Linux	Select <b>Window &gt; Class Browser</b> .

2. Select **Browser > New Member Function**.
3. In the **New C++ Member Function** window, enter the **Member Function Declaration**.
  - a. **Name**—Type a name for the member function.
  - b. **Return Type**—Enter an appropriate function return type.
  - c. **Parameters**—Type a list of function parameters.
  - d. **Namespaces required for parameters (optional)**—Type a list of namespaces required for parameters.
4. Click **Next**.

Figure 16.6 New Member Function wizard—File Locations



5. Enter **Member Function File Locations** and **Include Files** information.
6. Click **Finish**.
7. Review settings summary, then click **Generate**.

## The New Data Member Wizard

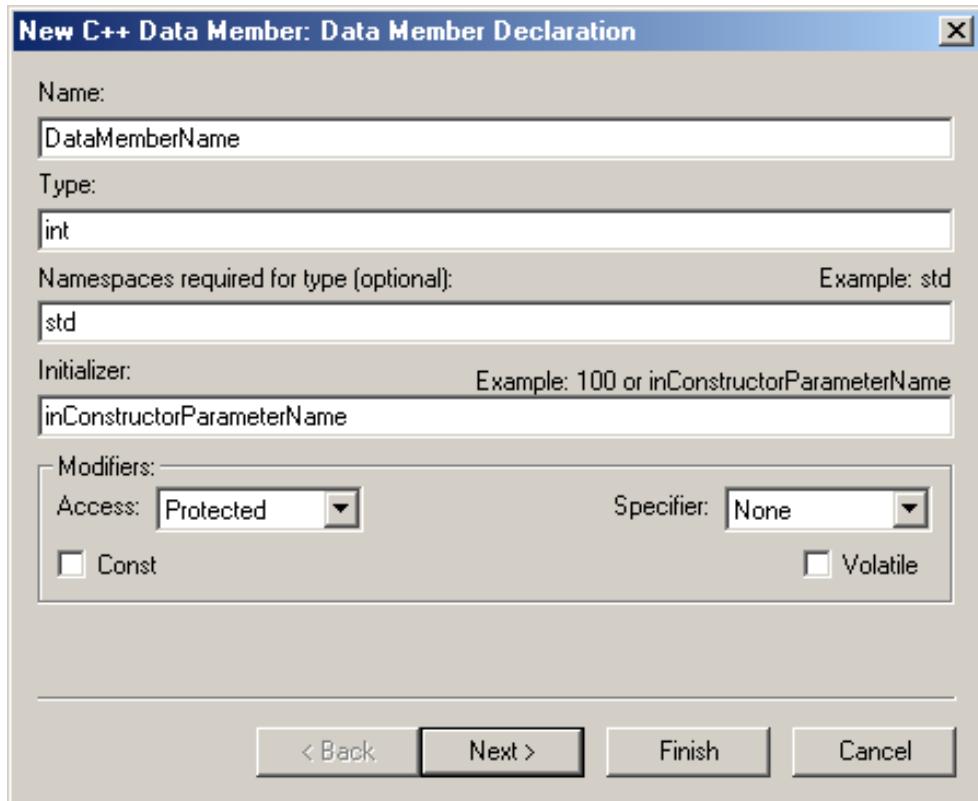
Use the **New Data Member** wizard to define the new data-member declaration, and to specify new data member file locations. The wizard offers additional options to further define the function.

## Using Browser Wizards

### The New Data Member Wizard

---

**Figure 16.7** New Data Member wizard



---

## Using the New Data Member Wizard

To use the New Data Member wizard, follow these steps:

1. Open the **Class Browser** window, as explained in [Table 16.3 on page 186](#).

**Table 16.3** Opening the Class Browser window

On this host...	Do this...
Windows	Select <b>View &gt; Class Browser</b> .
Macintosh	Select <b>Window &gt; Class Browser</b> .

**Table 16.3 Opening the Class Browser window (*continued*)**

On this host...	Do this...
Solaris	Select <b>Window &gt; Class Browser</b> .
Linux	Select <b>Window &gt; Class Browser</b> .

2. Select **Browser > New Data Member**.
3. In the **New C++ Data Member** window, enter the **Name**, **Type**, **Namespaces required for type (optional)**, **Initializer**, and **Modifiers**.
  - a. **Name**—Type a name for the data member in this field.
  - b. **Type**—Enter an appropriate data-member type in this field.
  - c. **Namespaces required for type (optional)**—(Optional) Enter a list of namespaces required for the type in the **Type** field. A sample namespace is `std`.
  - d. **Initializer**—(Optional) Enter an initial value for the data member in this field. Sample initializers are `100` and `inConstructorParameterName`.
  - e. **Modifiers**—Select the access level and type for the new data member.
4. Click **Next**.
5. Specify **Data Member File Locations**.

This section lets you specify file locations associated with the new member functions, including these fields: **Declaration**, **Definition (not available in this wizard)**, **Include file automatically added for member type**, and **Additional header include files**.

- a. **Declaration**—This field shows you the **data member's declaration file location**.
- b. **Definition**—This field is not available in this wizard.
- c. **Include file automatically added for member type**—This field shows you if an include file will be automatically added for the data-member type.
- d. **Additional header include files**—Enter in this field a list of other include files for the new data member, in addition to the file listed in the previous field. Example files are `<string>` and `YourHeader.h`.

6. Click **Finish**.

## **Using Browser Wizards**

### *The New Data Member Wizard*

---

7. Review settings summary, then click **Generate**.

# Debugger

---

This section contains these chapters:

- [Introduction to Debugging](#)
- [Basic Debugging](#)
- [Advanced Debugging](#)
- [Using Breakpoints](#)
- [Using Watchpoints](#)
- [Using Eventpoints](#)



# Introduction to Debugging

---

This chapter explains how to work with the debugger in the CodeWarrior™ IDE. Use the debugger to perform these tasks:

- Find problems in source code—the debugger helps you find errors, or *bugs*, in source code. Bugs prevent a computer program from performing tasks as expected.
- Control program execution—the debugger provides control over a computer program as it executes. Choose to execute an entire program, a single operation, or the individual machine-level instructions that carry out a single operation.
- View memory use in the program—the debugger includes windows that show how a computer program manipulates memory. For example, use a window to view changes to variables used in the program.

This chapter contains these sections:

- [What is a Debugger?](#)
- [What is a Symbolics File?](#)

## What is a Debugger?

A debugger controls program execution and shows the internal operation of a computer program. Use the debugger to find problems while the program executes. Also use the debugger to observe how a program uses memory to complete tasks.

The CodeWarrior debugger provides these levels of control over a computer program:

- Execution of one statement at a time
- Suspension of execution after reaching a specific point in the program
- Suspension of execution after changing a specified memory location

After the debugger suspends program execution, use various windows to perform these tasks:

- View the function-call chain
- Manipulate variable values

- 
- View register values in the computer processor

## What is a Symbolics File?

A symbolics file contains debugging information generated by the IDE for a computer program. The debugger uses this information to provide control over program execution. For example, the debugger uses the symbolics file to find the source code that corresponds to the executing object code of the computer program.

Symbolics files usually contain this information:

- Routine names
- Variables names
- Variable locations in source code
- Variable locations in object code

The IDE supports several types of symbolics files. Some programs generate separate symbolic files, while others do not. For example, when using CodeView on Windows, the IDE places the symbolics file inside the generated binary file.

# Basic Debugging

---

This chapter explains how to perform basic debugging tasks in the CodeWarrior™ IDE. Basic debugging tasks take place in these windows:

- Thread window—use this window to control program execution, manipulate memory contents, and view source code.
- Message window—use this window to view operation results and general error messages.

This chapter contains these sections:

- [“Thread Window” on page 193](#)
- [“Common Debugging Actions” on page 195](#)

## Thread Window

The debugger suspends the execution of processes in a computer program. The Thread window displays information about a suspended process during a debugging session. Use the Thread window to perform these tasks:

- View the call chain for a routine
- View routine variables, both local and global
- View a routine in source, assembler, or mixed-mode code

[Figure 18.1 on page 194](#) shows the Thread window. [Table 18.1 on page 194](#) explains the items in the window.

## Basic Debugging

### Thread Window

Figure 18.1 Thread window

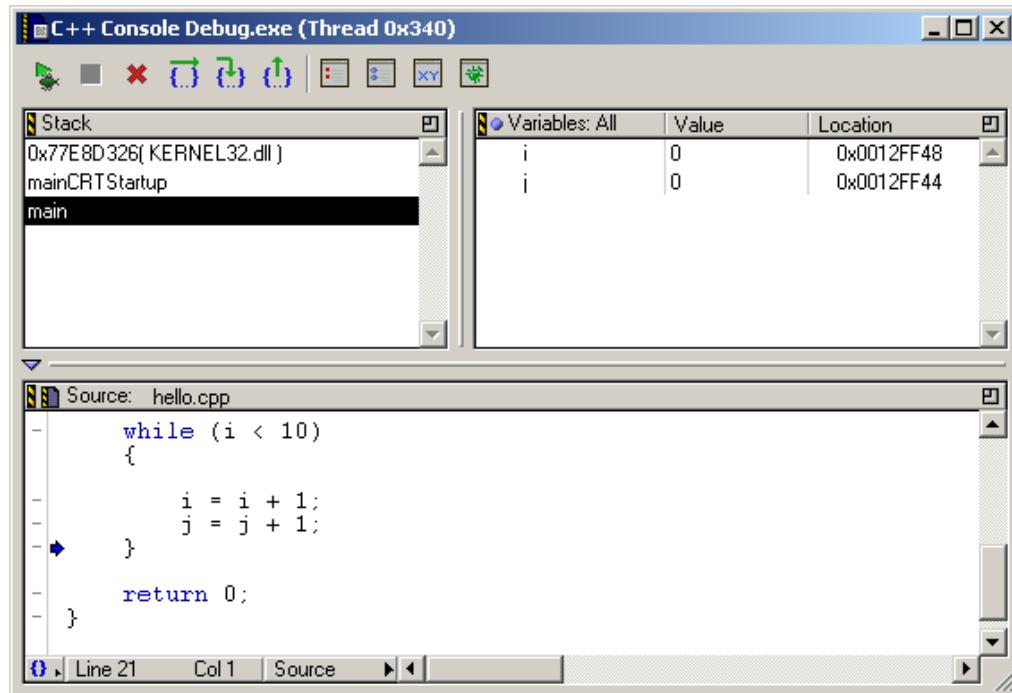


Table 18.1 Thread window—items

Item	Explanation
Debugger toolbar	Controls the current debugging session
Stack pane	Displays the current subroutine calling chain, with the most current routine name at the bottom
Variables pane	Displays the local and global variables used by the current routine. Click the Variables Pane Listing icon ( ) to switch between these display states: <ul style="list-style-type: none"><li>All—show all local and global variables in the code</li><li>Auto—show only the local variables of the routine to which the current-statement arrow points</li><li>None—show no variables. Use this type to improve stepping performance for slow remote connections.</li></ul>
Source pane	Displays the source files in the current build target, including any comments and white space.

**Table 18.1 Thread window—items (*continued*)**

Item	Explanation
Functions list box	Displays a list of functions declared in the file.
Line and Column Indicator	Displays the current line and column number of the text-insertion cursor.
Source list box	Select how to display the source code in the Sources pane. The options consist of: <ul style="list-style-type: none"> <li>• Source—source code appears in the programming-language instructions used to create it</li> <li>• Assembler—source code appears in the assembly-language instructions used to implement each line</li> <li>• Mixed—source code alternates with the assembly-language instructions that implement each line of code</li> </ul>

## Common Debugging Actions

Using the debugger involves a common set of debugging tasks. Use this set of tasks to correct source-code errors, control program execution, and observe memory behavior:

- Start the debugger.
- Step into, out of, or over routines.
- Stop, resume, or kill program execution.
- Run the program.
- Restart the debugger.

---

### Starting the Debugger

Use the **Debug** command to begin a debugging session. The debugger takes control of program execution, starting at the main entry point of the program.

1. Click **Debug**  on the debugger toolbar.
  
2. The debugger starts a new debugging session and opens a new Thread window.

## Basic Debugging

### Common Debugging Actions

---

<b>NOTE</b>	Some projects require additional configuration before the debugging session can begin. If needed, the IDE prompts for permission to perform this configuration automatically.
-------------	---

---

## Stepping Into a Routine

Use the **Step Into** command to execute one source-code statement at a time and follow execution into a routine call. The command follows execution into called routines.

1. Step Into a routine.
  - Choose **Debug > Step Into**, or
  - Click **Step Into**  on the debugger toolbar.
2. The debugger executes one statement. The current-statement arrow moves to the next statement:
  - If the current statement does not call a routine, the current-statement arrow moves to the next statement in the source code.
  - If the current statement calls a routine, program execution enters the called routine. The current-statement arrow moves to the next statement in the called routine.

---

## Stepping Out of a Routine

Use the **Step Out** command to execute the rest of the current routine and stop program execution after the routine returns to its caller. This command causes execution to return up the calling chain.

1. Step out of a routine.
  - Choose **Debug > Step Out**, or
  - Click **Step Out**  on the debugger toolbar.
2. The current routine executes and returns to its caller, then program execution stops.

## Stepping Over a Routine

Use the **Step Over** command to navigate linearly through the source code, executing each statement along the way. Use this command to observe a program as it executes each line of code. The command executes a routine call without displaying its lines in the Thread window.

1. Step over a routine call.
  - Choose **Debug > Step Over**, or
  - Click **Step Over**  on the debugger toolbar.
2. Program execution continues through the routine call.

---

<b>NOTE</b>	After program execution steps over code and reaches the end of a routine, the current-statement arrow points to the next statement that executes.
-------------	---

---

## Stopping Program Execution

Use the **Break** or **Stop** commands to suspend program execution during a debugging session.

1. Stop program execution.
  - (Windows) Choose **Debug > Break**, or
  - (Macintosh, Solaris, and Linux) Choose **Debug > Stop**, or
  - Click **Stop**  on the debugger toolbar.
2. The operating system surrenders control to the debugger, which stops program execution.

## Resuming Program Execution

Use the **Resume** command to continue executing a suspended debugging session. If the debugging session is already active, use this command to switch view from the Thread window to the executing program.

---

**NOTE** The Resume command appears only for those platforms that support it.

---

1. Resume the program.
  - Choose **Project > Resume**, or
  - Click **Debug**  on the debugger toolbar.
2. The stopped session continues, or the view changes to the running program.

---

## Killing Program Execution

Use the **Kill** command to completely terminate program execution and end the debugging session. This behavior differs from stopping a program, as stopping temporarily suspends execution.

1. Kill program execution.
  - Choose **Debug > Kill**, or
  - Click **Kill** 
2. The debugger terminates program execution and ends the debugging session.

---

## Running a Program

Use the **Run** command to execute a program normally, without debugger control.

1. Run the project.

- Choose **Project > Run**, or
  - Click **Run**  on the debugger toolbar.
2. The project runs.

---

## Restarting the Debugger

Use the **Restart** command after stopping program execution. The debugger goes back to the beginning of the program and begins execution again. This behavior is equivalent to killing execution, then starting a new debugging session.

1. Choose **Debug > Restart**.
2. The debugger returns to the beginning of the program and begins controlled execution.

## **Basic Debugging**

*Common Debugging Actions*

---

# Advanced Debugging

---

This chapter explains how to perform advanced debugging tasks in the CodeWarrior™ IDE. This is a sample list of advanced tasks:

- Use the Processes window—the Processes window shows individual processes and tasks that the debugger can control.
- Use the Expressions window—the Expressions window lets you monitor and edit frequently used variables, structure members, and array elements.
- Use the Register window—the Register window lets you monitor and edit register contents of a processor.
- Use the Memory window—the Memory window lets you monitor and edit memory usage of an application.

This chapter contains these sections:

- [“What is Advanced Debugging?” on page 202](#)
- [“Symbol Hint” on page 202](#)
- [“Contextual Menus” on page 203](#)
- [“Symbolics Window” on page 205](#)
- [“Processes Window” on page 208](#)
- [“Expressions Window” on page 210](#)
- [“Global Variables Window” on page 213](#)
- [“Registers Window” on page 214](#)
- [“Log Window” on page 218](#)
- [“Variable Window” on page 220](#)
- [“Array Window” on page 221](#)
- [“Memory Window” on page 223](#)
- [“Multi-core Debugging” on page 228](#)

# What is Advanced Debugging?

In addition to controlling program execution, the IDE includes several windows for handling advanced program elements:

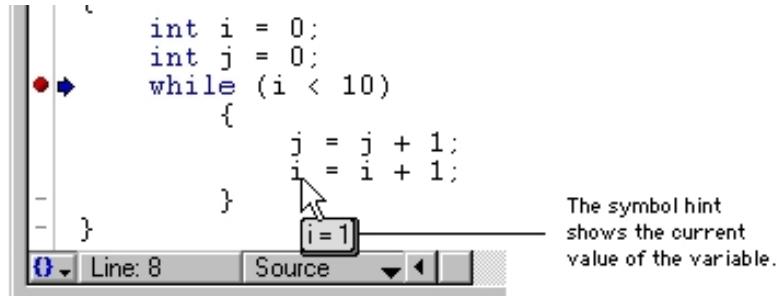
- symbolics files—these files contain information generated by the IDE for a project. The debugger uses symbolics files to display information about program structure and to control that structure during execution.
- processes—the debugger can control most processes associated with the running program as well as manually assigned processes.
- expressions—the debugger can manipulate various expressions and their changing values during program execution.
- variables—the debugger can monitor and modify global and local variables used during program execution.
- registers—the debugger provides views and manipulation of some register contents.
- memory—the debugger can manipulate memory structures, such as array elements, used during program execution.

## Symbol Hint

The symbol hint shows information about variable values. This information appears automatically while the debugger is active.

Select the [Show variable values in source code](#) option in the [Display Settings](#) preference panel to use the symbol hint.

**Figure 19.1 Symbol hint**



## Toggling the Symbol Hint

Enable the symbol hint to view information about program variables in source views.

1. Select **Edit > Preferences**.  
The IDE Preferences window appears.
2. Select **Display Settings** in the IDE Preference Panels list.  
The Display Settings preference panel appears.
3. Toggle **Show variable values in source code**:
  - Selected—enables the symbol hint
  - Cleared—disables the symbol hint
4. Click **Apply** or **Save** to confirm your changes to the preference panel.
5. Close the IDE Preferences window.

---

## Using the Symbol Hint

During a debugging session, use the symbol hint to view information about program variables.

1. Rest the cursor over a variable in a source view.
2. After a brief pause, the symbol hint appears and shows the current variable value.

## Contextual Menus

The contextual menu provides a shortcut to frequently used menu commands. The available menu commands change, based on the context of the selected item.

Sample uses of the contextual menu for debugging tasks include:

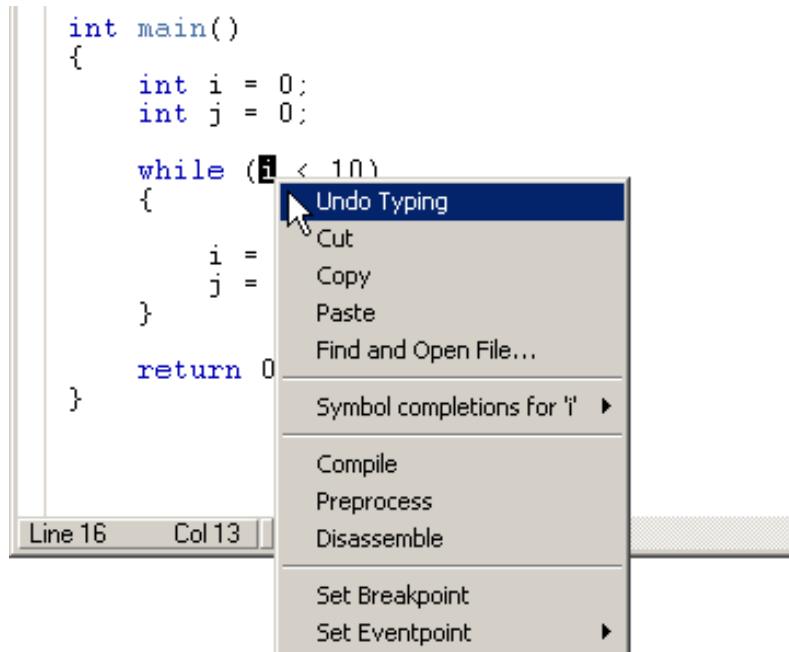
- Changing the format of variables displayed in variable panes or the General Registers window
- Manipulating breakpoints and the program counter in source panes

- Viewing memory in separate windows

**TIP** Experiment using the contextual menu in various IDE windows to discover additional features.

---

**Figure 19.2 Contextual menus**



---

## Using Contextual Menus

Use contextual menus to more conveniently apply context-specific menu commands to selected items.

- Open the contextual menu.
- Windows: Right-click an item.
- Macintosh: Control-click or click and hold on an item.
- Solaris: Ctrl-click or click and hold on an item.

- Linux: Ctrl-click or click and hold on an item.

The contextual menu appears, displaying menu commands applicable to the selected item.

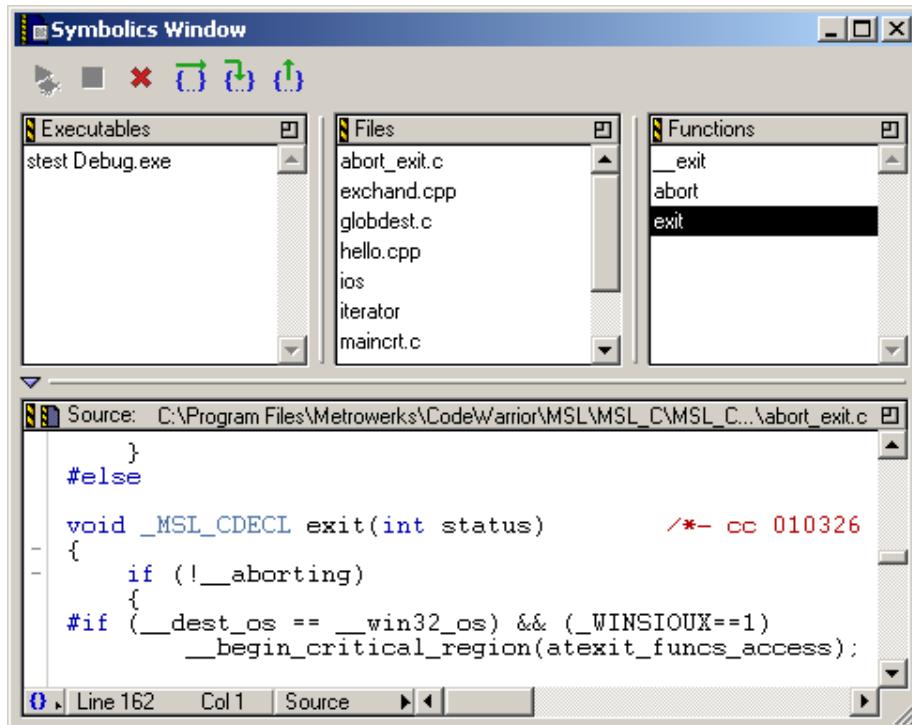
## Symbolics Window

The **Symbolics** window displays information that the IDE generates for the active file. Symbolics information includes program variables, functions, data structures, and source files.

Select the **Activate Browser** option in the **Build Extras** target settings panel in order to generate the symbolics information during the next build or debugging session.

[Figure 19.3](#) shows the Symbolics window. [Table 19.1 on page 206](#) explains the items in the window.

**Figure 19.3** Symbolics window



**Table 19.1 Symbolics window—items**

Item	Icon	Explanation
Debugger toolbar		Contains buttons that represent common debugging commands, such as stopping the program and stepping through code.
Executables pane		Lists recently used executable files that contain symbolics information.
Files pane		Lists source files in the build target being debugged, for the selected executable file.
Functions pane		Lists the functions declared in the file selected in the Files pane.
Source pane		Displays the source code in the file selected in the Files pane.

---

## Opening the Symbolics Window

The Symbolics window displays information generated by the IDE for a file.

To open the Symbolics window, do one of these tasks:

- Select **View > Symbolics** or **Window > Symbolics window**.
- Open a symbolics file. The IDE typically appends .xSYM or .iSYM, to the names of these files.
- Open an executable file for which the IDE previously generated symbolics information. The IDE typically appends .exe or .app to the names of the files.



Alternatively, click the Symbolics button in the Thread window toolbar to open the Symbolics window.

---

## Using the Executables Pane

The **Executables** pane lists recently opened executable files for which the IDE generated symbolics information.

To use the pane, select an executable file in the list. The Files pane updates to display information for the selected executable file.

## Using the Files Pane

The **Files** pane lists the source files in the build target being debugged, for the selected executable file in the Executables pane.

To use the pane, select a file in the list. The Functions pane and Source pane update to display information for the selected file.

---

## Using the Functions Pane

The **Functions** pane lists functions declared in the selected file in the Files pane.

---

**TIP**      The [\*\*Sort functions by method name in symbolics window\*\*](#) option changes the order in which the Functions pane lists functions.

---

To use the pane, select a function in the list. The Source pane updates to display source code for the selected function.

---

## Using the Source Pane

The **Source** pane displays source code for the selected function in the Functions pane, using the fonts and colors specified in the IDE Preferences window.

To use the pane, select a function in the Functions pane. The corresponding source code appears in the Source pane.

If the selected function does not contain source code, the Source pane displays the message **Source text or disassembly not available**.

---

**NOTE**      Use the Source pane in the Symbolics window to view source code, copy source code, and set breakpoints. Use an editor window to modify the source code. Use a Thread window to view the currently executing statement.

---

# Processes Window

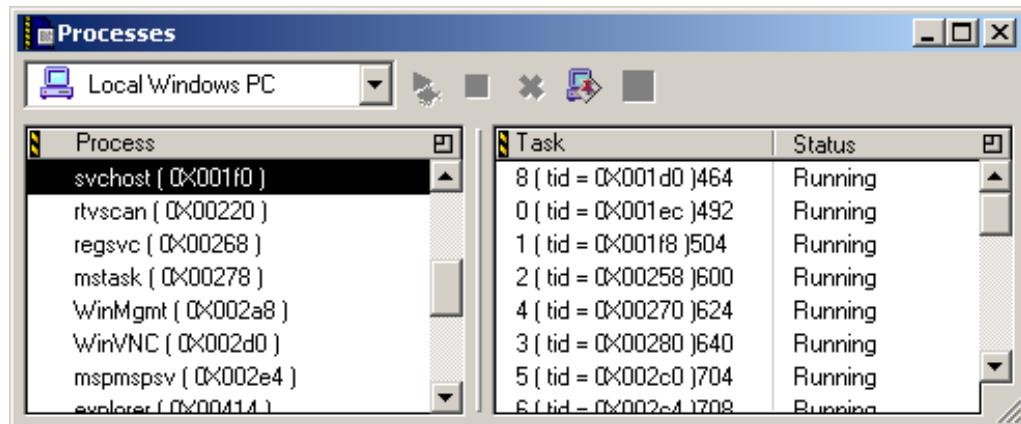
The **Processes** window manipulates processes on the host computer:

- running processes
- tasks for selected processes
- some hidden processes

The debugger also controls manually added processes.

[Figure 19.4](#) shows the Processes window. [Table 19.2](#) explains the items in the window.

**Figure 19.4** Processes window



**Table 19.2** Processes window—items

Item	Icon	Explanation
Processes selector	Local Windows PC	Select an item for which to display active processes.
Debugger toolbar		Toolbar buttons represent common debugging commands, such as debugging the program and killing execution.
Attach to Process		Click to use the debugger to control the selected process in the Process pane.

**Table 19.2 Processes window—items (*continued*)**

Item	Icon	Explanation
Stack Crawl window		Click to open a Thread window for the selected process in the Process pane.
Process pane		Lists active processes for the selected machine.
Task pane		Lists the status of tasks for a selected process in the Process pane.

---

## Opening the Process Window

Use the **Process Window** command to view and manipulate active processes on a selected machine.

---

<b>NOTE</b>	The Processes window appears only for those platforms that support it.
-------------	--

---

[Table 19.3](#) explains how to open the Processes window.

**Table 19.3 Opening the Processes window**

On this host...	Do this...
Windows	Select <b>View &gt; Processes</b> .
Macintosh	Select <b>Window &gt; Processes Window</b> .
Solaris	Select <b>Window &gt; Processes Window</b> .
Linux	Select <b>Window &gt; Processes Window</b> .

---

## Using the Process Pane

Use the **Process** pane to view active processes on a selected machine. Processes under debugger control appear in bold or with a check mark.

1. Use the Processes selector to select the item for which to view active processes.

2. Select a process in the Process pane.

The Task pane displays all tasks assigned to the selected process.

---

## Using the Task Pane

Use the **Task** pane to view tasks under debugger control. Select a process in the Process pane to display its tasks under debugger control in the Task pane. Double-click a task to open it in a new Thread window, or choose the task name and click the Stack Crawl Window button.

---

## Attaching the Debugger to a Process

Click the **Attach to Process** button to assign a selected process to a new debugging session. This assignment allows the debugger to control processes that it does not otherwise recognize. For example, you can click the Attach to Process button to assign dynamic link libraries or shared libraries to the debugger.

1. Use the Processes selector to select the item for which to view active processes.
2. Select a process to attach to the debugger.

3. Click Attach to Process  .

The debugger assumes control of the selected process. Processes under debugger control appear in bold or with a check mark.

## Expressions Window

The **Expressions** window helps you monitor and manipulate these kinds of items:

- global and local variables
- structure members
- array elements

[Figure 19.5 on page 211](#) shows the Expressions window. [Table 19.4 on page 211](#) explains the items in the window.

Figure 19.5 Expressions window

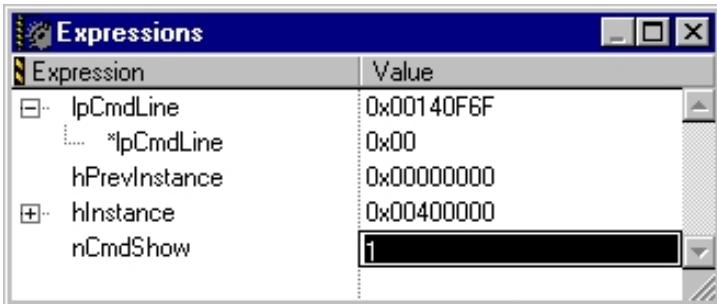


Table 19.4 Expressions window—items

Item	Explanation
Expression column	Lists expressions and expression hierarchies. Click the hierarchical controls to expand or collapse the expression view.
Value column	Shows the current value of each corresponding expression in the Expression column. Double-click a value to change it.

## Opening the Expressions Window

Use the Expressions window to inspect frequently used variables as their values change during a debugging session.

To open the Expressions window, select **View > Expressions** or **Window > Expressions Window**.



Alternatively, click the Expressions button in the Thread window toolbar to open the Expressions window.

## Adding Expressions

The Expressions window handles various ways of adding expressions for inspection.

To add an expression to the Expressions window, do this:

- Select the desired expression and choose **Data > Copy to Expression**, or
- Use the contextual menu with a selected expression, or

- Drag and drop an expression from another window into the Expressions window.

The Expressions window updates to reflect the added expression. Drag expressions within the window to reorder them.

---

## **Adding a Constant Value to a Variable**

You can enter an expression in the Expressions window that adds a constant value to a variable. Suppose `x` is a short integer type in the variable context of some function scope in C++ code. You can enter the expression `x+1` and the IDE computes the resulting value just as you would compute it on a calculator.

1. Select the variable to which you want to add a constant value.

For example, select `x`.

2. Enter an expression that adds a constant value to the variable.

For example, append `+1` to `x` so that the resulting expression is `x+1`.

The IDE adds the constant value to the variable and displays the result in the Expressions window.

---

## **Making a Summation of Two Variables**

You can enter an expression in the Expressions window that computes the sum of two variables. Suppose `x` is a short integer type in the variable context of some function scope in C++ code. You can enter the expression `x+y` and the IDE computes the resulting value just as you would compute it on a calculator.

1. Select the variable to which you want to add another variable.

For example, select `x`.

2. Enter an expression that adds a second variable to the first variable.

For example, append `+y` to `x` so that the resulting expression is `x+y`.

The IDE computes the sum of the two variables and displays the result in the Expressions window.

---

## Removing Expressions

The Expressions window handles various ways of removing expressions that no longer require inspection.

To remove an expression from the Expressions window, do this:

- Select the expression to remove and choose **Edit > Delete** or **Edit > Clear**, or
- Select the expression to remove and press the Backspace or Delete key.

The Expressions window updates to reflect the removed expression.

---

**NOTE** Unlike the Variable window, the Expressions window does not remove a local variable after program execution exits the routine that defines the variable.

---

## Global Variables Window

The **Global Variables** window shows all global and static variables used by the executing program. Use this window to observe changes in variable values as the program executes.

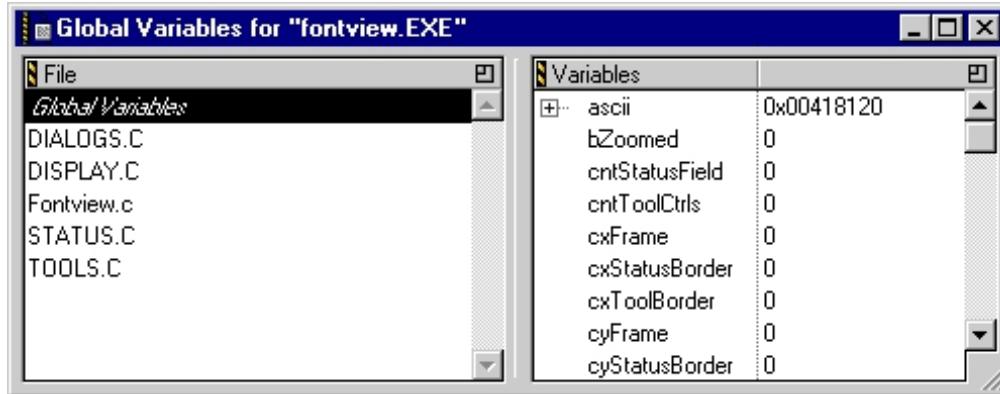
---

**TIP** Use multiple Variable windows to observe global variables in separate windows.

---

[Figure 19.6 on page 214](#) shows the Global Variables window. [Table 19.5 on page 214](#) explains the items in the window.

**Figure 19.6 Global Variables window**



**Table 19.5 Global Variables window—items**

---

Item	Explanation
File	Lists source files that declare global or static variables. Click a source file to view its static variables. Click <b>Global Variables</b> to view all global variables declared in the program.
Variables	Lists variables according to the file selected in the File pane. Double-click a variable to display it in a separate Variable window.

---

## Opening the Global Variables Window

Use the Global Variables window to display global variables declared in a program or static variables declared in the source files the comprise the program.

To open the Global Variables window, select **View > Global Variables** or **Window > Global Variables Window**.

## Registers Window

The **Registers** window gives you a hierarchical view of these register types:

- general registers—contents of the central processing unit (CPU) of the host computer

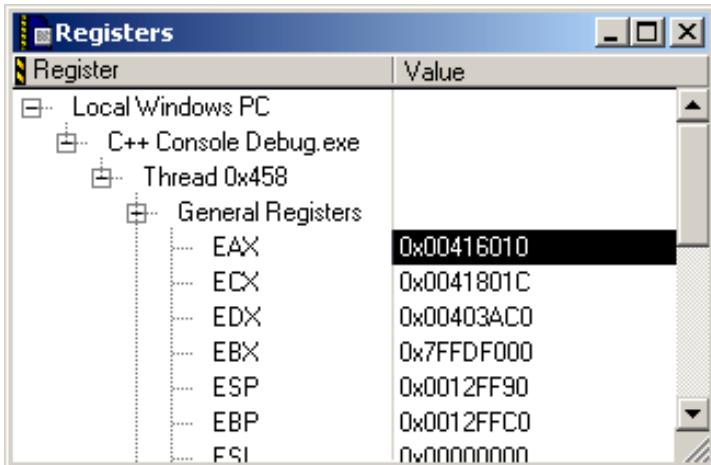
- floating-point unit (FPU) registers—contents of the FPU registers
- registers specific to the host computer

You can use the Register window to perform these tasks:

- expand the hierarchical items in the window and view their contents
- select and modify register values
- view documentation for individual registers (depending on the register)

[Figure 19.7](#) shows a sample Registers window.

**Figure 19.7 Registers Window**



## General Registers

The **General Registers** are the register contents of the central processing unit (CPU) of the host computer. The exact listing of these registers depends on the host CPU and on the current build target. See the *Targeting* documentation for additional information.

## FPU Registers

The **FPU Registers** are the register contents of the floating-point unit (FPU) of the host computer. The exact listing of these registers depends on the host FPU and on the current build target. See the *Targeting* documentation for additional information.

## Host-specific Registers

The Registers window also lists additional register contents for registers specific to the host computer. The exact listing of these registers depends on the host computer and on the current build target. See the *Targeting* documentation for additional information.

---

### Opening the Registers Window

Open the **Registers** window to inspect and modify various register contents.

[Figure 19.7](#) explains how to open the Registers window.

**Table 19.6 Opening the Registers window**

On this host...	Do this...
Windows	Select <b>View &gt; Registers</b> .
Macintosh	Select <b>Window &gt; Registers Window</b> .
Solaris	Select <b>Window &gt; Registers Window</b> .
Linux	Select <b>Window &gt; Registers Window</b> .

---

### Viewing Registers

View registers to inspect and modify their contents.

1. Open the **Registers** window.
2. Expand the hierarchical list to view register groups.  
Expanding the list shows the register groups that you can view or change.
3. Expand a register group.  
Expanding a group shows its contents, by register name and corresponding value.

## Changing Register Values

Change register values during program execution in order to examine program behavior.

1. Open the **Registers** window.
2. Expand the hierarchical list to view the names and corresponding values of the register that you want to modify.
3. Double-click the register value that you want to change.  
The value highlights.
4. Enter a new register value.
5. Press **Enter** or **Return**.

The register value changes.

---

## Changing Register Data Views

Change register data views to see register contents in a different format. For example, you can change the view of a register from binary format to hexadecimal format.

1. Open the **Registers** window.
2. Expand the hierarchical list to view the names and corresponding values of the register that you want to view in a different format.
3. Select the register value that you want to view in a different format.  
The value highlights.
4. Select **Data > View as *format***, where *format* is the data format in which you want to view the register value.  
The available formats depend on the selected register value.
5. The register value changes format.

6. Select **Data > View as Default** to restore the original data format.

Alternatively, you can use a contextual menu to change the data format, as explained in [Table 19.7 on page 218](#).

**Table 19.7 Changing the data format by using a contextual menu**

On this host...	Do this...
Windows	Right-click the register value and select <b>View as format</b> .
Macintosh	Control-click the register value and select <b>View as format</b> .
Solaris	Click and hold on the register value and select <b>View as format</b> .
Linux	Click and hold on the register value and select <b>View as format</b> .

---

## Opening Registers in a Separate Registers Window

Open registers in a separate Register Window to narrow the scope of registers that appear in a single window.

1. Open the **Registers** window.
2. Expand the hierarchical list to view the register or register group that you want to view in a separate Registers window.
3. Double-click the register or register group.
4. A new Registers window opens.

The new Registers window lists the name and value of the register that you double-clicked, or the names and values of the register group that you double-clicked.

# Log Window

The **Log** window displays messages during program execution. Select the **Log System Messages** option in the **Debugger Settings** panel to activate the Log window.

The IDE allows saving Log window contents to a .txt (text) file and copying text from the Log window to the system clipboard.

Windows-hosted Log window messages include:

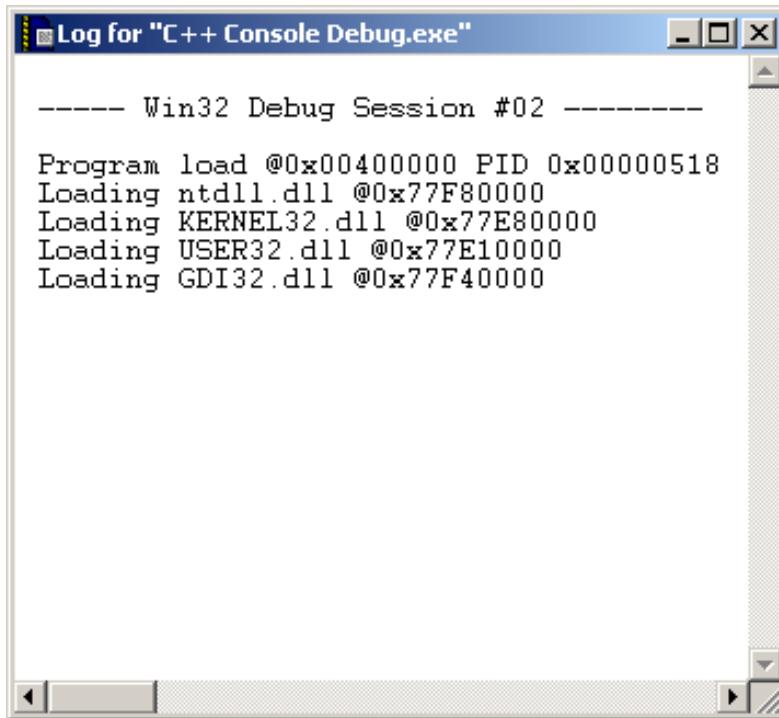
- Dynamic Link Library (DLL) loading and unloading
- debugging `printf()` messages

Macintosh-hosted Log window messages include:

- PowerPC™ code fragments
- `DebugStr()` messages

[Figure 19.8](#) shows the Log window.

**Figure 19.8 Log window**



---

## Opening the Log Window

Use the **Debugger Settings** preference panel to enable message logging. The Log window records these types of messages for a program during a debugging session:

- the start of new tasks
  - routine entry and exit
  - Windows: DLL loading and unloading, and debug printf() messages
  - Macintosh: PowerPC code-fragment loading and DebugStr() messages
1. Select the **Log System Messages** option in the **Debugger Settings** preference panel.
  2. Select **Project > Debug**.

The Log window appears. It allows selecting, copying, and saving logged text to a file for later analysis. See the *Targeting* documentation for additional information.

## Variable Window

A Variable window allows manipulation of a single variable used in source code. For a local variable, the window closes after program execution exits the routine that defines the variable.

[Figure 19.9](#) shows the Variable window.

**Figure 19.9 Variable window**



---

## Opening a Variable Window

A Variable window manipulates a single variable or variable hierarchy. A Variable window containing a local variable closes after program execution exits the routine that defines that variable.

1. Select a variable in any window pane that lists variables.

2. Open a Variable window:

- Select **Data > View Variable**, or
- Double-click the variable.

A Variable window appears. Double-click a value to change it.

---

**TIP**

Use Variable windows to monitor individual variables independently of other windows. For example, use a Variable window to continue monitoring a variable that leaves the current scope of program execution in the Thread window.

---

Alternatively, use a contextual menu to open a variable window, as explained in [Table 19.8](#).

**Table 19.8 Opening a Variable window by using a contextual menu**

On this host...	Do this...
Windows	Right-click the variable and select <b>View Variable</b> .
Macintosh	Control-click the variable and select <b>View Variable</b> .
Solaris	Click and hold on the variable, then select <b>View Variable</b> .
Linux	Click and hold on the variable, then select <b>View Variable</b> .

## Array Window

An Array window allows manipulation of a contiguous block of memory, displayed as an array of elements. The window lists array contents sequentially, starting at element 0.

The Array window title shows the base address bound to the array. The base address can bind to an address, a variable, or a register. An array bound to a local variable closes after the routine that defines the variable returns to the calling routine.

For array elements cast as structured types, a hierarchical control appears to the left of each element. Use these hierarchical controls to expand or collapse the display of each element's contents.

[Figure 19.10](#) shows an Array window. [Table 19.9](#) explains the items in the window.

Figure 19.10 Array window

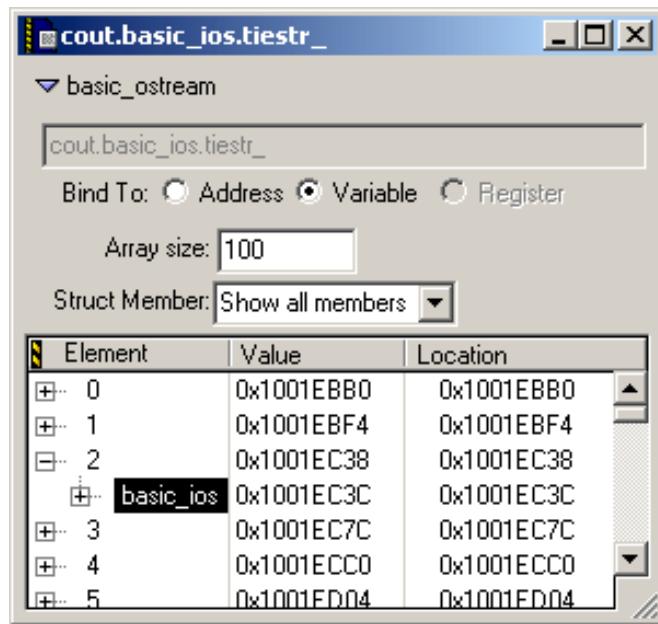


Table 19.9 Array window—items

Item	Icon	Explanation
Hierarchical control	▼	Click to collapse the view of the information pane.
Bind To		Select the base address of the array: <b>Address</b> , <b>Variable</b> , or <b>Register</b> .
Array size	Array size: 100	Enter the number of elements to display in the Array window.
Struct Member	Struct Member: Show all members	Select a specific member to show in each element, or show all members.
Element		Shows the array elements in a hierarchical list.
Value		Shows the value of each array element.
Location		Shows the address in memory of each array element.

## Opening an Array Window

Use the **View Array** command to manipulate a contiguous memory block as an array of elements in an Array window.

1. Select the array that you want to view.
2. Select **Data > View Array**.

A new Array window appears.

<b>TIP</b>	Drag and drop a register name or variable name to an Array window to set the base address. Use the <b>View Memory As</b> command to interpret memory displayed in an Array window as a different type.
------------	--

Alternatively, use a contextual menu to open an Array window, as explained in [Table 19.10 on page 223](#).

**Table 19.10 Opening an Array window by using a contextual menu**

On this host...	Do this...
Windows	Right-click the array and select <b>View Array</b> .
Macintosh	Control-click the array and select <b>View Array</b> .
Solaris	Click and hold on the array, then select <b>View Array</b> .
Linux	Click and hold on the array, then select <b>View Array</b> .

## Memory Window

The Memory window manipulates program memory contents in various data types. Use this resizable window to perform these tasks:

- View memory
- Change individual memory bytes
- Set watchpoints

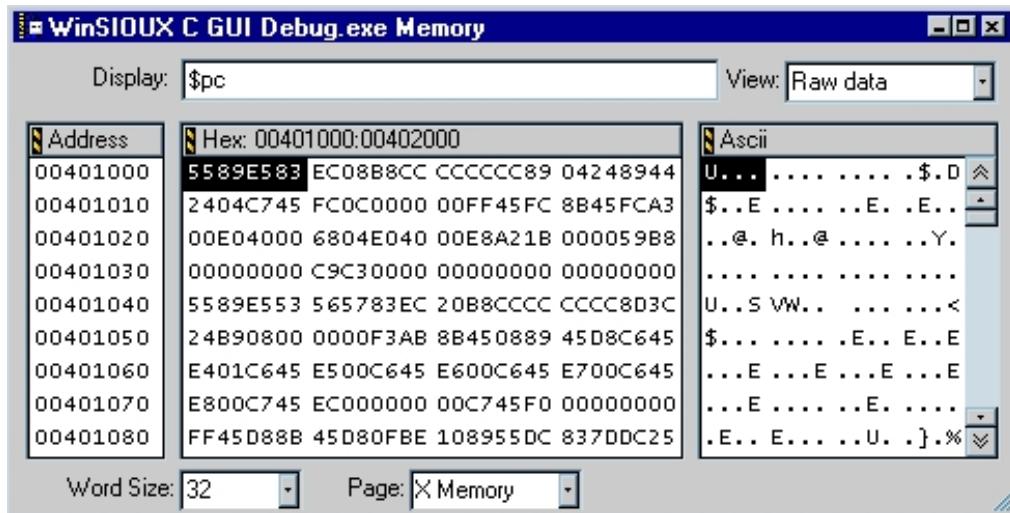
---

**CAUTION** Arbitrarily changing memory contents could degrade the stability of the IDE, another program, or the operating system itself. Understand the consequences of manipulating memory.

---

[Figure 19.11](#) shows the Memory window. [Table 19.11 on page 224](#) explains the items in the window.

**Figure 19.11** Memory window



**Table 19.11** Memory window—items

Item	Icon	Explanation
Display		Enter a symbol representing the starting address of memory to display. Valid symbols include addresses and non-evaluating expressions, such as <code>main</code> or <code>x</code> .
View	<input type="button" value="Raw data"/>	Select the data format in which to view memory contents.
Memory Space (for processors that support multiple memory spaces)		Choose the memory space in which to view selected variables or source code.
Previous Memory Block		Click to view the preceding block of memory.

Table 19.11 Memory window—items (*continued*)

Item	Icon	Explanation
Next Memory Block		Click to view the succeeding block of memory.
Address		Displays a contiguous range of memory addresses, beginning with the address entered in the <b>Display</b> field.
Hex		Displays a hexadecimal representation of the memory addresses shown in the <b>Address</b> pane.
Ascii		Displays an ASCII representation of the memory addresses shown in the <b>Address</b> pane.
Word Size	Word Size: <input type="text" value="32"/>	Select the bit size of displayed words.
Page (for processors that support multiple pages)	Page: <input type="text" value="X Memory"/>	Select the memory-space page in which to view source code.

## Viewing and Changing Raw Memory

Use the **View Memory** command to view and change the raw contents of memory.

1. Select an item or expression that resides at the memory address to be examined.
2. Choose **Data > View Memory**.

A new Memory window appears.

3. Select **Raw data** from the **View** list pop-up.

The contents of memory at the selected location appears in both hexadecimal and ASCII.

Scroll through memory by selecting the **Address, Hex, or ASCII** pane of the Memory window and then using the up and down arrow keys. Display a different memory location by changing the expression in the **Display** field.

Change the word size displayed in the Memory window by using the **Word Size** list pop-up. The choices are 8, 16, and 32 bits.

Change the contents of a particular memory location by double-clicking on that location in either the hexadecimal or ASCII pane of the Memory window.

Replace the current value by entering a hexadecimal value in the **Hex** pane or a string of ASCII characters in the **ASCII** pane.

Alternatively, use a contextual menu to view and change memory, as explained in [Table 19.12](#).

**Table 19.12 Opening a Memory window by using a contextual menu**

On this host...	Do this...
Windows	Right-click the item and select <b>View Memory</b> .
Macintosh	Control-click the item and select <b>View Memory</b> .
Solaris	Click and hold on the item, then select <b>View Memory</b> .
Linux	Click and hold on the item, then select <b>View Memory</b> .

---

## **Viewing Memory Referenced by a Pointer**

Use the **View Memory** command to inspect the memory referenced by a pointer—including an address stored in a register.

1. Select a pointer in a source window.
2. Choose **Data > View Memory**.

A new Memory window appears.

3. Select **Raw data** from the **View** list pop-up.

The contents of memory referenced by the pointer appears in both hexadecimal and ASCII.

---

## **Viewing Different Memory Spaces**

Use the **Page** list pop-up to view a particular memory space.

---

**NOTE**

This feature is available only for processors that support multiple memory spaces.

---

1. Select the name of a variable or a function in a source window.
2. Choose **Data > View Memory**.  
A Memory window appears.
3. Select a memory space from the **Page** list pop-up.
4. Select **Raw data** from the **View** list pop-up if inspecting a variable. Select **Disassembly**, **Source**, or **Mixed** from the **View** list pop-up if inspecting source code.

The Memory window displays the selected memory-space page.

---

## **Setting a Watchpoint in the Memory Window**

To set a Watchpoint using the **Memory** window, follow these steps:

1. **Run/Debug** your program.
2. From the Main Toolbar, choose **Data > View Memory**.  
This opens the **Memory** window.
3. Select a range of bytes in the **Memory** window.  
Do not double-click the range of bytes.

4. From the Main Toolbar, choose **Debug > Set Watchpoint**.

---

**NOTE** A red line appears under the selected variable in the Variable window, indicating that you have set a Watchpoint. You can change the color of this line in the **Display Settings** panel of the IDE Preferences window (**Edit > IDE Preferences**).

---

---

## Clearing Watchpoints from the Memory window

To clear a Watchpoint from the Memory window, follow these steps:

1. Select a range of bytes in the Memory window.
2. Choose **Debug > Clear Watchpoint**.

To clear *all* Watchpoints from the Memory window:

1. Open the Memory window.  
You do not have to select a range of bytes.
2. Choose **Debug > Clear All Watchpoints**.

---

**NOTE** All Watchpoints clear automatically when the target program terminates or the debugger terminates the program, and reset next time the program runs.

---

## Multi-core Debugging

The IDE allows simultaneous debugging of multiple projects. This feature provides multi-core debugging capability for some embedded processors. By configuring each project to operate on a single core, the IDE can debug multiple cores by debugging multiple projects.

Configuring multi-core debugging involves these tasks:

- Configuring specific target settings for each project

- For some cores, specifying a configuration file for initializing multi-core debugging

For more information, see the *Targeting* documentation.



# Using Breakpoints

---

This chapter explains how to use breakpoints in the CodeWarrior™ IDE. Breakpoints alter normal program execution, optionally according to specified conditions or events.

This chapter contains these sections:

- [“About Breakpoints” on page 231](#)
- [“Breakpoints Window” on page 231](#)
- [“Common Breakpoint Tasks” on page 233](#)

## About Breakpoints

Breakpoints change the flow of normal program execution. Use them to suspend execution, examine current program state, and trigger specific actions.

The IDE recognizes these major breakpoint types:

- Regular breakpoints—halt program execution on specific source lines
- Conditional breakpoints—halt program execution after meeting specified conditions
- Temporary breakpoints—halt program execution and then remove the breakpoint that caused the halt

## Breakpoints Window

The Breakpoints window shows all breakpoints defined in the active project. For each breakpoint, the window shows this information:

- The source file that contains the breakpoint
- The line that contains the breakpoint
- The current status of the breakpoint
- The breakpoint class

## Using Breakpoints

### Breakpoints Window

---

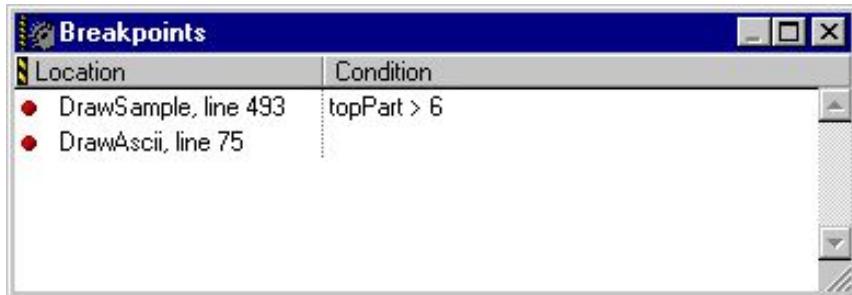
- (Optional) The condition associated with the breakpoint

Use the Breakpoints window to perform these tasks:

- Manipulate breakpoint conditions
- Change breakpoint status
- View the source line that contains each breakpoint
- View the class of each breakpoint

[Figure 20.1](#) shows the Breakpoints window. [Table 20.1](#) explains the items in the window.

**Figure 20.1 Breakpoints window**



**Table 20.1 Breakpoints window—items**

Item	Explanation
Status	Click the displayed marker to toggle breakpoint status: <ul style="list-style-type: none"><li>•  indicates an active breakpoint</li><li>•  indicates an inactive breakpoint</li></ul>
Location	A list of all breakpoints currently set in the project files.
Class	Indicates the breakpoint class: <ul style="list-style-type: none"><li>• regular</li><li>• conditional</li><li>• temporary</li></ul>
Condition	Displays the condition attached to the breakpoint. To attach a condition to a breakpoint, double-click the Condition column, then type a conditional expression.

# Common Breakpoint Tasks

You can perform these breakpoint tasks:

- opening the Breakpoints window
- setting breakpoints in source panes
- clearing breakpoints in source panes
- clearing all breakpoints
- clearing breakpoints in the Breakpoints window
- setting temporary breakpoints
- setting conditional breakpoints

---

**NOTE** The Solaris- and Linux-hosted IDEs rely on an external debugger to provide breakpoint support. The tasks in this section do not apply to these IDE hosts.

---

## Opening the Breakpoints Window

Use the **Breakpoints** window to view all breakpoints currently set in the program.

1. Open the **Breakpoints** window, as explained in [Table 20.2](#).

**Table 20.2 Opening the Breakpoints window**

On this host...	Do this...
Windows	Select <b>View &gt; Breakpoints</b> .
Macintosh	Select <b>Window &gt; Breakpoints Window</b> .
Solaris	Select <b>Window &gt; Breakpoints Window</b> .
Linux	Select <b>Window &gt; Breakpoints Window</b> .

2. The Breakpoints window appears.

---

**NOTE** Double-click a breakpoint in the Breakpoints window to display its associated source line in the Browser window.

---

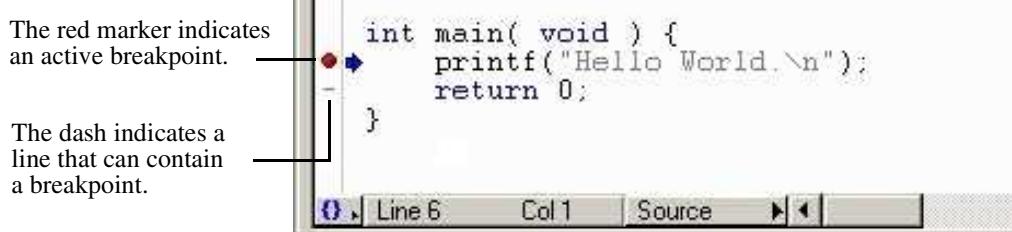
## Setting Breakpoints in Source Panes

Use the **Set Breakpoint** command to set a breakpoint. A breakpoint suspends execution of the target program at a specified line and returns control to the debugger. The debugger does not execute the line that contains the breakpoint.

Set breakpoints in any window that displays source code, such as these examples:

- editor windows
- the **Thread** window (with **Stack Crawl**, **Variable**, and **Source** panes)

**Figure 20.2 Setting Breakpoints**



1. In the source code, find the line at which you want to set the breakpoint.
2. Click the dash in the **Breakpoint** column (left edge of window) next to the line of source code.

The breakpoint appears. During the next program run, execution proceeds normally until reaching the breakpoint. The debugger then suspends program execution.

---

<b>NOTE</b>	Specifying a breakpoint in a file affects all other targets that include that file.
-------------	---

---

## Clearing Breakpoints in Source Panes

Use the **Clear Breakpoints** command to clear an individual breakpoint.

Clear breakpoints in any window that displays source code, such as these examples:

- editor windows

- the **Thread** window (with **Stack Crawl**, **Variable**, and **Source** panes)
1. Open any window that displays source code.
  2. Find the breakpoint that you want to clear. A red marker in the **Breakpoint** column (left edge of window) indicates a breakpoint.
  3. Click the red marker in the **Breakpoint** column.

The IDE clears the breakpoint.

---

## Clearing All Breakpoints

Use the **Clear All Breakpoints** command to clear all project breakpoints with a single command.

1. Select **Debug > Clear All Breakpoints**.
2. The IDE clears all breakpoints in project source files.

---

**NOTE**      Alternatively, use the Breakpoints window to clear breakpoints: select the breakpoint that you want to remove and press Delete.

---

---

## Clearing Breakpoints in the Breakpoints window

Use the **Clear Breakpoints** command to clear breakpoints in the **Breakpoints** window.

1. Open the **Breakpoints** window:
2. In the **Location** column, select the breakpoint that you want to clear.
3. Click the red marker to the left of the breakpoint.

The IDE clears the breakpoint.

## Setting Temporary Breakpoints

Use the **Temporary Breakpoint** command to set temporary breakpoints. A temporary breakpoint causes the debugger to halt program execution only once. The debugger removes the temporary breakpoint after suspending program execution. Essentially, setting a temporary breakpoint is equivalent to using the **Run To Cursor** command.

1. Open any window that displays source code.
2. In the source code, find the line at which you want to set the temporary breakpoint.
3. Set the temporary breakpoint, as explained in [Table 20.3](#).

**Table 20.3 Setting the temporary breakpoint**

On this host...	Do this...
Windows	Alt-click the dash in the Breakpoint column.
Macintosh	Option-click the dash in the Breakpoint column.
Solaris	Alt-click the dash in the Breakpoint column.
Linux	Alt-click the dash in the Breakpoint column.

The temporary breakpoint appears.

---

**NOTE** For source code with multiple breakpoints, the debugger suspends program execution at each breakpoint. Suppose that some source code contains a breakpoint followed by temporary breakpoint. The debugger suspends program execution at the breakpoint first and then again at the temporary breakpoint.

---

## Setting Conditional Breakpoints

Use the **Conditional Breakpoint** command to set a conditional breakpoint. A conditional breakpoint is a breakpoint with an associated conditional expression. The

debugger evaluates the expression to determine whether to suspend program execution at that breakpoint.

A conditional breakpoint behaves in two different ways:

- If the expression evaluates to true (non-zero value), the debugger suspends program execution.
- If the expression evaluates to false (zero), the breakpoint has no effect. Program execution continues without interruption.

1. Set a breakpoint that you want to associate with a conditional expression.
2. Open the **Breakpoints** window.
3. Double-click the condition column of the breakpoint.
4. Enter a conditional expression in the **Condition** text box.
5. Choose **Project > Debug**.

During subsequent debugging runs, the debugger evaluates the conditional breakpoint to determine whether to stop the program at that breakpoint.

---

**NOTE**      Alternatively, drag-and-drop an expression from a source view or from the Expression window into the Breakpoints window.

---

## **Using Breakpoints**

*Common Breakpoint Tasks*

---

# Using Watchpoints

---

This chapter explains how to use Watchpoints in the CodeWarrior® IDE, and contains these sections:

- [“About Watchpoints” on page 239](#)
- [“Using Watchpoints in the Memory Window” on page 240](#)
- [“Using Watchpoints in the Watchpoints Window” on page 242](#)
- [“Using Watchpoints in Other Windows” on page 245](#)

## About Watchpoints

A Watchpoint tracks locations in memory that you want the debugger to observe. Set a Watchpoint when you want the IDE to alert you to which part of the program changed a particular location in memory.

Unlike a Breakpoint, a Watchpoint can detect when *any* part of the program affects the global data. When the program writes a new value to that location or area of memory, the debugger suspends program execution and notifies you with an alert.

Typically, the variables or memory locations on which you set a Watchpoint are underlined in red in the **Memory**, **Variable**, or **Symbolics** windows. (However, you can change the alert’s color to your preference.)

When you get an alert, you can examine the call chain, inspect or change variables, step through your code, or use any of the debugger’s other facilities. (In particular, from the debugger level, you *can* change the contents of the location that triggered the Watchpoint without triggering it again.) Use the **Run** command (or the **Run** button on the debugger toolbar) to continue execution from the Watchpoint.

---

<b>NOTE</b>	Since the debugger cannot detect Watchpoints for variables stored on the stack or in registers, you cannot set Watchpoints on local variables.
-------------	--

---

## Using Watchpoints

*Using Watchpoints in the Memory Window*

---

While you can view the program's current Watchpoints in the **Watchpoints** window, you can set and clear a Watchpoint in several IDE windows. To view, set, or clear Watchpoints, you must first debug your program.

A project can have a different maximum number of watchpoints, depending on the target. The IDE generally limits the acceptable range for watchpoints to memory that it can write-protect. This range also depends on the host and on the application. For more information, see the *Targeting* documentation.

---

## Viewing Watchpoints

To see a list of all Watchpoints you have set in your program, open the **Watchpoints** window.

To view Watchpoints, follow these steps:

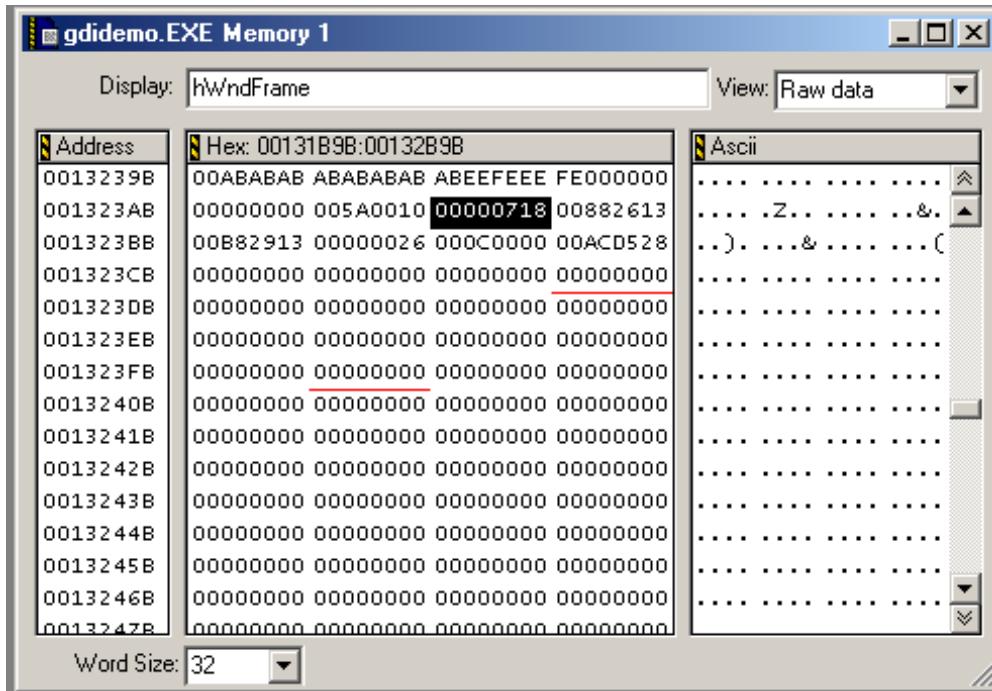
1. **Run/Debug** your program.
2. Open the Watchpoints window, as explained in [Table 21.1](#).

**Table 21.1 Opening the Watchpoints window**

On this platform...	Do this...
Windows	Select <b>View &gt; Watchpoints</b> .
Macintosh	Select <b>Window &gt; Watchpoints</b> .
Solaris	Select <b>Window &gt; Watchpoints</b> .
Linux	Select <b>Window &gt; Watchpoints</b> .

## Using Watchpoints in the Memory Window

The **Memory** window displays the contents of memory in hexadecimal and corresponding ASCII character values.

**Figure 21.1 Memory Window**

---

## Setting a Watchpoint in the Memory Window

To set a Watchpoint using the **Memory** window, follow these steps:

1. **Run/Debug** your program.
2. From the Main Toolbar, choose **Data > View Memory**.  
This opens the **Memory** window.
3. Select a range of bytes in the **Memory** window.  
Do not double-click the range of bytes.
4. From the Main Toolbar, choose **Debug > Set Watchpoint**.

## Using Watchpoints

*Using Watchpoints in the Watchpoints Window*

---

<b>NOTE</b>	A red line appears under the selected variable in the Variable window, indicating that you have set a Watchpoint. You can change the color of this line in the <b>Display Settings</b> panel of the IDE Preferences window ( <b>Edit &gt; IDE Preferences</b> ).
-------------	--

---

## Clearing Watchpoints from the Memory window

To clear a Watchpoint from the Memory window, follow these steps:

1. Select a range of bytes in the Memory window.
2. Choose **Debug > Clear Watchpoint**.

To clear *all* Watchpoints from the Memory window:

1. Open the Memory window.  
You do not have to select a range of bytes.
2. Choose **Debug > Clear All Watchpoints**.

<b>NOTE</b>	All Watchpoints clear automatically when the target program terminates or the debugger terminates the program, and reset next time the program runs.
-------------	--

---

# Using Watchpoints in the Watchpoints Window

The **Watchpoints** window lists the Watchpoints in your current build target by memory address.

**Figure 21.2 Watchpoints window**

Location	Length	Description	Condition
0x001323FF	4	gdidemo.EXE Memory 1	
0x001323FB	4	gdidemo.EXE Memory 1	
0x00132407	4	gdidemo.EXE Memory 1	
0x0013239B	4	gdidemo.EXE Memory 1	
0x00132433	4	gdidemo.EXE Memory 1	
0x0013245F	4	gdidemo.EXE Memory 1	
0x0013246B	4	gdidemo.EXE Memory 1	

---

### Clearing a Watchpoint from the Watchpoints window

To clear a Watchpoint from the **Watchpoints** window, follow these steps:

1. Open the Watchpoints window, as explained in [Table 21.2](#).

**Table 21.2 Opening the Watchpoints window**

On this platform...	Do this...
Windows	Select <b>View &gt; Watchpoints</b> .
Macintosh	Select <b>Window &gt; Watchpoints</b> .
Solaris	Select <b>Window &gt; Watchpoints</b> .
Linux	Select <b>Window &gt; Watchpoints</b> .

2. Select an existing Watchpoint in the Watchpoints window.
3. Choose **Debug > Clear Watchpoint**.

Alternately, you can right-click the Watchpoint and select **Clear Watchpoint**. You can also highlight a Watchpoint and press the Delete key to clear a Watchpoint in the Watchpoints window.

The variable disappears from the Watchpoints window.

## Using Watchpoints

*Using Watchpoints in the Watchpoints Window*

---

### Disabling a Watchpoint from the Watchpoints window

To disable a Watchpoint from the **Watchpoints** window, follow these steps:

1. Open the Watchpoints window, as explained in [Table 21.3](#).

**Table 21.3 Opening the Watchpoints window**

On this platform...	Do this...
Windows	Select <b>View &gt; Watchpoints</b> .
Macintosh	Select <b>Window &gt; Watchpoints</b> .
Solaris	Select <b>Window &gt; Watchpoints</b> .
Linux	Select <b>Window &gt; Watchpoints</b> .

2. Select an existing Watchpoint in the Watchpoints window.

3. Choose **Debug > Disable Watchpoint**.

The Watchpoint grays out. The Watchpoint remains intact, however, the program does not stop at this Watchpoint or alert you when data changes.

Alternately, you can right-click the Watchpoint and select **Disable Watchpoint**.

---

### Enabling a Watchpoint from the Watchpoints window

To enable a Watchpoint from the **Watchpoints** window, follow these steps:

1. Open the Watchpoints window, as explained in [Table 21.4](#).

**Table 21.4 Opening the Watchpoints window**

On this platform...	Do this...
Windows	Select <b>View &gt; Watchpoints</b> .
Macintosh	Select <b>Window &gt; Watchpoints</b> .
Solaris	Select <b>Window &gt; Watchpoints</b> .
Linux	Select <b>Window &gt; Watchpoints</b> .

2. Select an existing, *disabled* Watchpoint in the Watchpoints window.

While intact, the disabled Watchpoint is grayed-out and the program will not stop or alert you when data changes.

3. Choose **Debug > Enable Watchpoint**.

The program now stops at this Watchpoint and alerts you when data changes.

Alternately, you can right-click the Watchpoint and select **Enable Watchpoint**.

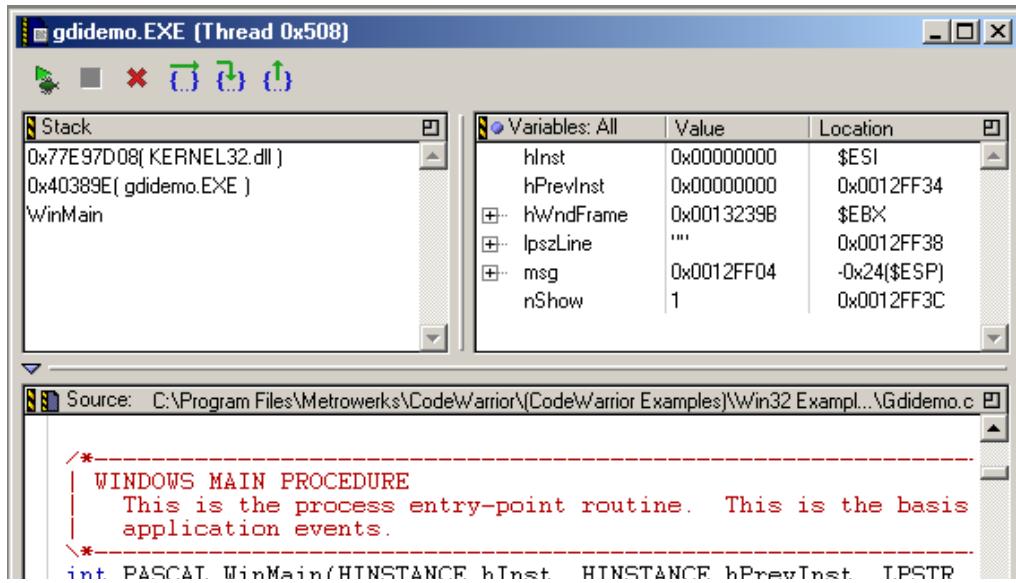
## Using Watchpoints in Other Windows

As a general rule, you can right-click a variable in most windows to set and clear a Watchpoint.

### Using Watchpoints in the Thread Window

The **Thread** window ([Figure 21.3](#)), also known as the **Debugger**, **Stack Crawl**, and **Program** windows, shows debugging information about the currently suspended process.

**Figure 21.3 Thread window**



## Using Watchpoints

*Using Watchpoints in Other Windows*

---

---

## Setting a Watchpoint in the Thread Window

Follow these steps to set a Watchpoint in the Variables pane in the Thread window:

1. **Debug/Run** your program.  
The Thread window opens.
2. In the **Variables** pane in the Thread window, select the variable.
3. Choose **Debug > Set Watchpoint**.

Alternately, right-click the variable and **Set Watchpoint**.

Follow these steps to set a Watchpoint in the source pane of the Thread window:

1. **Debug/Run** your program.  
The Thread window opens.
2. In the **Source** pane in the Thread window, double-click the variable.
3. Choose **Debug > Set Watchpoint**.  
Alternately, right-click the variable and **Set Watchpoint**.

---

## Clearing a Watchpoint in the Thread Window

To clear a watchpoint in the Thread window, follow these steps:

1. **Debug/Run** your program.  
The Thread window opens.
2. In the Source pane in the Thread window, double-click the variable.
3. Choose **Debug > Clear Watchpoint**.  
Alternately, right-click the variable and **Clear Watchpoint**.

## Enabling and Disabling Watchpoints in the Thread window

To enable a Watchpoint in the **Thread** window, follow these steps:

1. **Debug/Run** your program.

The Thread window opens.

2. In the Source pane in the Thread window, double-click the variable.

3. Choose **Debug > Enable/Disable Watchpoint**.

Alternately, right-click the variable and **Enable/Disable Watchpoint**.

While intact, a disabled Watchpoint is grayed-out and the program does not stop or alert you when data changes.

An enabled Watchpoint suspends program execution and alerts you when data changes.

## Using Watchpoints in the Variable Window

The **Variable** window displays a single variable and lets you edit its contents.

---

## Setting a Watchpoint in the Variable Window

To set a Watchpoint in the Variable window, follow these steps:

1. In the **Thread** window, double-click on a variable in the Variables pane to open a **Variable** window.

2. Select the variable from the **Variable** window.

3. Choose **Debug > Set Watchpoint**.

Alternately, right-click the variable and **Set Watchpoint**.

## Using Watchpoints

*Using Watchpoints in Other Windows*

---

## Setting a Conditional Watchpoint

A conditional watchpoint is a watchpoint that stops program execution at a given point when a specified *condition* is met. Use the **Variable** window to create conditional watchpoints.

To set a conditional watchpoint, follow these steps:

1. Select a variable from the Variable window.
2. Select **Debug > Set Watchpoint**.
3. Open the Watchpoints window, as explained in [Table 21.5 on page 248](#).

**Table 21.5 Opening the Watchpoints window**

On this platform...	Do this...
Windows	Select <b>View &gt; Watchpoints</b> .
Macintosh	Select <b>Window &gt; Watchpoints</b> .
Solaris	Select <b>Window &gt; Watchpoints</b> .
Linux	Select <b>Window &gt; Watchpoints</b> .

4. Double-click the **Condition** field in the Watchpoints window for the watchpoint that you just created.

The field becomes editable.

5. Enter an expression in the Condition field.

The debugger evaluates this expression to determine whether to stop program execution at the watchpoint. If the expression evaluates to true (non-zero), the debugger stops program execution at the watchpoint. If the expression evaluates to false (zero), program execution continues past the watchpoint.

6. Choose **Project > Debug**.

---

**NOTE**

When the debugger reaches a Watchpoint, it stops execution before it executes the statement.

---

## Clearing Watchpoints from the Variable window

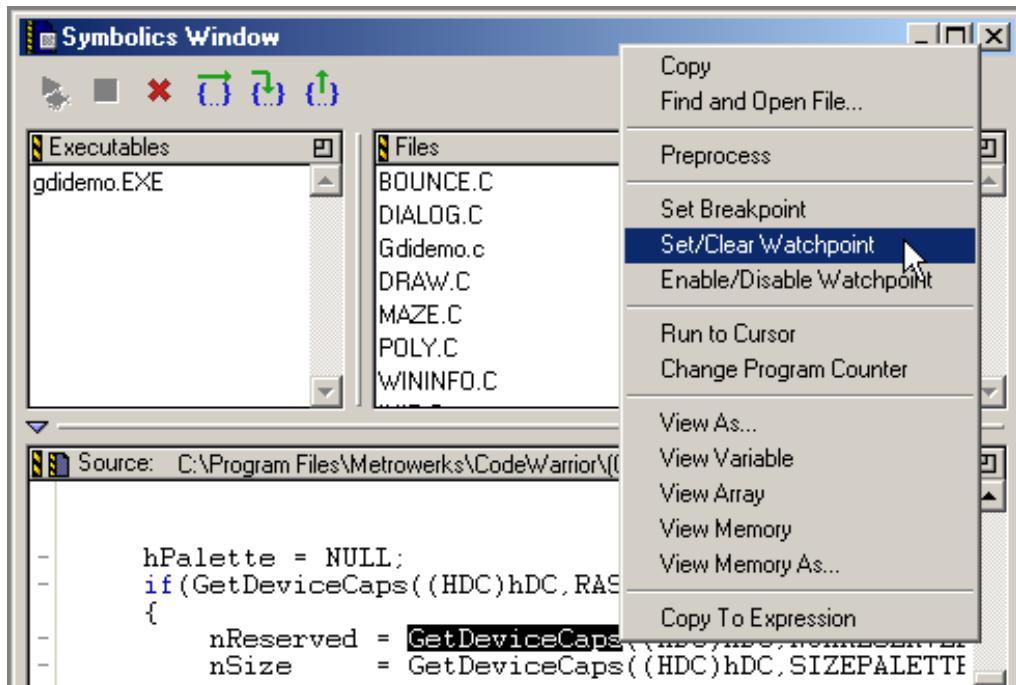
To clear a watchpoint from the Variable window, follow these steps:

1. Select a Watchpoint in the Variable window.
2. Choose **Debug > Clear Watchpoint**.

## Using Watchpoints in the Symbolics Window

The **Symbolics** window lets you view any file in the build target for which the IDE generated a symbolics file, and lets you view or edit all global variables for each project that contains symbolics information. The Symbolics window appears once you open a file that contains symbolics information.

Figure 21.4 Symbolics window



## Using Watchpoints

*Using Watchpoints in Other Windows*

---

### Setting or Clearing a Watchpoint in the Symbolics Window

To set a Watchpoint in the **Symbolics** window, follow these steps:

1. Select **Project > Debug**.  
A debugging session starts
2. Open the Symbolics window.
3. Highlight a variable in the **Source** pane of the Symbolics window.
4. Select **Debug > Set/Clear Watchpoint**.

The IDE sets the watchpoint. Select the variable again and choose the same menu command to clear the watchpoint.

Alternatively, use a contextual menu to set or clear the watchpoint, as explained in [Table 21.6](#).

**Table 21.6** Clearing a watchpoint by using a contextual menu

On this host...	Do this...
Windows	Right-click the variable and select <b>Set/Clear Watchpoint</b> .
Macintosh	Control-click the variable and select <b>Set/Clear Watchpoint</b> .
Solaris	Control-click the variable and select <b>Set/Clear Watchpoint</b> .
Linux	Ctrl-click the variable and select <b>Set/Clear Watchpoint</b> .

# Using Eventpoints

---

This chapter explains how to use eventpoints in the CodeWarrior™ IDE, and contains these sections:

- [“About Eventpoints” on page 251](#)
- [“Using Eventpoints in the Breakpoints Window” on page 256](#)
- [“Using Eventpoints in Other Windows” on page 259](#)
- [“Using Eventpoints in the Symbolics Window” on page 261](#)

## About Eventpoints

An eventpoint is a conditional breakpoint that performs a specific task instead of halting program execution. An eventpoint can behave in various ways.

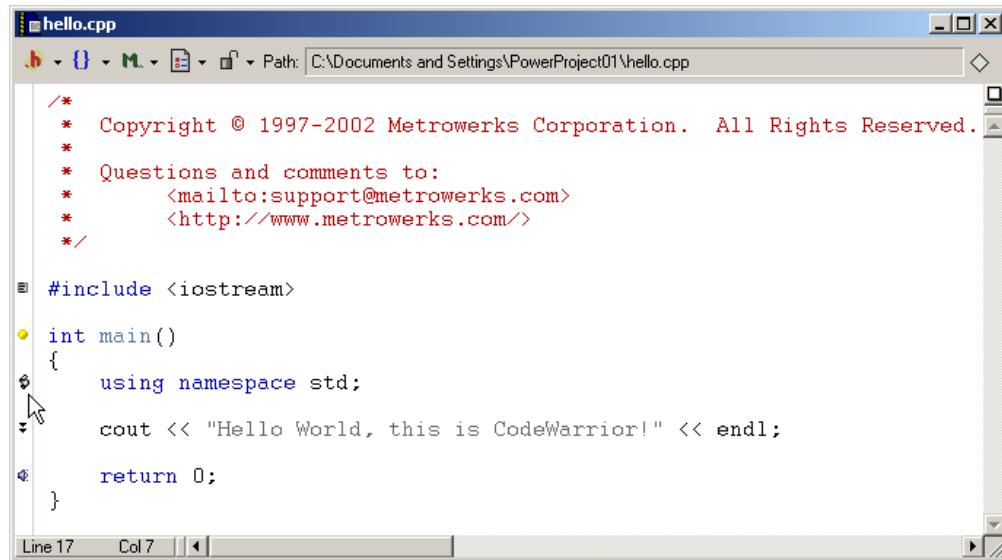
You can set these types of eventpoints:

- **Log Point**—logs or *speaks* a string or expression; records messages to the log window
- **Pause Point**—pauses execution just long enough to refresh debugger data
- **Script Point**—runs a script, application, or other item when it reaches the eventpoint in a line of code. The eventpoint triggers when execution reaches that line in the code.
- **Skip Point**—skips execution on a source line where you set an eventpoint
- **Sound Point**—plays a sound when the program finds an eventpoint

## Using Eventpoints

*About Eventpoints*

**Figure 22.1 Eventpoints in an editor window**



**Table 22.1 Eventpoint icons**

Eventpoint	Icon	Explanation
Log Point	█	The IDE logs or speaks a string or expression.
Pause Point	●	The IDE pauses execution just long enough to refresh debugger data.
Script Point	⌚	The IDE runs a script, application, or other item.
Skip Point	▼	The IDE skips execution of a source line on which you set the eventpoint.
Sound Point	🔊	The IDE plays a sound when it finds an eventpoint.
Inactive eventpoint	○	This icon shows that an eventpoint is inactive (disabled).

## Log Point

A Log Point is an eventpoint that logs or speaks a string or expression. A Log Point records messages to the log window. You can configure the message that appears in the log window.

**Figure 22.2 Log Point settings**



Select at least one of these Log Point settings:

- **Log Message**—the default setting for Log Point, the IDE writes the message you type in the **Message** field in an IDE log window when it finds an eventpoint.
- **Speak Message**—select to have the debugger use the operating system to *speak* the message you type in the **Message** field.

---

**NOTE** (Windows) Install the Speech software development kit (SDK) in order to use the Speak Message option.

---

- **Treat as Expression**—evaluates the text you typed in the Message field. For example, if you enter "n = 100" in the Message field, the debugger parses that text as an expression.

---

**TIP** Include quotation marks around your expression, for example, "n = 100".

---

- **Stop in Debugger**—select to have the debugger stop when it finds an eventpoint.

## Pause Point

A Pause Point suspends program execution just long enough to refresh debugger data. In a typical debugging session, the debugger cannot refresh the data unless it stops program execution. A Pause Point, however, lets you set an eventpoint to see if memory changes, for example, without stopping the debugging session.

## Script Point

You can configure the program to run a script, application, or other item when it reaches the eventpoint in a line of code. The eventpoint triggers when execution reaches that line in the code. For example, the Mac OS can launch AppleScripts or applications; Windows can execute a file as if you had used a Windows command line.

## Skip Point

You can set a Skip Point so that the IDE does not execute a source line on which you set the eventpoint. This eventpoint is especially useful when you are aware of a line that you need to fix, but would like to go ahead and debug the rest of the program. You can set a Skip Point and the debugger will debug the rest of the application but skip that particular line.

---

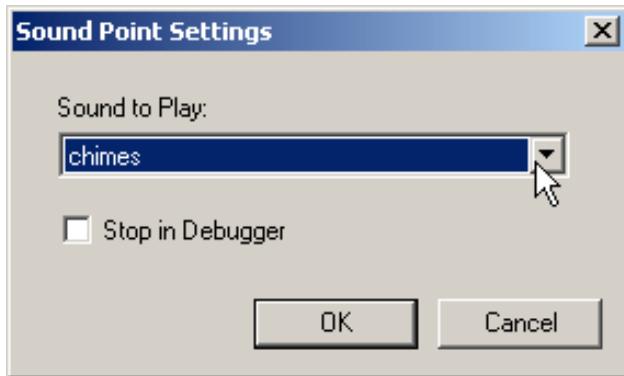
**NOTE** Skip Points do not currently work in Java.

---

## Sound Point

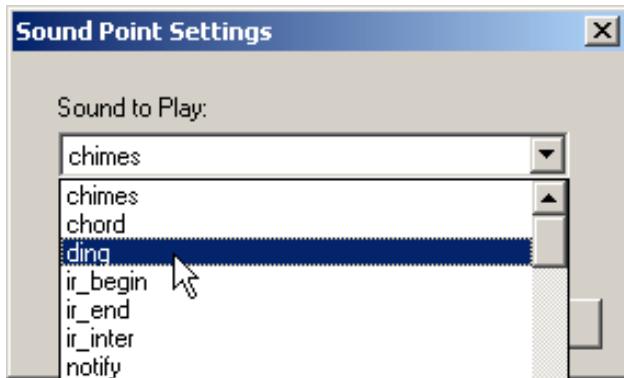
A Sound Point is an audible alert. You can set a Sound Point so that when you step or run through code, the IDE plays a sound when it finds an eventpoint. Unlike a Log Point set to **Speak Message**, which speaks the message you specify, the Sound Point plays a simple notification sound.

**Figure 22.3 Sound point settings**



You can configure the sound you would like to hear when the program finds an eventpoint. To configure speech, use a Log Point.

**Figure 22.4 Sound point settings drop-down menu**



---

## Setting Eventpoints

You can use the editor and debugger windows to set eventpoints.

1. Open either an editor or debugger window that contains source code.
2. In the source code, find the line on which you want to set the eventpoint.

## Using Eventpoints

*Using Eventpoints in the Breakpoints Window*

---

3. Double-click the line or symbol on which you want to set the eventpoint.
4. Choose **Debug > Set Eventpoint**. Alternately, right-click the symbol and select **Set Eventpoint**.

Choose one of the eventpoints—Log, Pause, Script, Skip, or Sound—in the drop-down menu. You can choose more than one eventpoint for the same line or symbol.

5. Debug the program.

During subsequent debugging runs, the eventpoints trigger their associated tasks as defined by their IDE plug-ins.

---

## Clearing Eventpoints

You can clear eventpoints from most IDE windows.

1. Open any window that displays source code and has an eventpoint.
2. Click on the eventpoint *icon* in the window's breakpoints column.  
This clears the eventpoint.

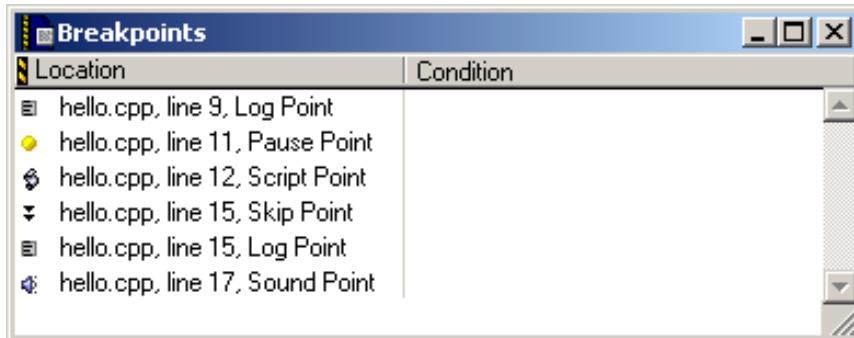
Alternately, to this:

1. Double-click the line or symbol on which you want to clear the eventpoint.
2. Choose **Debug > Clear Eventpoint**.

# Using Eventpoints in the Breakpoints Window

The **Breakpoints** window lists the eventpoints in your current build target by memory address.

**Figure 22.5 Breakpoints window**



---

### Clearing an Eventpoint from the Breakpoints window

To clear an eventpoint from the Breakpoints window, follow these steps:

1. Open the Breakpoints window, as explained in [Table 22.2](#).

**Table 22.2 Opening the Breakpoints window**

On this platform...	Do this...
Windows	Select <b>View &gt; Breakpoints</b> .
Macintosh	Select <b>Window &gt; Breakpoints</b> .
Solaris	Select <b>Window &gt; Breakpoints</b> .
Linux	Select <b>Window &gt; Breakpoints</b> .

2. Click on the text that has the eventpoint you want to clear.
3. Choose **Debug > Clear Eventpoint**.

---

### Disabling an Eventpoint from the Breakpoints window

To disable an eventpoint from the Breakpoints window, follow these steps:

1. Open the Breakpoints window, as explained in [Table 22.2 on page 257](#).

## Using Eventpoints

*Using Eventpoints in the Breakpoints Window*

---

**Table 22.3 Opening the Breakpoints window**

On this platform...	Do this...
Windows	Select <b>View &gt; Breakpoints</b> .
Macintosh	Select <b>Window &gt; Breakpoints</b> .
Solaris	Select <b>Window &gt; Breakpoints</b> .
Linux	Select <b>Window &gt; Breakpoints</b> .

2. Click the eventpoint icon.

This action toggles the eventpoint, disabling it. Click the icon again to enable the eventpoint.

---

## Enabling an Eventpoint from the Breakpoints window

To enable an eventpoint from the Breakpoints window, follow these steps:

1. Open the Breakpoints window, as explained in [Table 22.4](#).

**Table 22.4 Opening the Breakpoints window**

On this platform...	Do this...
Windows	Select <b>View &gt; Breakpoints</b> .
Macintosh	Select <b>Window &gt; Breakpoints</b> .
Solaris	Select <b>Window &gt; Breakpoints</b> .
Linux	Select <b>Window &gt; Breakpoints</b> .

2. Click the icon of a *disabled* eventpoint.

This toggles the eventpoint, enabling it. You can click the icon again to disable the eventpoint.

---

## Setting a Conditional Eventpoint

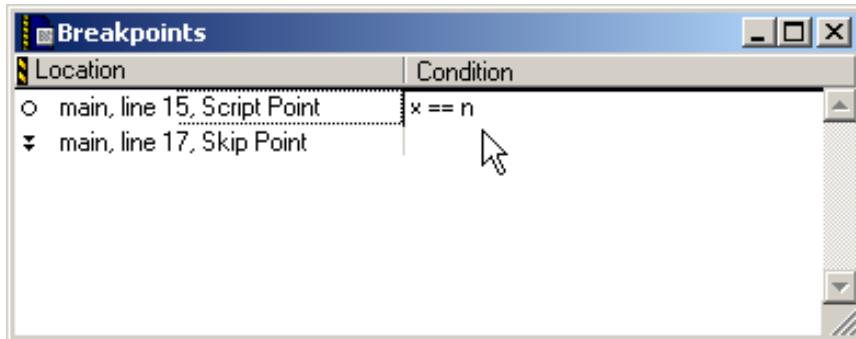
A conditional eventpoint is an eventpoint that initiates an event at a given point in the source code when a specified *condition* is met. Similar to a conditional breakpoint,

which stops program execution if a condition is met, a conditional eventpoint performs an action. Use the Breakpoints window to create conditional eventpoints.

To set a conditional eventpoint, follow these steps:

1. Open the Breakpoints window.
2. In the **Condition** field next to the eventpoint, enter your condition. For example, enter a condition like this:  $x == n$ .

**Figure 22.6 Conditional eventpoint**



The next time you debug the program, the IDE initiates an event based on the conditions you set for the eventpoint.

## Using Eventpoints in Other Windows

As a general rule, you can right-click a variable in most windows to set and clear an eventpoint.

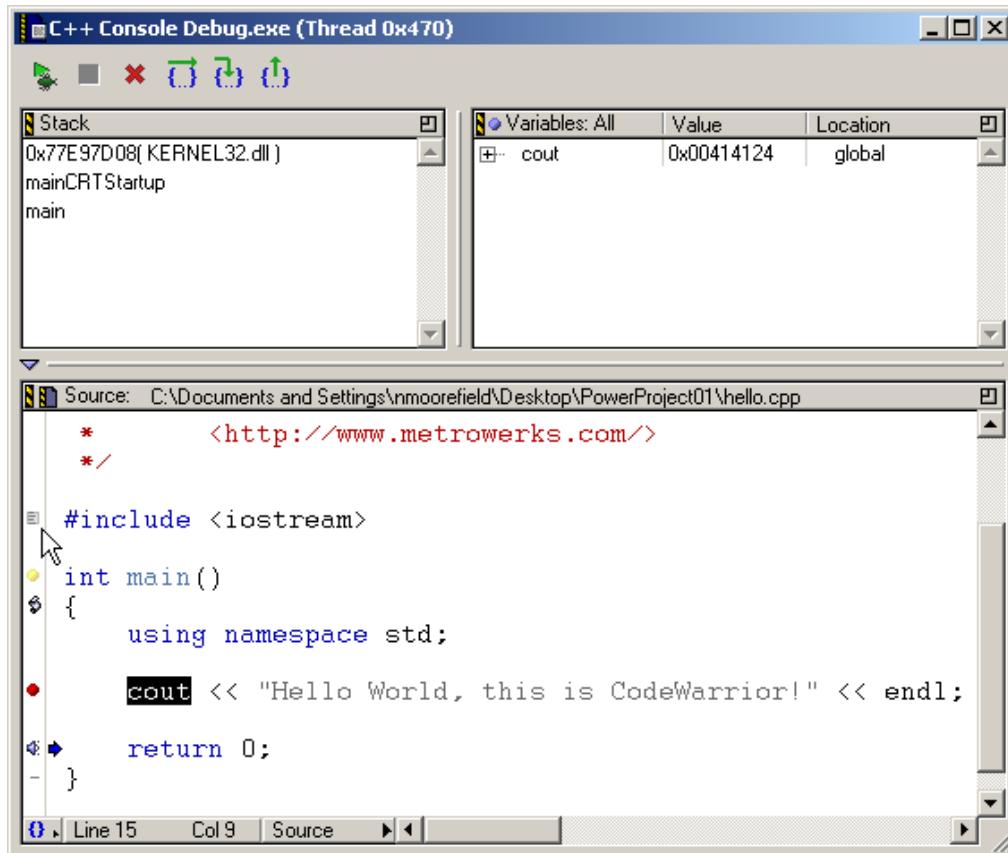
### Using Eventpoints in the Thread Window

The **Thread** window, also known as the **Debugger**, **Stack Crawl**, and **Program** windows, shows debugging information about the currently suspended process.

## Using Eventpoints

*Using Eventpoints in Other Windows*

**Figure 22.7 Thread window**



## Setting an Eventpoint in the Thread Window

Follow these steps to set an eventpoint in the Variables pane in the Thread window:

1. **Debug** your program.

The Thread window appears.

2. In the **Variables** pane in the Thread window, select the variable.

3. Choose **Debug > Set Eventpoint**.

Follow these steps to set an eventpoint in the **Source** pane in the Thread window:

1. **Debug** your program.

The Thread window appears.

2. In the **Source** pane in the Thread window, double-click the variable.

3. Choose **Debug > Set Eventpoint**.

Alternately, right-click the variable and choose **Set Eventpoint**.

---

## **Clearing an Eventpoint in the Thread Window**

To clear an eventpoint in **Variables** pane in the Thread window, follow these steps:

1. **Debug** your program.

The Thread window appears.

2. In the Variables pane in the Thread window, double-click the variable.

3. Choose **Debug > Clear Eventpoint**.

To clear an Eventpoint in the **Source** pane in the Thread window, follow these steps:

1. In the Source pane in the Thread window, double-click the variable.

2. Choose **Debug > Clear Eventpoint**.

Alternately, right-click the variable and choose **Clear Eventpoint**.

# **Using Eventpoints in the Symbolics Window**

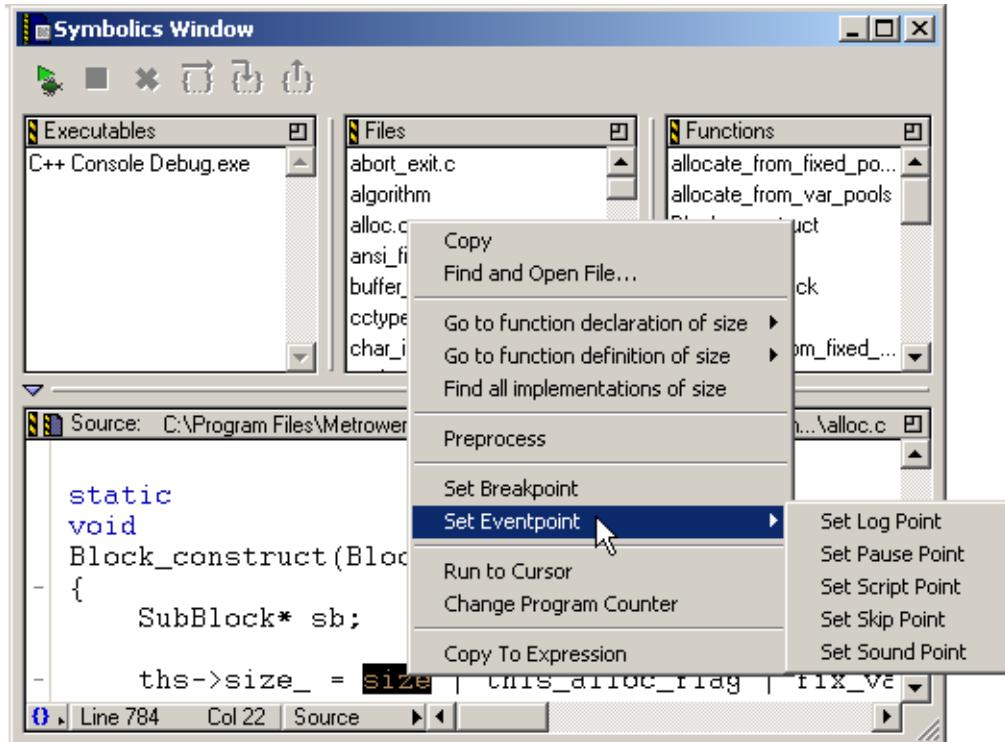
The **Symbolics** window lets you view any file in the build target for which the IDE generated a symbolics file, and lets you view or edit all global variables for each project that contains symbolics information. The Symbolics window appears once you open a file that contains symbolics information.

---

## Using Eventpoints

Using Eventpoints in the Symbolics Window

Figure 22.8 Symbolics window



## Setting an Eventpoint in the Symbolics Window

To set an eventpoint in the **Symbolics** window, follow these steps:

1. Choose **Project > Debug**.  
The Thread window appears.
2. Open the **Symbolics** window, as explained in [Table 22.5](#).

**Table 22.5 Opening the Symbolics window**

On this platform...	Do this...
Windows	Select <b>View &gt; Symbolics</b> .
Macintosh	Select <b>Window &gt; Symbolics Window</b> .
Solaris	Select <b>Window &gt; Symbolics Window</b> .
Linux	Select <b>Window &gt; Symbolics Window</b> .

3. Select the variable in the **Source** pane of the **Symbolics** window.
4. Choose **Debug > Set Eventpoint**.

Alternately, right-click the variable and choose **Set Eventpoint**.

---

### Clearing an Eventpoint in the Symbolics Window

To clear an eventpoint in the **Symbolics** window, follow these steps:

1. Choose **Project > Debug**.  
The Thread window appears.
2. Open the **Symbolics** window, as explained in [Table 22.6](#).

**Table 22.6 Opening the Symbolics window**

On this platform...	Do this...
Windows	Select <b>View &gt; Symbolics</b> .
Macintosh	Select <b>Window &gt; Symbolics Window</b> .
Solaris	Select <b>Window &gt; Symbolics Window</b> .
Linux	Select <b>Window &gt; Symbolics Window</b> .

3. Select the variable in the **Source** pane of the **Symbolics** window.
4. Choose **Debug > Clear Eventpoint**.

Alternately, right-click the variable and choose **Clear Eventpoint**.

---

## **Using Eventpoints**

*Using Eventpoints in the Symbolics Window*

---

# Compilers and Linkers

---

This section contains these chapters:

- [Compilers](#)
- [Linkers](#)



# Compilers

---

This chapter explains how to work with compilers in the CodeWarrior™ IDE. The IDE uses compilers to complete these tasks:

- Generate object code—the compiler translates source code into object code. Sample source code includes C++ files and Java files. Object code represents the same source instructions in a language that the computer directly understands.
- Flag syntax errors—the compiler highlights source code with syntax errors. Syntax errors result from failing to follow valid structure in a programming language. In C++, a common syntax error is to forget to conclude a statement with a semicolon.

Read this chapter to learn more about typical tasks for working with compilers.

This chapter contains these sections:

- [Choosing a Compiler](#)
- [Compiling Projects](#)

## Choosing a Compiler

Choose a compiler to determine how the IDE interprets source code. The IDE uses a *plug-in* compiler architecture. This architecture provides these features:

- Modularity—the IDE associates a specific compiler plug-in with a particular programming language or environment. For example, a compiler plug-in exists for C++ source code, and another compiler plug-in exists for Java source code.
- Flexibility—as new programming languages develop, the IDE can use new compiler plug-ins.

The IDE associates common file-name extensions with various plug-in compilers. For example, most Java files have the file-name extension `.java`. The IDE associates these files with the Java compiler. The **File Mappings** panel provides control over such associations.

# Compiling Projects

Compile projects to process the source files that comprise a computer program and generate object code. The compiler flags syntax errors in the source files.

Use these tasks to compile projects:

- Compile source files.
- Set the build order or link order.
- Update a project or its files.
- Create an executable file from a project.
- Run an application created from the project.
- Remove object code.

This section explains how to perform each task.

---

## Compiling Source Files

Use the **Compile** commands to compile source files into binary files. The IDE can compile a single file, multiple files, or all files in an open project.

1. Enable the Project window that contains the desired files to be compiled.
2. Select one or more files.
3. Choose **Project > Compile**.

The IDE compiles the selected files.

---

<b>NOTE</b>	The <b>Project</b> menu contains most commands for compiling and linking projects. However, depending on the project type, some commands might be disabled or renamed.
-------------	--

---

---

## Setting the Build and Link Order of Files

Use the **Link Order** view in the Project window to specify the order in which the compiler and linker process files. Establishing the proper link order prevents link

errors caused by file dependencies. The **Link Order** view is sometimes called the **Segments** view or **Overlays** view, depending on the target.

1. Click the **Link Order** tab in a Project window.
2. Click and drag files into the desired link order.

The IDE changes the link order. The build begins at the top of the link order, processes each file, and concludes at the bottom of the link order.

---

<b>NOTE</b>	The IDE uses the new link order during subsequent <b>Update</b> , <b>Make</b> , <b>Run</b> , and <b>Debug</b> operations.
-------------	---

---

## Updating Projects

Use the **Bring Up To Date** command to compile, but not link, the newly added, modified, and touched files in a project. Unlike the **Make** and **Run** commands, the **Bring Up To Date** command does not produce a binary file.

1. Select the project to update.
2. Choose **Project > Bring Up To Date**.

The IDE compiles all uncompiled project files.

---

## Making Executable Files

Use the **Make** command to compile the newly-added, modified, and touched files in a project, then link them into a binary file. Unlike the **Run** command, the **Make** command does not execute the binary file. The **Make** command is useful for creating dynamic link libraries (DLLs), shared libraries, code resources, or tools.

1. Select the project to make.
2. Choose **Project > Make**.

The IDE processes the project and creates a binary file.

## Running Application Projects

Use the **Run** command to perform these tasks:

- Compile and link a project (if necessary).
- Create a standalone application.
- Change project settings (if required).
- Save the application.
- Run the application.

Note, the **Run** command is not available if the project creates a non-executable file like a dynamic linked library (DLL), shared library, library, code resource, or tool.

1. Select the project to run.
2. Choose **Project > Run**.

---

## Synchronizing File Modification Dates

Use the **Synchronize Modification Dates** command to update the modification dates of all files stored in a project. This command is useful for handling files from a third-party editor that does not share file-status information with the IDE.

1. Select the project window.
2. Choose **Project > Synchronize Modification Dates**.

The IDE checks the file-modification dates and marks modified files for re-compilation.

---

## Removing Object Code

Use the **Remove Object Code** command to remove binary object code stored in the project file and reduce project size.

1. Open the desired project to remove object code.
2. Choose **Project > Remove Object Code**.

3. Set compaction options as desired.
  - Select **Recurse subprojects** to remove object code from all subprojects in the project file.
  - Select **Compact targets** to remove these items:
    - Target data files with the .tdt extension.
    - Browser data.
    - Dependency information.
    - Additional data cached by the IDE.
4. Select the method by which the IDE removes the object code.
  - Click **All Targets** to remove object code from all build targets.
  - Click **Current Target** to remove object code only from the active build target.

The IDE removes the specified object code from the project.

## **Compilers**

### *Compiling Projects*

---

# Linkers

---

This chapter explains how to work with linkers in the CodeWarrior™ IDE. The IDE uses linkers to complete these tasks:

- Combine code—the linker combines source-file object code with object code from library files and other related files. The combined code represents a complete computer program.
- Create a binary file—the linker processes the complete computer program and generates a binary file. Sample binary files include applications and shared libraries.

Read this chapter to learn more about typical tasks for working with linkers.

This chapter contains these sections:

- [Choosing Linkers](#)
- [Linking Projects](#)

## Choosing Linkers

Choose a linker to determine the binary file type produced by the IDE. This list describes common binary files:

- Applications—applications, or executable files, represent a wide body of computer programs. Common applications include word processors, web browsers, and multimedia players.
- Libraries—libraries contain code for use in developing new computer programs. Libraries simplify programming tasks and enhance re-usability.
- Specialized files—files designed for highly efficient operation in a specific context. Such files usually support a particular combination of hardware and software to perform tasks.

The IDE provides various linkers for software development. The **Target Settings** panel contains an option for selecting a linker. The IDE maps to each linker a group of recognized file-name extensions. These mappings determine how the IDE interprets each file.

# Linking Projects

Link projects to process object code and generate a binary file. Refer to the CodeWarrior *Targeting* documentation for more information about linkers for specific computer systems. This section explains general-purpose linker tasks.

---

## Generating Project Link Maps

Use the **Generate Link Map** command to create a link-map file that contains function and cross-section information about the generated object code. The link map reveals the files, libraries, and functions ignored by the IDE while producing the binary output.

The IDE stores the link-map file in the project folder. The file uses the same name as the build target, with a **.MAP** or **.xMAP** extension.

1. Select the project window.
2. Choose **Edit > target\_name Settings**.
3. Select the linker panel in the **Target Settings Panels** list.
4. Select the **Generate Link Map** option.
5. Click **Save**.
6. Choose **Project > Make**.

The IDE generates the link-map file.

# RAD (Rapid Application Development)

---

This section contains these chapters:

- [Creating RAD Projects](#)
- [Layout Editing](#)
- [Component Editing](#)
- [Object Wiring](#)
- [RAD Browsing](#)



# Creating RAD Projects

---

This chapter explains how to create Rapid Application Development (RAD) projects in the CodeWarrior™ IDE. RAD projects provide these programming aids:

- Graphical construction—instead of manually coding a user interface, use the RAD tools to create the interface visually. The IDE creates rudimentary code that implements the interface.
- Component manipulation—the RAD tools manipulate graphical items, or *components*, that make up a user interface.
- Catalog organization—use component collections, or *catalogs*, to quickly construct a user interface. Create custom catalogs of frequently used components to simplify RAD tasks.

Read this chapter to learn more about typical tasks for creating RAD projects.

This chapter contains these sections:

- [“RAD Wizards” on page 277](#)
- [“Working with Designs” on page 278](#)
- [“Working with Layouts” on page 282](#)

## RAD Wizards

Use RAD wizards to create RAD projects in the IDE. The wizards provide options for specifying RAD-related items:

- Names—specify the name of the RAD project, or the name of an item to place inside an existing RAD project.
- Locations—specify the location in which to store the RAD project, or the item to place inside an existing RAD project.
- Parameters—specify additional information about the project. The selected RAD wizard determines the available parameters.

The wizard uses these options to create a new RAD project.

## Using a RAD Wizard

Use the **New** command to create a RAD project.

1. Open the project.
2. Choose **File > New**.
3. Select the type of RAD project to create. These include:
  - Java Applet Wizard
  - Java Application Wizard
  - Java Bean Wizard
4. Type a name in the **Project name** text box and specify **Location**.
5. Click **OK**.
6. For each subsequent wizard window:
  - Enter the information requested, then click **Next**.  
OR
  - Click **Finish** at any time to complete the project using default values.The wizard builds the RAD project according to the specified custom or default parameters.

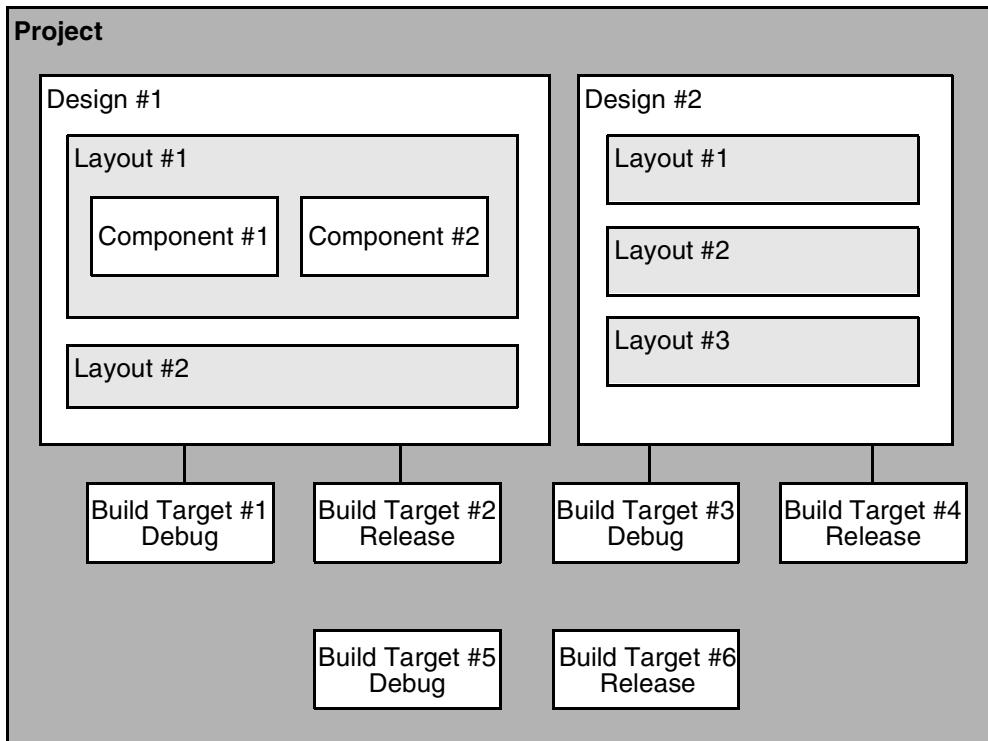
## Working with Designs

RAD projects contain one or more designs. A *design* organizes a particular set of user interfaces and components in an application. A design typically contains these items:

- Layouts—the layouts visually represent user interfaces that contain one or more interface components. For example, a design could contain layouts that depict different preference windows used in an application.
- Components—individual elements like controls, lists, or menus that comprise the user interface of a layout. For example, the design contains the components that make up the various layouts in an application.
- Build targets—the build targets used by the IDE to compile the design. For example, the design contains a debugging build target and a release build target.

Use the Design tab in the Project window to view a list of designs in the RAD project.

**Figure 25.1 Design hierarchy diagram**



---

## **Adding a Design to a Project**

Use the **New** command to add a design to a project.

1. Open the project.
2. Choose **File > New**.
3. Click the **Project** tab to display a list of available wizards.
4. Select the appropriate wizard for the design to add.

5. Enable the **Add Design to Project** checkbox.
6. Type a design name in the **Design Name** text box.
7. Choose the project's name from the **Project** list pop-up.
8. Click **OK**.
9. Click **Finish**.

The design is added to the project and displayed in the **Design** view of the project window.

---

## Removing a Design from a Project

Use the **Remove** command to remove a design from a project.

1. Click the **Targets** tab in the project window.
2. Select the design that you want to remove.
3. Remove the design from the project, as explained in [Table 25.1](#).

**Table 25.1 Removing the design from the project**

On this host...	Do this...
Windows	Select <b>Edit &gt; Delete</b> .
Macintosh	Select <b>Edit &gt; Clear</b> .
Solaris	Select <b>Edit &gt; Clear</b> .
Linux	Select <b>Edit &gt; Clear</b> .

Alternatively, use a contextual menu to remove the design, as explained in [Table 25.2 on page 281](#).

**Table 25.2 Removing a design by using a contextual menu**

On this host...	Do this...
Windows	Right-click the design and select <b>Delete</b> .
Macintosh	Control-click the design and choose <b>Clear</b> .
Solaris	Click and hold on the design, then choose <b>Clear</b> .
Linux	Click and hold on the design, then choose <b>Clear</b> .

The IDE removes the design from the project.

---

## **Viewing a Design**

Use the **Design** view tab in the project window to view all RAD designs in the current build target. Use the **Current Target** list pop-up to change which build target's designs to view.

- Click the **Design** view tab in the project window.

The current build target's designs are shown.

---

## **Creating an Empty Design**

Create an empty design in a legacy project to add design objects to that project. In order to add an empty design, at least one build target must not yet belong to an existing design in the project.

1. Open a project.

The project window appears.

2. Click the **Targets** tab in the project window.

The Targets view appears. This view shows the existing designs in the project and their corresponding build targets.

3. Choose **Project > Create Design**.

The **Create New Design** dialog box appears.

---

4. Enter a design name in the **Name for new design** field.
5. Select the checkbox next to each build target that you want to assign to the new design.

Clear the checkbox next to each build target that you do not want to assign to the new design.

---

**NOTE**

You must assign at least one build target to the new design.

---

6. Click **OK**.

The IDE creates an empty design and associates with it the build targets that you selected. The new design appears in the Targets view of the project window.

## Working with Layouts

RAD projects can contain one or more layouts. A *layout* visually organizes a particular set of components in a user interface. A layout contains these items:

- Components—the user-interface components that comprise a graphical interface. For example, you can create a layout that contains components that make up an alert box. The individual components include the alert-box window, the alert message, and an OK button.
- Views—the views of a complex user-interface. For example, you can create a multi-tabbed window in one layout. The layout could also contain the different states of the window that might appear as users perform various actions.

A pane in the layout editor contains a hierarchical list of the components in a layout.

---

### Adding a Layout to a Design

Use the **New** command to add a layout to a RAD design.

1. Click the **Design** tab in the project window.
2. Choose **File > New**.
3. Click the **Object** tab.

4. Select the **Java Frame Wizard**.
5. Select the project from the **Project** list pop-up.
6. Select the design from the **Design** list pop-up.
7. Click **OK**.
8. Specify the **Frame Class** information, then click **Finish**.

A new layout appears in the Design tab. Double-click this layout to open it and view its contents in the layout editor.

---

## **Removing a Layout from a Design**

Use the **Remove** command to remove a layout from a design.

1. Click the **Designs** tab in the project window.
2. Select the layout that you want to remove.
3. Remove the layout from the project, as explained in [Table 25.3](#).

**Table 25.3 Removing the layout from the project**

<b>On this host...</b>	<b>Do this...</b>
Windows	Select <b>Edit &gt; Delete</b> .
Macintosh	Select <b>Edit &gt; Clear</b> .
Solaris	Select <b>Edit &gt; Clear</b> .
Linux	Select <b>Edit &gt; Clear</b> .

Alternatively, use a contextual menu to remove the layout, as explained in [Table 25.4 on page 284](#).

**Table 25.4 Removing a layout by using a contextual menu**

<b>On this host...</b>	<b>Do this...</b>
Windows	Right-click the layout and select <b>Delete</b> .
Macintosh	Control-click the layout and choose <b>Clear</b> .
Solaris	Click and hold on the layout, then choose <b>Clear</b> .
Linux	Click and hold on the layout, then choose <b>Clear</b> .

The IDE removes the layout from the project.

# Layout Editing

---

This chapter explains how to edit Rapid Application Development (RAD) layouts in the CodeWarrior™ IDE. Use layouts to perform these tasks:

- Construct a user interface—a layout visually represents a user interface. Construct a layout with various components. For example, you can construct a dialog-box layout that contains button, scrollbar, and panel components.
- Manage components—organize related components into catalogs. For example, you can collect frequently used components into a single catalog.
- Manipulate components—examine or change each component in a layout. For example, you can change properties and events.

This chapter contains these sections:

- [“Layout Wizards” on page 285](#)
- [“Layout Editor” on page 286](#)
- [“Component Palette” on page 290](#)
- [“Object Inspector” on page 294](#)

## Layout Wizards

Use layout wizards to create new layouts for RAD projects. The IDE uses the information that you specify in a layout wizard to perform these tasks:

- name the layout.
- display the layout onscreen
- generate the source code that implements the layout at runtime

The wizard provides a series of steps that help you to create a new layout.

# Layout Editor

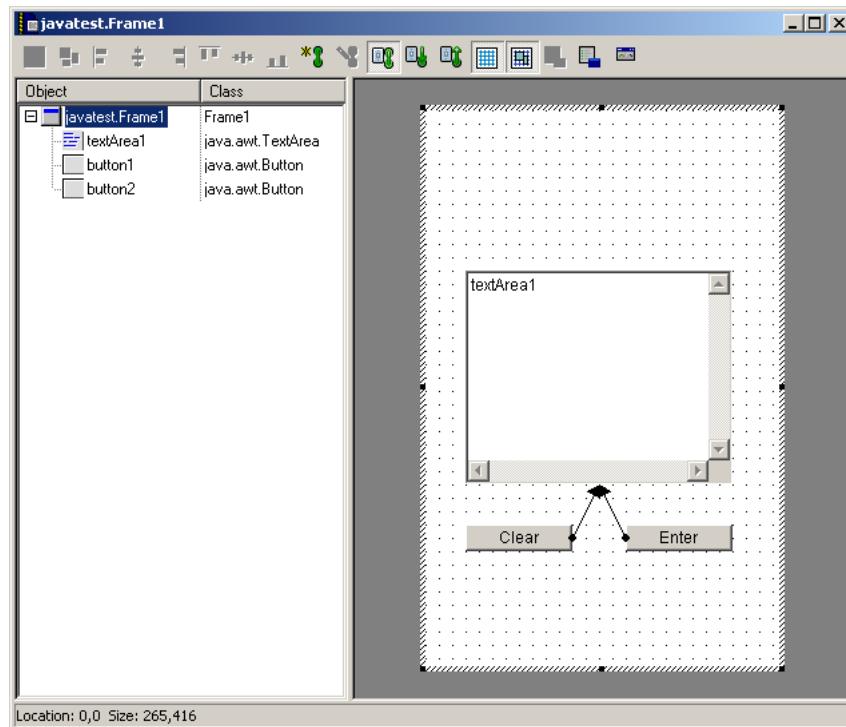
Use the **Layout Editor** to design a graphical user interface (GUI) for a RAD project. The IDE uses RAD plug-ins to display the layout in the Layout Editor.

Use the Layout Editor in conjunction with the Component Palette to perform these tasks:

- add components, like controls, lists, and menus, to an interface
- position interface components
- resize interface components
- view components in an interface
- remove components from an interface

[Figure 26.1](#) shows the Layout Editor. [Table 26.1 on page 287](#) explains the items in the Layout Editor.

**Figure 26.1 Layout Editor**



**Table 26.1 Layout Editor—items**

Item	Icon	Description
Group button		Select two or more components in the layout, then click this button to combine those selected components so that they move together as a group. Clicking this button is equivalent to selecting <b>Layout &gt; Group</b> .
Ungroup button		Select a group in the layout, then click this button to separate the components so that they move independently of one another. Clicking this button is equivalent to selecting <b>Layout &gt; Ungroup</b> .
Left Edges button		Select two or more components in the layout, then click this button to align the selected components along their left edges. Clicking this button is equivalent to selecting <b>Layout &gt; Align &gt; Left Edges</b> .
Vertical Center button		Select two or more components in the layout, then click this button to align the selected components along their vertical centers. Clicking this button is equivalent to selecting <b>Layout &gt; Align &gt; Vertical Center</b> .
Right Edges button		Select two or more components in the layout, then click this button to align the selected components along their right edges. Clicking this button is equivalent to selecting <b>Layout &gt; Align &gt; Right Edges</b> .
Top Edges button		Select two or more components in the layout, then click this button to align the selected components along their top edges. Clicking this button is equivalent to selecting <b>Layout &gt; Align &gt; Top Edges</b> .
Horizontal Center button		Select two or more components in the layout, then click this button to align the selected components along their horizontal centers. Clicking this button is equivalent to selecting <b>Layout &gt; Align &gt; Horizontal Center</b> .
Bottom Edges button		Select two or more components in the layout, then click this button to align the selected components along their bottom edges. Clicking this button is equivalent to selecting <b>Layout &gt; Align &gt; Bottom Edges</b> .
New Wire button		Click this button, then create a wire by clicking in the layout or by dragging the cursor from one component to another in the layout.
Edit Wire button		Select a wire in the layout, then click this button to edit the wire properties. Clicking this button is equivalent to selecting <b>Layout &gt; Edit Wire</b> .

**Table 26.1 Layout Editor—items (*continued*)**

Item	Icon	Description
Show All Wires button		Clicking this button toggles visibility of all wires in the layout. Clicking this button is equivalent to selecting <b>Layout &gt; Show All Wires</b> .
Show Incoming Wires button		Select a component in the layout, then click this button to view all wires that define the selected component as their destination component. Clicking this button is equivalent to selecting <b>Layout &gt; Show Incoming Wires</b> .
Show Outgoing Wires button		Select a component in the layout, then click this button to view all wires that define the selected component as their source component. Clicking this button is equivalent to selecting <b>Layout &gt; Show Outgoing Wires</b> .
Grid button		Clicking this button toggles visibility of a grid in the layout editor. Clicking this button is equivalent to selecting <b>Layout &gt; Display Grid</b> .
Snap To Grid button		Clicking this button toggles whether components snap to the grid in the layout editor as you move them in the layout. Clicking this button is equivalent to selecting <b>Layout &gt; Snap To Grid</b> .
Customize button		Select a component in the layout, then click this button to open a customization utility for the selected component. Clicking this button is equivalent to selecting <b>Layout &gt; Customize</b> . This button grays out if the component does not support a utility.
Properties button		Click this button to open the object inspector and manage the properties of the selected component. Clicking this button is equivalent to selecting <b>Layout &gt; Properties</b> .
Component Palette button		Click this button to open the Component Palette.
Component list		Lists all components in the current layout. In this list, you can expand and collapse view, drag a component to reorder it in the list, open a contextual menu for a selected component, and clear a selected component.
Layout manager		Shows the user interface for the current layout. The IDE generates source code that implements this user interface.

## Opening a Layout

To open a layout in a design, follow these steps.

1. Click the **Design** tab in the project window.
2. Double-click the layout name.

A layout editor appears and shows the layout that you double-clicked.

---

## Using Contextual Menus in the Layout Editor

Use the contextual menu in the layout editor to execute specific commands on the selected component. The available commands depend on the item that you select.

[Table 26.2](#) explains contextual-menu commands for the layout editor.

**Table 26.2 Contextual menu commands in the layout editor**

Command	Description
Align	Aligns selected objects first with the grid in the layout window, then second in relation to other selected objects. You can show or hide the grid in the layout editor.
Resize	Resizes selected components in relationship to other selected components
Send To Back	Moves selected objects so that they appear behind all other objects in the layout
Bring To Front	Moves selected objects so that they appears in front of all other objects in the layout
Display Grid	If checked, a grid appears in the layout window. If not checked, the grid does not appear.
Snap To Grid	If checked, the layout editor automatically aligns objects with the graphical grid. If not checked, you can freely move objects in the layout.
Customize	Opens a customization utility for the selected component. This command is available only for those components that support customization utilities.
Properties	Opens the object inspector, which shows the properties and events for the selected component.

**Table 26.2 Contextual menu commands in the layout editor**

Command	Description
New Wire	Opens the Create Wire wizard. Use this wizard to create a new wire in the layout.
Edit Wire	Opens the Create Wire wizard for the selected wire. Use this wizard to modify the wire details.

Follow these steps to use a contextual menu in the layout editor:

1. Select an item in the layout.
2. Open a contextual menu, as explained in [Table 26.3](#).

**Table 26.3 Using a contextual menu in the layout editor**

On this host...	Do this...
Windows	Right-click the item.
Macintosh	Control-click the item.
Solaris	Click and hold on the item.
Linux	Click and hold on the item.

3. Select a command in the contextual menu.

The IDE applies to the item the command that you select.

## Component Palette

Use the **Component Palette** to find and select components that you want to add to a layout in the layout editor. The Component Palette is a tool-based version of the Component Catalog. The components listed in the Component Catalog appear as buttons in the Component Palette. Click these buttons, then draw components in the layout.

Figure 26.2 Component Palette

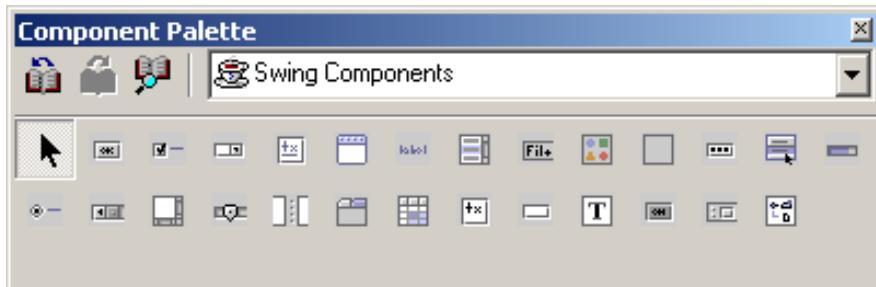


Table 26.4 Component Catalog window—items

Component	Icon	Description
Open Catalog button		Click to open a catalog file and add it to the All Catalogs list in the Component Catalog.
Close Catalog button		Click to close the active catalog shown in the <a href="#">Catalog list pop-up</a> .
Component Catalog button		Click to open the Component Catalog for the active catalog shown in the <a href="#">Catalog list pop-up</a> .
Catalog list pop-up	Swing Components	Use to select a component catalog.
Component tool buttons	(Sample tool)	Shows the available tools for the selected component catalog. Each tool button represents a component.

## Opening the Component Palette

Open the **Component Palette** to select components to add to a layout.

1. Open the **Component Palette** window, as explained in [Table 26.5 on page 292](#).

**Table 26.5 Opening the Component Palette window**

<b>On this host...</b>	<b>Do this...</b>
Windows	Select <b>View &gt; Component Palette</b> .
Macintosh	Select <b>Window &gt; Component Palette</b> .
Solaris	Select <b>Window &gt; Component Palette</b> .
Linux	Select <b>Window &gt; Component Palette</b> .

2. The component palette appears.



Alternatively, click the Component Palette button in the layout editor.

---

## **Adding a Component to a Layout**

You can add new components to a layout by using the **Component Palette** or the **Component Catalog**.

### **Component Palette**

Click the component that you want to use, then click in the layout shown in the layout editor to place that component in the layout.

The component appears in the layout.

### **Component Catalog**

1. Select a component in the Component Catalog.
2. Drag the selection from the Component Catalog to the layout shown in the layout editor.

The component appears in the layout.

---

## **Removing a Component from a Layout**

Remove a component that you no longer use from the layout shown in the layout editor.

1. Select the component that you want to remove.
2. Remove the component, as explained in [Table 26.6](#).

**Table 26.6 Removing a component**

On this host...	Do this...
Windows	Select <b>Edit &gt; Delete</b> .
Macintosh	Select <b>Edit &gt; Clear</b> .
Solaris	Select <b>Edit &gt; Clear</b> .
Linux	Select <b>Edit &gt; Clear</b> .

The IDE removes the component from the layout.

---

## Moving a Component in a Layout

Follow these steps to reposition components in the layout editor:

1. Select the component that you want to move.  
Shift-click additional components that you want to move.
2. Reposition the selected components.
  - Click and drag the components to a new position.
  - Press the arrow keys to move the components to a new position.

The IDE repositions the selected components.

---

## Resizing a Component in a Layout

Use the **Resize Component** command to resize the selected component in a layout. Resize handles appear around a selected component, one in each corner and midway along each side. Use these resize handles to resize the selected component.

A heavy outline appears as you drag a resize handle. This outline represents the final shape of the component after you release the cursor.

1. Select the component that you want to resize.  
Resize handles appear around the component.
2. Click and drag one of the resize handles.  
The heavy outline appears.
3. Release the cursor after resizing the component.  
The component changes size.

## Object Inspector

Use the **Object Inspector** to display and edit these items in the layout:

- component properties
- events
- wires

Use dialog boxes, list pop-up menus, contextual menus, or the keyboard to enter information about these items.

Figure 26.3 Object Inspector

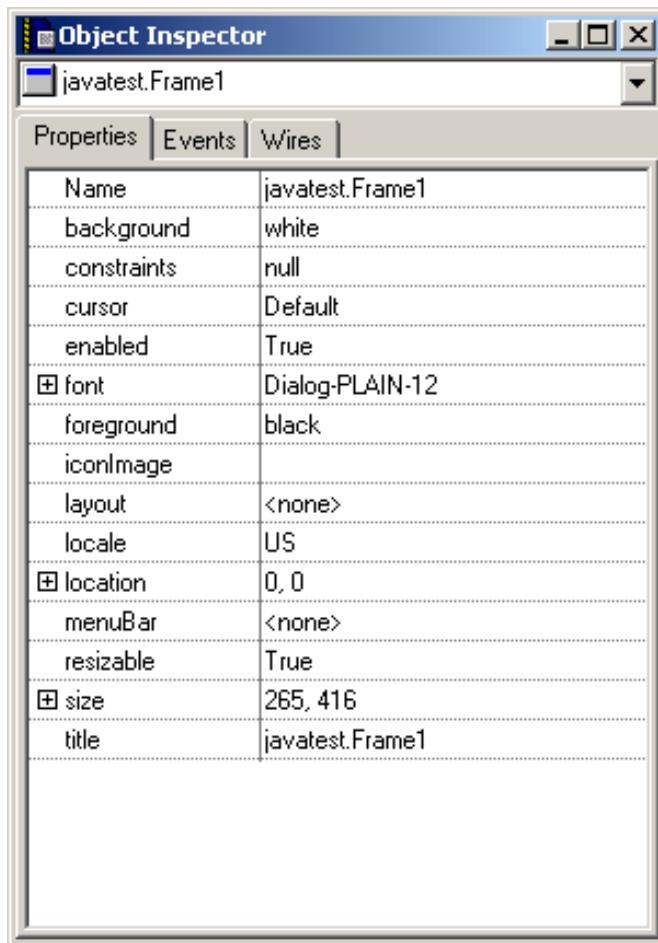


Table 26.7 Object Inspector—items

Component	Icon	Description
Object list pop-up		Select an object to edit its properties and events.
Properties tab		Click to show and edit properties for the component shown in the <a href="#">Object list pop-up</a> .

**Table 26.7 Object Inspector—items (continued)**

Component	Icon	Description
Events tab		Click to show and edit events for the component shown in the <a href="#">Object list pop-up</a> .
Wires tab		Click to display and edit the wires defined in the layout.
Options list pop-up		Select a preset value and assign it to a property.
Show Options button		Click to assign a set of values to the property.

---

## Opening the Object Inspector

Open the **Object Inspector** to examine and modify component properties and events.

1. Open the Object Inspector, as explained in [Table 26.8](#).

**Table 26.8 Opening the Object Inspector**

On this host...	Do this...
Windows	Select <b>View &gt; Object Inspector</b> .
Macintosh	Select <b>Window &gt; Object Inspector</b> .
Solaris	Select <b>Window &gt; Object Inspector</b> .
Linux	Select <b>Window &gt; Object Inspector</b> .

2. The object inspector appears.

Alternatively, use a contextual menu to open the Object Inspector, as explained in [Table 26.9](#).

**Table 26.9 Opening the Object Inspector by using a contextual menu**

On this host...	Do this...
Windows	Right-click the component and select <b>Properties</b> .
Macintosh	Control-click the component and select <b>Properties</b> .

**Table 26.9 Opening the Object Inspector by using a contextual menu (*continued*)**

<b>On this host...</b>	<b>Do this...</b>
Solaris	Click and hold on the component, then select <b>Properties</b> .
Linux	Click and hold on the component, then select <b>Properties</b> .



# Component Editing

---

This chapter explains how to edit Rapid Application Development (RAD) components in the CodeWarrior™ IDE. Component editing covers these categories:

- Component catalogs—organize related components into groups or catalogs. For example, create a catalog to store frequently used components.
- Menu editors—the IDE provides specialized editors for constructing menus and pop-up menus in RAD projects.

Read this chapter to learn more about typical tasks for editing components.

This chapter contains these sections:

- [“Component Catalog Window” on page 299](#)
- [“Working with Catalogs” on page 302](#)
- [“Working with Menu Editors” on page 304](#)

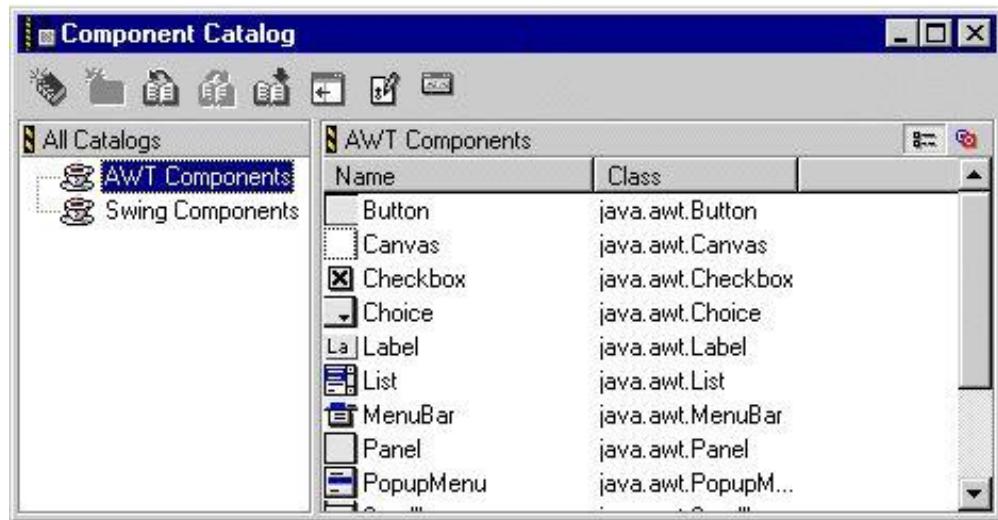
## Component Catalog Window

Use the **Component Catalog** window to manage catalog files and their graphical user interface (GUI) components. Use this window to perform these tasks:

- create new component catalog files
- open and close multiple catalog files
- import or copy GUI components from one catalog into another
- view GUI components within catalog files
- access GUI component properties for editing

**Component Editing**  
*Component Catalog Window*

**Figure 27.1 Component Catalog window**



**Table 27.1 Component Catalog window—items**

Component	Icon	Description
New Catalog		Click to create a new catalog file.
New Folder		Click to create a new folder within the active catalog.
Open Catalog		Click to open a catalog file and add it to the All Catalogs list.
Close Catalog		Click to close the selected catalog in the All Catalogs list.
Import Components		Click to import .jar files (beans) and add it to the All Catalogs list.
Toggle Index View		Click to toggle the appearance of the Catalog list.

**Table 27.1 Component Catalog window—items (*continued*)**

Component	Icon	Description
Edit Item Properties		To open an Object Inspector window for editing of the selected GUI component, click Edit Item Properties.
Component Palette		To open the active catalog's GUI component palette, click Component Palette.
All Catalogs list		Displays a list of all current open catalog files.
Components list		Displays a list of all GUI components stored in the active catalog file. To edit a component, select it, then click Edit Item Properties.
List View		Click the List View icon to display the active catalog's GUI components in the Components list in list format.
Live View		Click the Live View icon to display the active catalog's GUI components in the Components list in active graphic form.

---

## Opening the Component Catalog

Open the **Component Catalog** to create, manipulate, and manage component catalog files.

1. Open the Component Catalog, as explained in [Table 27.2](#).

**Table 27.2 Opening the Component Catalog**

On this host...	Do this...
Windows	Select <b>View &gt; Component Catalog</b> .
Macintosh	Select <b>Window &gt; Component Catalog</b> .
Solaris	Select <b>Window &gt; Component Catalog</b> .
Linux	Select <b>Window &gt; Component Catalog</b> .

2. The Component Catalog appears.

# Working with Catalogs

Use these tasks to work with catalogs:

- Manage components.
- Create and remove catalogs.
- Add and remove components in catalogs.

This section explains how to perform each task.

---

## Creating a Component Catalog

Use the **New** command to create component catalog files for storing frequently used or customized RAD components.

1. Open the project.
2. Choose **File > New**.
3. Click the **File** view tab in the **New** window.
4. Select **Component Catalog File** from the File list.
5. Type a name in the **File Name** text box.
6. Specify a location in the **Location** text box.
7. Click **OK**.

The IDE creates the new catalog file and displays it in the component catalog.

Alternatively, click the New Catalog toolbar button in the Component Catalog in order to create a new catalog file.

---

## Adding Components to a Catalog File

Populate a catalog file with the components that you use frequently. In adding components to a catalog file, you copy one or more components from an existing catalog file.

1. Open the **Component Catalog**.
2. Select a catalog file in the **All Catalogs** list to display its components.
3. Add one or more components to the catalog file.

## **Adding a single component**

Click-and-drag the selected component from the **Components** list onto the destination catalog file shown in the **All Catalogs** list.

## **Adding multiple components simultaneously**

1. Select the components, as explained in [Table 27.3](#).

**Table 27.3 Selecting components**

<b>On this host...</b>	<b>Do this...</b>
Windows	Ctrl-click each component.
Macintosh	Shift-click each component.
Solaris	Shift-click each component.
Linux	Shift-click each component.

2. Click-and-drag the selected components from the **Components** list onto the destination catalog file shown in the **All Catalogs** list.

## **Adding components by using copy and paste**

1. Select the components in the **Components** list that you want to copy.
2. Select **Edit > Copy**.
3. Select the destination catalog file in the **All Catalogs** list.
4. Select **Edit > Paste**.

## Removing Components from a Catalog File

Remove from a component catalog file the components that you no longer use.

1. Open the **Component Catalog**.
2. Select the component catalog file in the **All Catalogs** list to display its components.
3. Select the component that you want to remove.
4. Remove the component, as explained in [Table 27.4](#).

**Table 27.4 Removing the component**

On this host...	Do this...
Windows	Select <b>Edit &gt; Delete</b> .
Macintosh	Select <b>Edit &gt; Clear</b> .
Solaris	Select <b>Edit &gt; Clear</b> .
Linux	Select <b>Edit &gt; Clear</b> .

The IDE removes the selected component from the catalog file.

## Working with Menu Editors

Perform these tasks to work with the menu editor:

- Manage menus or menu items.
- Create and remove menus or menu items.
- Modify menus or menu items.

This section explains how to perform each task.

## Menu Editor

Use the **Menu Editor** to display and edit RAD menus and pop-up menus. To open a Menu Editor, double-click any menubar or pop-up menu component in the layout editor.

**Figure 27.2** Menu Editor window



**Table 27.5** Menu Editor window—items

Item	Icon	Explanation
Menu		A named menu in the menu bar.
Menu placeholder		An unnamed menu in the menu bar.
Menu Item placeholder		An unnamed menu item in the menu.

## Creating a Menu or Menu Item

Use the Menu Editor to add menus and menu items to a menubar component in the layout.

1. Select the menu component that you want to edit.
2. Select **Layout > Customize**.
3. The Menu Editor appears.
4. Double-click the placeholder for the menu or menu item in the Menu Editor (the placeholder appears as a rectangle).
5. Enter menu or menu-item information in the **Menu Item Attributes** window.
6. Click **OK**.

The menu or menu item appears in the Menu Editor. Continue customizing the menu, then close the Menu Editor window.

Alternatively, you can open the Menu Editor by performing one of these tasks:

- double-clicking the menu component in the layout editor
- selecting the menu component in the layout and clicking the Customize toolbar button in the layout editor
- using a contextual menu, as explained in [Table 27.6](#).

**Table 27.6 Opening the Menu Editor by using a contextual menu**

On this host...	Do this...
Windows	Right-click the menu component and select <b>Customize</b> .
Macintosh	Control-click the menu component and select <b>Customize</b> .
Solaris	Click and hold on the menu component, then select <b>Customize</b> .
Linux	Click and hold on the menu component, then select <b>Customize</b> .

## Removing a Menu or Menu Item

Remove from the Menu Editor the menus and menu items that you no longer use.

1. Double-click a menubar component in the layout editor.
2. Select the menu or menu item in the Menu Editor that you want to remove.
3. Remove the menu or menu item, as explained in [Table 27.7](#).

**Table 27.7 Removing a menu or a menu item**

On this host...	Do this...
Windows	Select <b>Edit &gt; Delete</b> .
Macintosh	Select <b>Edit &gt; Clear</b> .
Solaris	Select <b>Edit &gt; Clear</b> .
Linux	Select <b>Edit &gt; Clear</b> .

4. Click **OK**.

The IDE removes the menu or menu item.

Alternatively, select the item that you want to remove and press **Delete**.

---

## Modifying a Menu or Menu Item

Use the Menu Editor to modify menus and menu items on a menubar component in a RAD layout.

1. Double-click a menubar component in the **Layout Editor**.
2. Double-click the menu or menu item in the **Menu Editor** window to modify.
3. Change the menu or menu item information in the **Menu Item Attributes** window.

4. Click **OK**.

The modified menu or menu item is shown in the Menu Editor window.

# Object Wiring

---

This chapter explains how to use object wiring in the CodeWarrior™ IDE. Object wiring establishes interactions between Rapid Application Development (RAD) components. To establish these relationships, you draw wires onscreen to connect one component to a related component.

Object wiring has these benefits:

- Visual metaphor—wires between objects indicate an interaction between those objects
- Source-code generation—after you draw each wire, the IDE generates a basic function or method that implements the wire in source code

This chapter contains these sections:

- [“About Object Wiring” on page 309](#)
- [“Working with Object Wires” on page 310](#)
- [“Using the Create Wire Wizard” on page 315](#)
- [“Inspecting Wires” on page 317](#)

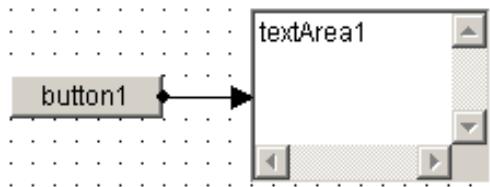
## About Object Wiring

A *wire* is a one-way relationship between two RAD components. The wire connects a *source* component to a *destination* component. A dot on one end of the wire denotes the source. An arrow at the other end of the wire denotes the destination.

An *action* is the operation that the destination component performs after an event on the source component occurs. The wire is a visual representation of this action.

[Figure 28.1 on page 310](#) shows a sample wire connecting a source component to a destination component. [Table 28.1 on page 310](#) defines the source component, destination component, and wire in terms of [Figure 28.1 on page 310](#).

**Figure 28.1 Sample wire**



**Table 28.1 Sample wire—items**

Item	Icon	Explanation
Source component		The component on which an event triggers an operation on the destination component.
Wire		Connects the source component to the destination component. The dot denotes the source and the arrow denotes the destination.
Destination component		The component on which an operation occurs as a result of an event on the source component.

You work with object wiring in these IDE elements:

- layout editor—draw wires, modify wires, and delete wires either graphically or by using a contextual menu
- **Create Wire** wizard—define the action represented by the wire
- object inspector—reorder wires with the same source component and action, and delete wires

## Working with Object Wires

You work with object wires in these ways:

- create a wire
- modify a wire
- delete a wire

---

This section explains how to perform these tasks.

---

## Creating a Wire by Drawing It

Draw wires in the layout editor to connect a source component to a destination component. The wire represents an action on the destination component.

To view the layout editor, open a RAD project and open a layout from the **Design** tab in the project window.

1. Identify the source and destination component in the layout.

These components have an interaction that you define by drawing a wire between them.

---

### NOTE

You do not need to select a source component or destination component in order to create a wire. In this case, the **Create Wire** wizard substitutes default values in place of the missing components. You can change these default values in the wizard.

---

2. Click the **New Wire** button in the layout editor toolbar: 
- The button highlights.
3. Position the cursor over the source component.
4. Drag the cursor from the source component to the destination component.  
The wire appears as you drag the cursor. Draw the wire so that it begins on top of the source component and ends on top of the destination component.
5. Release the cursor.  
The **Create Wire** wizard appears. Complete this wizard to specify the details of the wire.

## Creating a Wire by Using a Menu

Use the **New Wire** menu command to create a wire in the layout editor. To view the layout editor, open a RAD project and open a layout from the **Design** tab in the project window.

1. Select a source component in the layout.

Handles appear on the selected component.

<b>NOTE</b>	You do not need to select a source component in order to create a wire. In this case, the <b>Create Wire</b> wizard substitutes the layout itself in place of the missing source component.
-------------	---

2. Select **Layout > New Wire**.

The Layout menu becomes visible after you open the layout.

The Create Wire wizard appears. Complete this wizard to specify the details of the wire, such as the destination component and action.

---

## Creating a Wire by Using a Contextual Menu

Use the **New Wire** command in the layout editor contextual menu to create a wire in the layout. To view the layout editor, open a RAD project and open a layout from the **Design** tab in the project window.

1. Select a source component in the layout.
- Handles appear on the selected component.
2. Open the Layout Editor contextual menu, as explained in [Table 28.2](#).

**Table 28.2 Opening the Layout Editor contextual menu**

On this host...	Do this...
Windows	Right-click the selected component.
Macintosh	Control-click the selected component.

**Table 28.2 Opening the Layout Editor contextual menu (*continued*)**

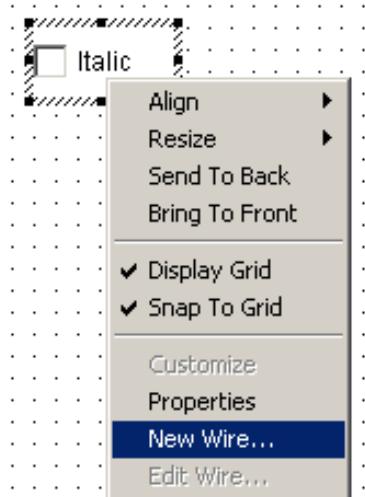
On this host...	Do this...
Solaris	Control-click the selected component.
Linux	Ctrl-click the selected component.

3. Select **New Wire** from the contextual menu.

[Figure 28.2](#) shows the contextual menu.

The Create Wire wizard appears. Complete this wizard to specify the details of the wire, such as the destination component and action.

**Figure 28.2 Layout Editor contextual menu - New Wire**



## Modifying a Wire by Using the Layout Editor

Modify an existing wire as you develop your layout in the layout editor. Use the **Create Wire** wizard to make changes to the wire data.

1. Bring forward the layout editor window.
2. Select the wire that you want to modify.

**TIP** You can click different parts of a particular wire to select it quickly.

If few wires connect the same source and destination component, try clicking the dot or arrow of a particular wire to select it quickly.

If many wires connect the same source and destination component, try clicking the wire itself to select it quickly.

---

3. Choose **Layout > Edit Wire**.

Alternatively, double-click the wire or use the layout editor contextual menu.

4. The Create Wire wizard appears.

Use the wizard to modify the current data for the wire.

---

## Deleting a Wire by Using the Layout Editor

Delete an existing wire in your layout to remove the interaction between its source and destination components.

While deleting the wire removes it from the layout, the IDE does not remove the generated source code that implements the wire.

This behavior helps preserve your source code. For example, if you modify the generated functions or methods to offer additional functionality, the IDE does not delete your work after you delete the wire. Instead, the IDE removes the wire from the layout.

1. Bring forward the layout editor window.

2. Select the wire that you want to delete.

Handles appear on the selected wire.

3. Select **Edit > Clear**.

The IDE removes the selected wire from the layout.

# Using the Create Wire Wizard

After you create a wire in the layout, the IDE collects this information:

- source component—the object on which an event triggers the wire
- destination component—the object on which an action occurs as a result of triggering the wire
- source event—the event that triggers the wire
- action—the operation that the destination component performs
- action parameters—details that qualify the action

Depending on how you create the wire, the IDE already has some of this information:

- create a wire between two components—source and destination components already identified
- create a wire with one component selected—source component already identified

Use the **Create Wire** wizard to customize any remaining information.

---

## Completing the Create Wire Wizard

Use the Create Wire wizard to specify details for wires that you create.

1. Bring forward the layout editor window.
2. Create a wire.

Draw the wire in the layout itself, or use menu commands to create the wire.

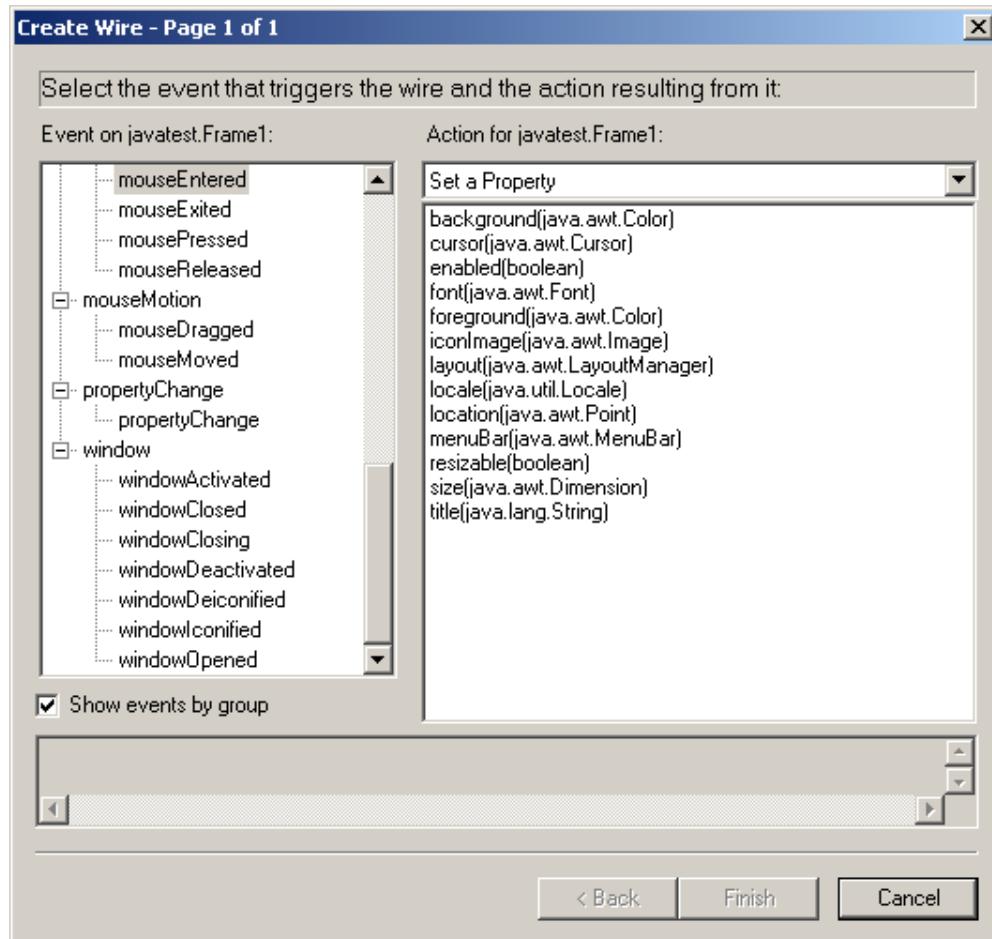
3. The Create Wire wizard appears.

Depending on how you created the wire, the wizard displays different pages to help you complete the wire. [Figure 28.3 on page 316](#) shows a sample page in the Create Wire wizard.

## Object Wiring

Using the Create Wire Wizard

**Figure 28.3 Create Wire wizard**



4. Specify the event that triggers the wire.
  - a. Scroll though the list of available events.
  - b. Select an event.

**NOTE**

Clear the **Show events by group** checkbox to list available events in alphabetical order. Select the checkbox to arrange available events by group.

5. Specify the destination component.

If you selected a destination component when creating the wire, the Create Wire wizard already has this information.

6. Specify the action performed on triggering the wire.

a. Select an action type from the pop-up menu:

- **Set a Property**—on the destination component
- **Call a Method**—on the destination component
- **Perform an Action**—(predefined action) on the destination component
- **Execute Your Own Code**—after the wire triggers

b. Select an action.

The available actions depend on the selected action type.

7. Complete additional information presented in the Create Wire wizard.

This information completes the data the IDE needs to implement the wire in source code.

8. Click **Finish**.

The Create Wire wizard displays a **Summary** dialog box.

9. Click **Generate** to create the wire, or click **Cancel** to discard the wire.

If you generate the wire, it appears in the layout. The IDE generates a basic wire implementation in source code, using your supplied information to complete the process.

---

<b>NOTE</b>	If you choose to <b>Execute Your Own Code</b> , an editor window opens and scrolls to the generated source code. Insert your source code at this location.
-------------	--

---

## Inspecting Wires

Use the Object Inspector to perform these tasks with wires:

- view wires defined in the current layout
- re-order wires with the same source component and action

- modify wires
- delete wires

---

## **Viewing Wires by Using the Object Inspector**

You can use the object inspector to see a list of wires currently defined in the layout editor.

1. Open the object inspector for the active RAD project:
2. Click the **Wires** tab in the object inspector.

Wire information for the current layout appears in the Wires tab.

Use the hierarchical controls in the Wires tab to collapse or expand the wire hierarchy.

---

## **Re-ordering Wires by Using the Object Inspector**

Use the object inspector to change the order of wires that have the same source component and action. The order listed in the object inspector represents the order of wire precedence.

1. Open the object inspector for the active RAD project:
  2. Click the **Wires** tab in the object inspector.
- The Wires tab appears.
3. Drag the wires listed in the Wires tab to change their order.

The order of wire precedence begins at the top of the hierarchical list and proceeds to the bottom of the list. For example, if Wire A appears above Wire B, Wire A has precedence over Wire B.

## Modifying a Wire by Using the Object Inspector

In addition to modifying a wire by using the layout editor, you can modify the wire by using the object inspector. Use the **Create Wire** wizard to make changes to the wire data.

1. Open the object inspector for the active RAD project.

2. Click the **Wires** tab in the object inspector.

The Wires tab appears.

3. In the Wires tab, find the wire that you want to modify.

Expand or collapse hierarchical controls in order to find the wire.

4. Double-click the wire in the Wires tab.

The Create Wire wizard appears. Use the wizard to modify the current data for the wire.

---

## Deleting a Wire by Using the Object Inspector

In addition to deleting a wire by using the layout editor, you can delete the wire by using the object inspector.

While deleting the wire removes it from the layout, the IDE does not remove the generated source code that implements the wire.

This behavior helps preserve your source code. For example, if you modify the generated functions or methods to offer additional functionality, the IDE does not delete your work after you delete the wire. Instead, the IDE removes the wire from the layout.

1. Open the object inspector for the active RAD project.

2. Click the **Wires** tab in the object inspector.

The Wires tab appears.

3. Select in the Wires tab the wire that you want to delete.

4. Press **Delete**:

The IDE removes the wire from the layout.

---

## Sorting the Wires Tab by Using a Contextual Menu

Use a contextual menu to change wire sorting in the Wires tab of the object inspector.

1. Open the object inspector for the active RAD project.
2. Click the **Wires** tab in the object inspector.

The Wires tab appears.

3. Right-click inside the Wires view.

A contextual menu appears.

4. Select the sorting method that you want to apply to the Wires tab. [Table 28.3](#) explains the resulting hierarchy after selecting each sorting method.

**Table 28.3 Selecting a sorting method**

Sorting Method	Resulting Hierarchy
View By Source Object	1. Source components 2. Wires associated with each source component
View By Source Object And Event	1. Source components 2. Events associated with each source component 3. Wires associated with each event
View By Destination Object	1. Destination components 2. Wires associated with each destination component

5. The IDE applies to the Wires tab the sorting method that you select.

# RAD Browsing

---

This chapter explains how to browse Rapid Application Development (RAD) projects in the CodeWarrior™ IDE. The browser generates additional information for RAD projects:

- Properties—the properties of a selected component in the RAD project. View the property names and their implementations in source code.
- Methods—the functions or methods of a selected component in the RAD project. View the method names and their implementations in source code.
- Events—the actions or events of a selected component in the RAD project. View the event names and their groupings within *event sets*.

Read this chapter to learn more about typical tasks for browsing RAD projects.

This chapter contains these sections:

- [“RAD Browser Window” on page 321](#)
- [“Browser RAD Wizards” on page 323](#)

## RAD Browser Window

The Browser window displays additional items for RAD projects:

- Tab control
- Properties tab
- Methods tab
- Events tab

This section explains each item.

### Tab Control

The browser displays a tab control for browsing the additional information generated for RAD projects. The tab control shows a series of tabs with different names. For example, the tab control for a Java RAD application contains these tabs:

- Java
- Properties
- Methods
- Events

Click a tab to view its browser information. For example, click the Properties tab to view information about component properties and source-code implementations.

---

<b>NOTE</b>	Some tabs display information only for public data not inherited from other member functions. For these tabs, the Browser Access Filters pop-up menu and the <b>Show Inherited</b> checkbox gray out.
-------------	---

---

## Properties Tab

The Properties tab displays information for a selected component in the Classes pane of the browser window. The tab contains these panes:

- Properties—view the properties of the selected component.
- Implementation—view the property implementation in the source code.

The browser allows source-code editing within the Implementation pane. The pane also shows the path to the source file.

## Methods Tab

The Methods tab displays information for a selected component in the Classes pane of the browser window. The tab contains these panes:

- Methods—view the functions or methods of the selected component.
- Implementation—view the method implementation in the source code.

The browser allows source-code editing within the Implementation pane. The pane also shows the path to the source file.

## Events Tab

The Events tab displays information for a selected component in the Classes pane of the browser window. The tab contains these panes:

- Event Sets—view a list of event sets for the selected component. Each event set contains one or more events.

- Events—view the events within the selected event set.

The browser allows source-code editing within the Implementation pane. The pane also shows the path to the source file.

## Browser RAD Wizards

The Browser menu contains commands for accessing various RAD wizards. Use these wizards to create new RAD items for which the browser generates information. These are sample browser RAD wizards:

- New Property—create a new property for a selected component.
- New Method—create a new method for a selected component.
- New Event Set—create a new event set for a selected component. Event sets contain one or more events.
- New Event—create a new event for a selected event set.

The selected browser RAD wizard steps through the process of creating the new item. After completing the wizard, the browser window updates to display information for the newly created item.

---

### Invoking the New Wizard

Use the **New Wizard** command on a selected component to assist in the creation of new:

- properties
- methods
- event sets
- events

for that component.

1. Select the component to modify.
2. Choose the appropriate new wizard:
  - **Browser > New Property** to add a new property.
  - **Browser > New Method** to add a method.
  - **Browser > New Event Set** to add an event set.

- **Browser > New Event** to add an event.
3. Enter the requested information into the window text boxes.
  4. Click **Add**.

The new property, method, event set, or event is added to the component.

# Preferences and Target Settings

---

This section contains these chapters:

- [Customizing the IDE](#)
- [Working with IDE Preferences](#)
- [Working with Target Settings](#)
- [Preference and Target Settings Options](#)



# Customizing the IDE

---

The CodeWarrior™ IDE enables you to customize menus, toolbars, and key bindings to suit your programming preferences. Use the **Customize IDE Commands** window—which consists of the Commands, Toolbar Items, and Key Bindings tabs—to build your customizations.

This chapter contains these sections:

- [“Customizing IDE Commands” on page 327](#)
- [“Customize Toolbars” on page 338](#)
- [“Customize Key Bindings” on page 344](#)

## Customizing IDE Commands

You can customize the menu commands in the IDE’s menu bar, as well as control the appearance of specific menu commands, create new command groups to distinguish menu commands, and associate a command line (Windows, Solaris, and Linux) or a script or application (Mac OS) with a new menu command. The customized menu commands you create have access to IDE information, such as the current editor selection, the frontmost window, and the current project and its output file.

[Figure 30.1 on page 328](#) shows the Customize IDE Commands window. [Table 30.1 on page 328](#) explains the buttons in the window.

## Customizing the IDE

### Customizing IDE Commands

Figure 30.1 Customize IDE Commands window

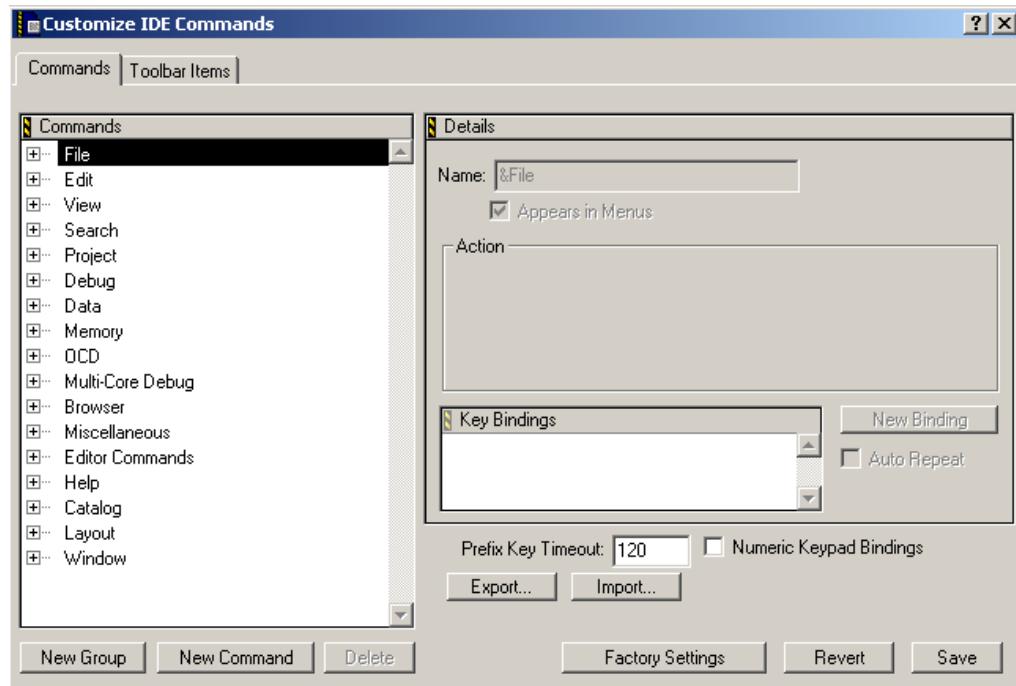


Table 30.1 Customize IDE Commands window—button explanations

Button name	Explanation
New Group	Click to add a new command group to the Commands list.
New Command	Click to add a new command setting within a group.
Factory Settings	Click to restore the default options for the current Customize IDE Commands (Commands and Toolbar Items) lists.
Revert	Click to restore the most recently saved options for the current Customize IDE Commands (Commands and Toolbar Items) lists.
Export	Click to save a file that contains commands and key bindings to use later in the Customize IDE Commands lists.
Import	Click to open a file that contains commands and key bindings to use in the current Customize IDE Commands lists.
Save	Click to save customizations to the Customize IDE Commands list.

## Commands Tab

Click the **Commands** tab at the top of the Customize IDE Commands window to display the commands view. Use this view to modify existing menu commands, and to create and remove command groups and menu commands.

---

### Modifying Existing Commands

You can use the **Commands** tab of the Customize IDE Commands window to examine and modify existing command groups and menu commands. This view includes a Commands list. This hierarchical list organizes individual menu commands into command groups. Click the hierarchical control next to a command group to expand that group and view its contents.

To examine a particular item, select it in the Commands list. The information for the selected item appears on the right-hand side of the Customize IDE Commands window. This window provides this information for each selected item:

- **Name**—This field shows the name of the selected item. If the IDE does not permit you to change the name of the selected items, this field is grayed out.
- **Appears in Menus**—Enable this checkbox to display the selected item in the specified position in the CodeWarrior menu bar. For example, enabling this checkbox for the **RunApp** menu command allows that menu command to appear under the **MyMenu** command group in the menu bar. Disabling the checkbox prevents the RunApp menu command from appearing in the menu bar under the MyMenu command group.
- **Action**—This section shows information about the action the selected item performs. For default menu commands, this section shows the command type, such as **Command** or **Hierarchical Menu**. For customized menu commands that you create, this section lets you specify a command line (Windows, Solaris, and Linux) or a script (Mac OS) that runs after you choose the customized menu command.
- **Key Bindings**—This area consists of the Key Bindings list, the **New Binding** button, and the **Auto Repeat** checkbox.

## Creating a New Command Group

To create a new command group for menu commands, follow these steps:

1. Click the **New Group** button.

The IDE creates a new command group called **New Group**, adds it to the Commands list, and displays its information in the Customize IDE Commands window.

2. Rename the new command group in the **Name** field.

Change the default name of New Group to describe your new command group.

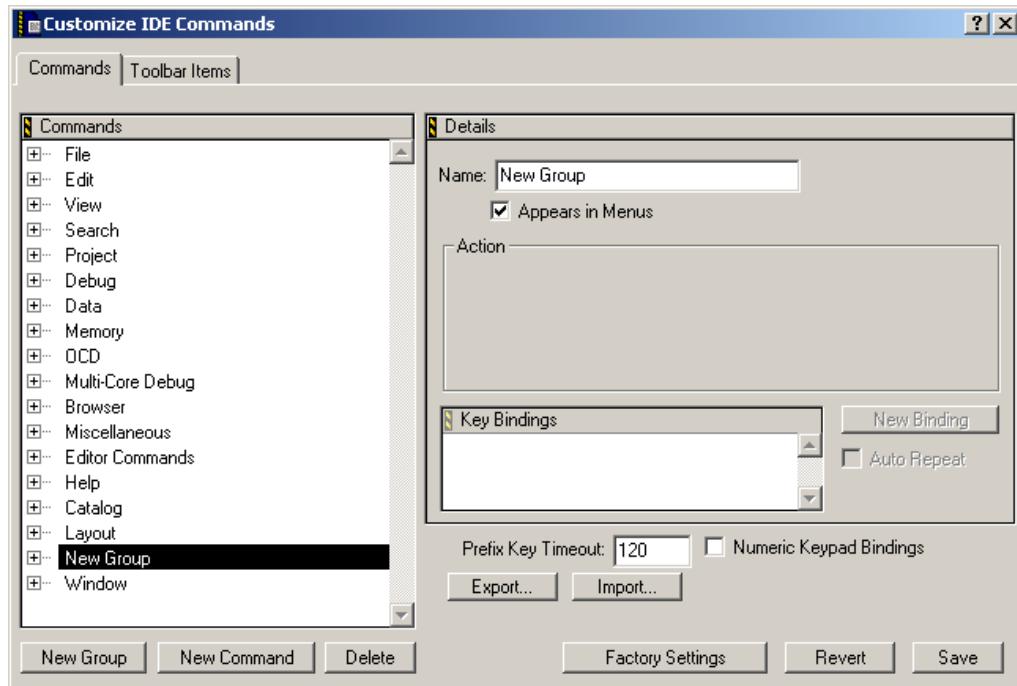
3. Use the **Appears in Menus** checkbox to toggle the appearance of the new command group in the IDE menu bar.

Select the Appears in Menus checkbox to display the new command group in the menu bar. Clear the checkbox if you do not want the command group to appear in the menu bar.

4. Click **Save**.

The IDE saves your new command group. If you selected the **Appears in Menus** checkbox, your new command group appears in the menu bar.

Figure 30.2 New Command group



## Creating a New Menu Command

To create a new menu command, follow these steps:

1. Select the command group you want to contain the new menu command.

You must select an existing command group in the Commands list.

2. Click the **New Command** button.

The IDE creates a new menu command named **New Command** and places it within the selected command group. The information for the new menu command appears in the Customize IDE Commands window.

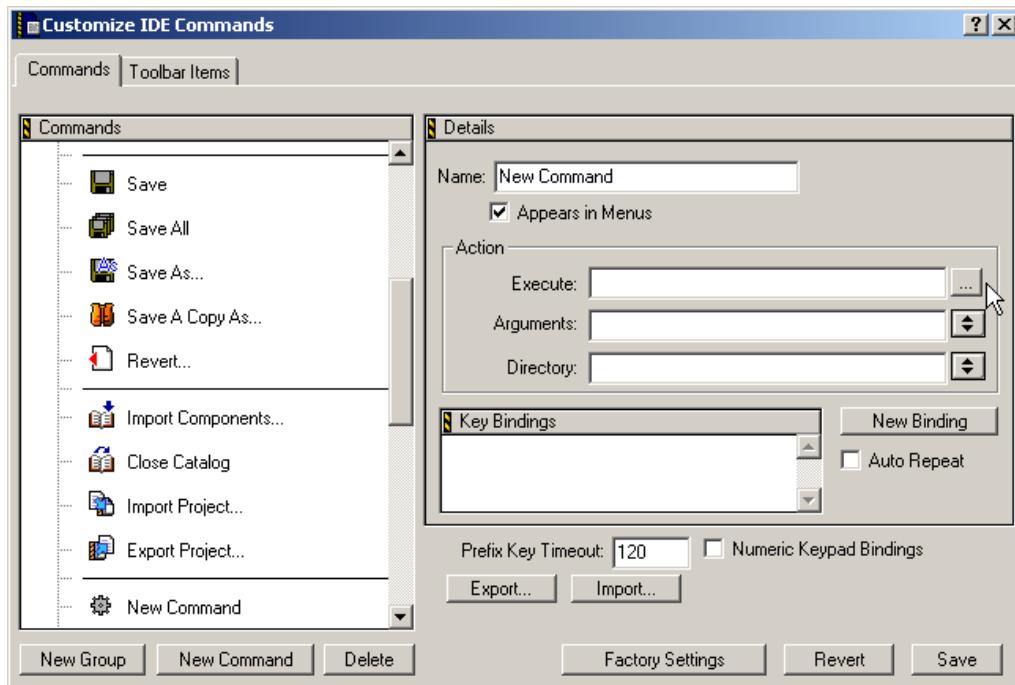
3. Enter a name for the new menu command.

You can change the default name of **New Command**. Enter a new name in the Name field of the Customize IDE Commands window.

4. Use the **Appears in Menus** checkbox to toggle the appearance of the new command within its command group.
5. Define the desired Action for the new menu command.
6. Click **Save**.

The IDE saves your new menu command. If you enabled the **Appears in Menus** checkbox, the new menu command appears within the selected command group.

**Figure 30.3 Command action fields**



---

## Defining Command Actions (Windows)

These fields help you associate an action with the new menu command:

- **Execute**—Enter in this field a command to run an application. Alternatively, click the ellipsis button next to the field to select the application using a standard dialog box.

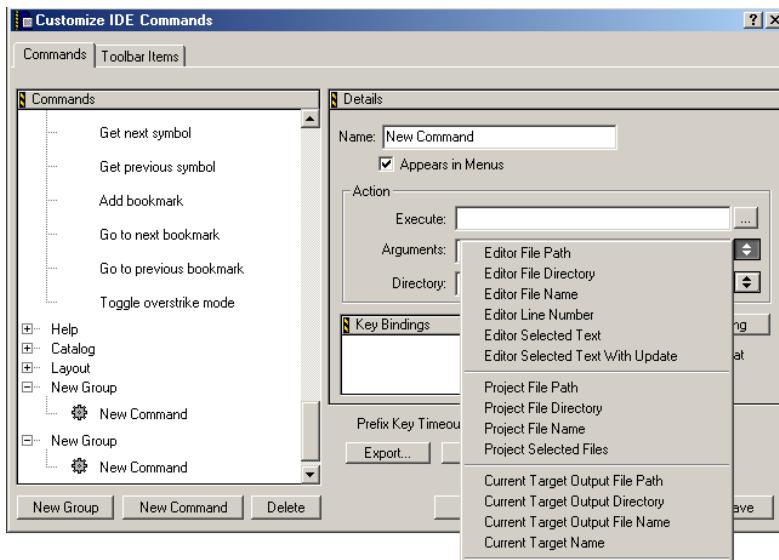
- **Arguments**—Enter in this field the arguments that the IDE must pass to the application specified in the Execute field. Alternatively, choose the desired arguments from the pop-up menu next to the field.
- **Directory**—Enter in this field the working directory the IDE should use when it executes the application specified in the Execute field. Alternatively, choose the desired directory from the pop-up menu next to the field.

## Pre-defined Variables in Command Definitions

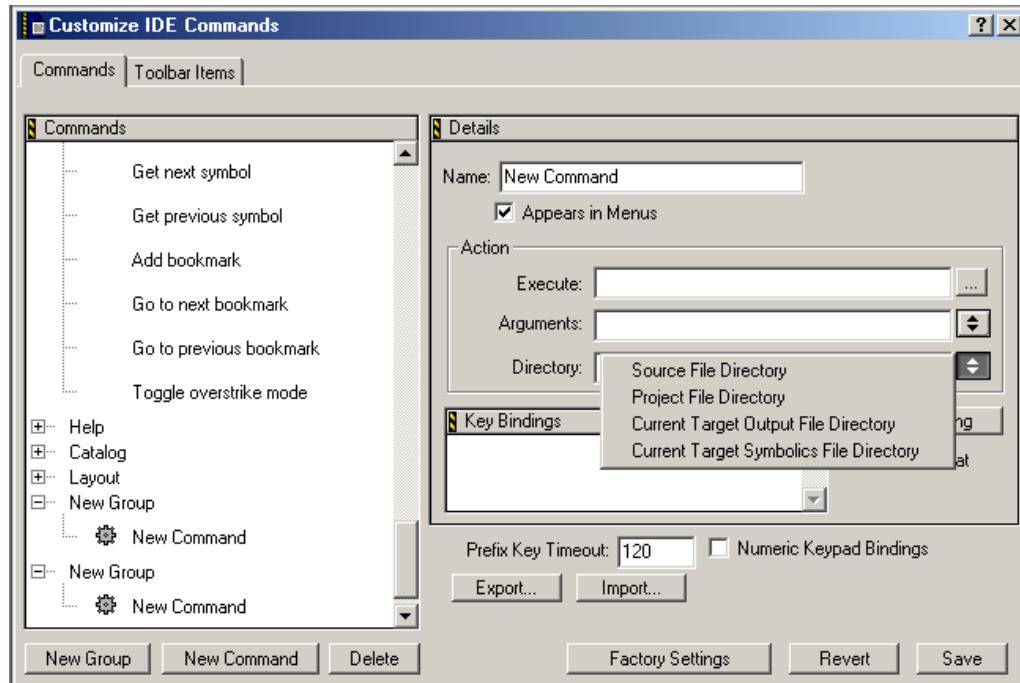
The IDE provides pre-defined variables for Windows, Solaris, and Linux (not Mac OS) to associate actions with commands. When you create a new command, you can use these pre-defined variables in command definitions to provide additional arguments that the IDE passes to the application (which is specified in the Execute field).

**NOTE** You can use variables that end with `Dir` as both argument and directory names.

Figure 30.4 Pre-defined Arguments variables



**Figure 30.5 Pre-defined Directory variables**



[Table 30.2](#) explains the pre-defined variables for command-line arguments.

**Table 30.2 Pre-defined variables in command definitions**

Variable	Command-line output
%sourceFilePath	<b>sourceFilePath</b> is the frontmost editor window's full path.
%sourceFileDir	<b>sourceFileDir</b> is the frontmost editor window's directory.
%sourceFileName	<b>sourceFileName</b> is the frontmost editor window's filename.
%sourceLineNumber	<b>sourceLineNumber</b> is the line number of the insertion point in the front window.
%sourceSelection	<b>sourceSelection</b> is the path to a temporary file containing the currently selected text.
%sourceSelUpdate	<b>sourceSelUpdate</b> is like <b>sourceSelection</b> , except the IDE waits for the command to finish and updates the selected text with the contents of the file.

**Table 30.2 Pre-defined variables in command definitions (*continued*)**

Variable	Command-line output
%projectFilePath	<b>projectFilePath</b> is the full path of the front project window.
%projectFileDir	<b>projectFileDir</b> is the directory of the front project window.
%projectFileName	<b>projectFileName</b> is the filename of the front project window.
%projectSelectedFiles	<b>%projectSelectedFiles</b> passes the selected filenames in the project window.
%targetFilePath	<b>targetFilePath</b> is the full path of the output file of the front project.
%targetFileDir	<b>targetFileDir</b> is the directory of the output file of the front project.
%targetFileName	<b>targetFileName</b> is the filename of the output file of the front project.
%currentTargetName	<b>%currentTargetName</b> passes the name of the current target of the frontmost window.
%symFilePath	<b>symFilePath</b> is the full path to the symbolics file of the front project (can be the same as targetFile, such as CodeView).
%symFileDir	<b>symFileDir</b> is the full directory to the symbolics file of the front project (can be the same as targetFile, such as CodeView)
%symFileName	<b>symFileName</b> is the full filename to the symbolics file of the front project (can be the same as targetFile, such as CodeView)

---

## Using a Pre-defined Variable

To use a pre-defined variable, follow these steps:

1. Create a new menu command.

The IDE creates a new menu command named **New Command** and places it within your selected command group. The information for the new menu command appears in the Customize IDE Commands window.

2. Enter a name for the new menu command.

You can change the default name of **New Command**. Enter a new name in the Name field of the Customize IDE Commands window.

3. Use the **Appears in Menus** checkbox to toggle the appearance of the new command within its command group.
4. Define the Action for the new menu command.
  - a. Enter in the **Execute** field a command line to run an application.
  - b. Next to the **Arguments** field, click on the arrow icon and select an argument listed in the pop-up menu.
  - c. Next to the **Directory** field, click on the arrow icon and select a directory listed in the pop-up menu.
5. Click **Save**.

The IDE saves your new menu command with the pre-defined variables. If you enabled the **Appears in Menus** checkbox, the new menu command appears within the selected command group.

---

## Defining Command Actions (Mac OS)

After you create a new menu command the **Customize IDE Commands** window shows the **Run App/Script** field. This field appears in the Action section of the window.

1. Click the **Choose** button next to the field to display a standard Open dialog box.
2. Use the dialog box to select an application or script.

The IDE launches this application or script each time you choose the menu command. The path to the selected application or script appears in the **Run App/Script** field.

---

## Deleting Command Groups and Menu Commands

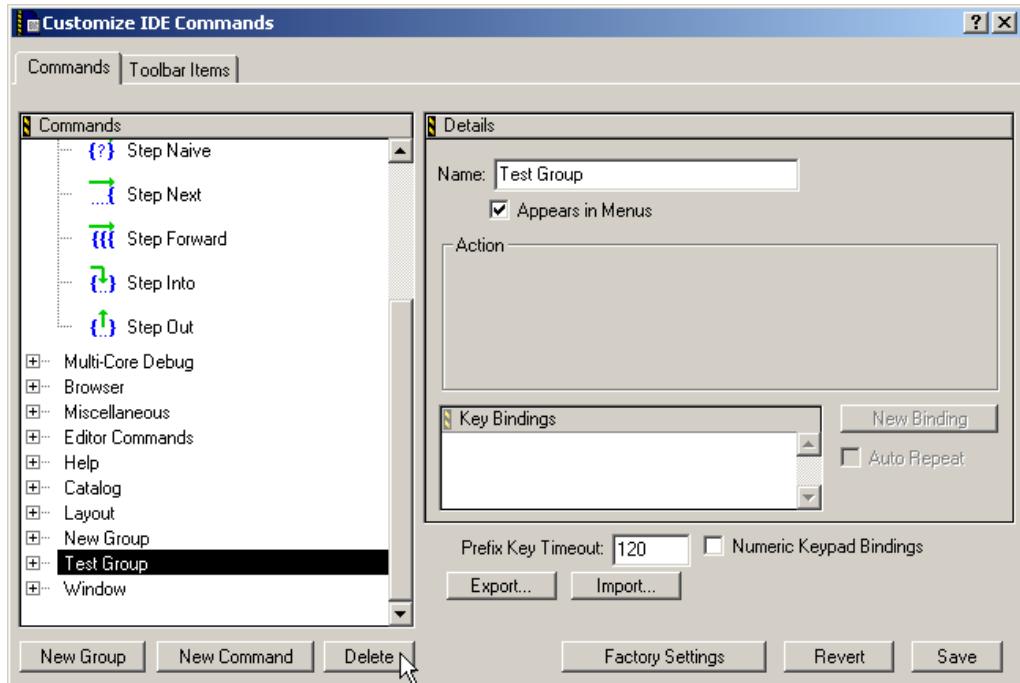
You can delete the command groups and menu commands that you create for the IDE. Once removed, the command groups no longer appear in the IDE's menu bar, and the menu commands no longer activate their associated command lines (Windows) or applications or scripts (Mac OS).

---

**NOTE** If you need to temporarily remove your customized command groups and menu commands, consider exporting your settings. If you export your settings, you do not need to reconstruct them if you want them in the future.

---

**Figure 30.6 Delete Command group**



To delete a command group or menu command, follow these steps:

1. Select the command group or menu command you wish to delete.

If necessary, click the hierarchical control next to a group to expand and view its contents.

2. Click **Delete**.

After clicking the **Delete** button, the selected command group or menu command disappears from the Commands list.

3. Click **Save**.

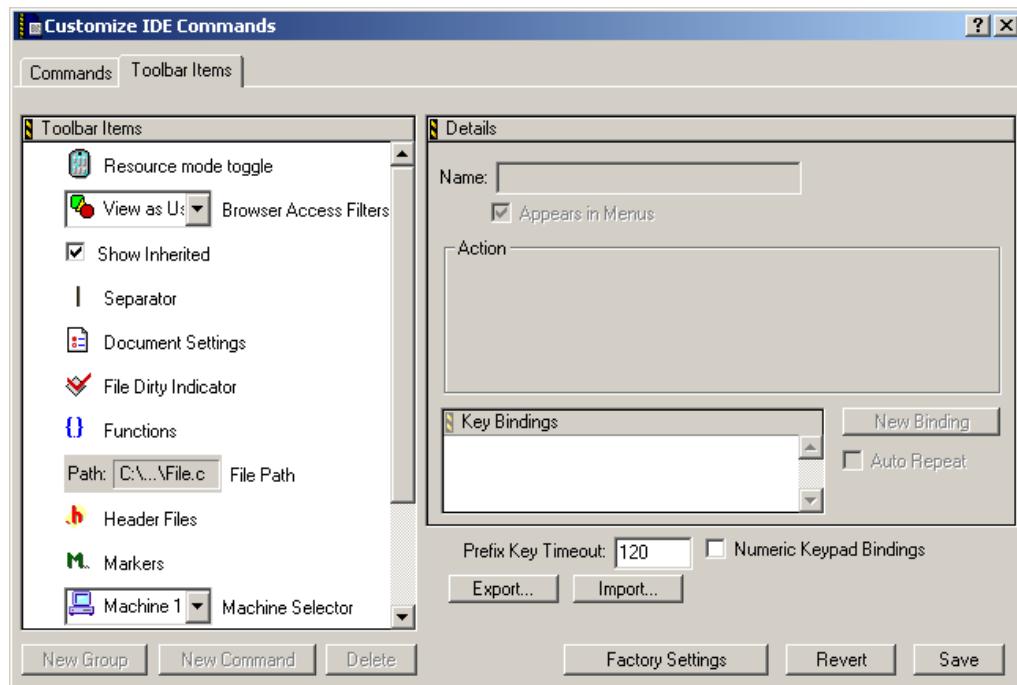
Clicking the **Save** button confirms the deletion. The IDE removes deleted command groups from its menu bar. Deleted menu commands disappear from their respective command groups.

## Customize Toolbars

You can customize your IDE toolbars to contain frequently used commands.

The IDE toolbars contain a series of elements. Each element typically represents a menu command. After you click the element, the IDE executes the associated menu command. The toolbar can also contain elements that execute actions other than menu commands.

**Figure 30.7 Toolbar Items tab**



This section explains these topics:

- [“Kinds of Toolbars” on page 339](#)
- [“Toolbar Elements” on page 340](#)
- [“Modify a Toolbar” on page 340](#)

## Kinds of Toolbars

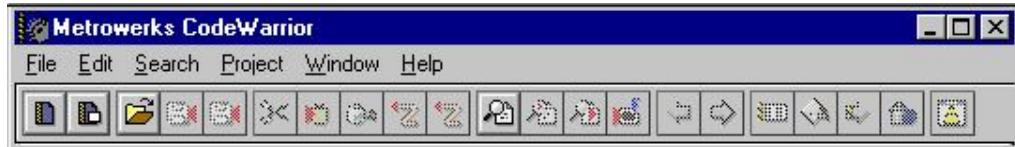
The CodeWarrior IDE uses two toolbar types:

- Main toolbar—This toolbar, also known as the floating toolbar, is always available.
- Window toolbars—These toolbars appear in particular windows, such as the Project window toolbar and the Browser window toolbar.

This distinction is important because you show, hide, clear, and reset the different toolbar types by using different sets of menu commands. These commands distinguish between the floating toolbar and the other window toolbars.

When you change one of these toolbar types, that change applies to every instance of that toolbar type you subsequently create. For example, if you modify the toolbar in an editor window, your changes appear in all editor windows opened thereafter.

**Figure 30.8 Main toolbar**



**Figure 30.9 Project window toolbar**



## Toolbar Elements

A toolbar can contain these elements:

- *Commands*—buttons that you click to execute IDE menu commands
- *Controls*—menus, such as Document Settings, Functions, Header Files, Markers, Version Control, and Current Target
- *Miscellaneous*—other elements, such as the File Dirty Indicator and File Path field
- *Scripts* (Mac OS)—buttons that you click to execute one of the scripts available through the Scripts menu

Click the **Toolbar Items** tab at the top of the Customize IDE Commands window to display the Toolbar view. Use this view to add new elements to a toolbar.

## Modify a Toolbar

You can modify a toolbar in these ways:

- Add a toolbar element
- Remove a toolbar element
- Clear all elements on a toolbar
- Reset a toolbar

In certain circumstances there are restrictions on which elements you can add or remove from a toolbar. For example, you cannot add a second instance of an element to the toolbar.

After you modify a toolbar, the changes apply to every instance of that toolbar. For example, if you customize the Project window toolbar, those changes will affect every Project window that you open, not just the toolbar in the active Project window. Your changes do not affect windows that are already open.

---

<b>TIP</b>	To display a ToolTip that names a toolbar element, rest the cursor over the element. On the Mac OS 9-hosted IDE, activate Balloon Help and rest the cursor over the element.
------------	--

---

## Adding a Toolbar Element

You add an element to a toolbar by dragging and dropping it from the Toolbar Items list onto a toolbar. This list is in the **Toolbar Items** view in the Customize IDE Commands window.

To add an element to a toolbar, follow these steps:

1. From the **Toolbar Items** list, select the *icon* next to the element that you want to add to a toolbar.

Make sure that the destination toolbar is visible.

2. Drag the element's icon from the Toolbar Items list to the destination toolbar.

On the Windows-hosted IDE, if the destination toolbar accepts the element, a framing bracket appears in the toolbar. This framing bracket shows you where the new element will appear after you release the cursor. If the destination toolbar does not accept the element, the framing bracket does not appear.

3. Release the element at the desired position.

After you release the element, the IDE inserts the element into the destination toolbar.

The toolbar might not accept an element for these reasons:

- The toolbar is full.
- The element already exists in the toolbar.
- You cannot use the element in the toolbar because the window does not support that element.
- You cannot add these elements to any toolbar except the editor window toolbar: **Document Settings**, **Functions**, **Header Files**, **Markers**, and **Version Control** menus; **File Dirty Indicator**; and **File Path** field.
- You cannot add the **Current Target** menu element to any toolbar except the Project window toolbar.

## Removing a Toolbar element

You remove a toolbar element by selecting it and using a contextual menu command to remove it from the toolbar.

To remove an element from a toolbar, follow these steps:

1. Display a contextual menu for the button that you want to remove, as explained in [Table 30.3 on page 342](#).

**Table 30.3 Displaying a contextual menu for a toolbar button**

On this host...	Do this...
Windows	Right-click the button.
Macintosh	Control-click the button.
Solaris	Control-click the button.
Linux	Ctrl-click the button.

2. Select the **Remove Toolbar Item** command from the contextual menu.

The IDE removes the button from the toolbar.

---

## Clearing All Buttons on Toolbars

You can clear all elements from a toolbar and build your own toolbar from scratch. [Table 30.4](#) explains how to clear the main (floating) toolbar and window toolbars.

**Table 30.4 Clearing toolbars**

On this host...	Do this to clear the main toolbar...	Do this to clear the window toolbar...
Windows	Select <b>View &gt; Toolbars &gt; Clear Main Toolbar</b> .	Select <b>View &gt; Toolbars &gt; Clear Window Toolbar</b> .
Macintosh	Select <b>Window &gt; Toolbar &gt; Clear Floating Toolbar</b> .	Select <b>Window &gt; Toolbar &gt; Clear Window Toolbar</b> .

**Table 30.4 Clearing toolbars (*continued*)**

On this host...	Do this to clear the main toolbar...	Do this to clear the window toolbar...
Solaris	Select Window > Toolbar > Clear Floating Toolbar.	Select Window > Toolbar > Clear Window Toolbar.
Linux	Select Window > Toolbar > Clear Floating Toolbar.	Select Window > Toolbar > Clear Window Toolbar.

## Reset Toolbars

Reset a toolbar to restore its default button set. explains how to reset the main (floating) toolbar and window toolbar by using menu commands.

**Table 30.5 Resetting a toolbar by using menu commands**

On this host...	Do this to reset the main toolbar...	Do this to reset the window toolbar...
Windows	Select View > Toolbars > Reset Main Toolbar.	Select View > Toolbars > Reset Window Toolbar.
Macintosh	Select Window > Toolbar > Reset Floating Toolbar.	Select Window > Toolbar > Reset Window Toolbar.
Solaris	Select Window > Toolbar > Reset Floating Toolbar.	Select Window > Toolbar > Reset Window Toolbar.
Linux	Select Window > Toolbar > Reset Floating Toolbar.	Select Window > Toolbar > Reset Window Toolbar.

Alternatively, you can use a contextual menu to reset the main toolbar or a window toolbar. Once you reset the toolbar, the IDE restores the default toolbar button set. explains how to reset the main (floating) toolbar and window toolbar by using a contextual menu.

**Table 30.6 Resetting a toolbar by using a contextual menu**

<b>On this host...</b>	<b>Do this to reset the main toolbar...</b>	<b>Do this to reset the window toolbar...</b>
Windows	Right-click the toolbar and select <b>Reset Toolbar</b> .	Right-click the toolbar and select <b>Reset Toolbar</b> .
Macintosh	Control-click the toolbar and select <b>Reset Toolbar</b> .	Control-click the toolbar and select <b>Reset Toolbar</b> .
Solaris	Click and hold on the toolbar, then select <b>Reset Toolbar</b> .	Click and hold on the toolbar, then select <b>Reset Toolbar</b> .
Linux	Click and hold on the toolbar, then select <b>Reset Toolbar</b> .	Click and hold on the toolbar, then select <b>Reset Toolbar</b> .

## Customize Key Bindings

You can customize the keyboard shortcuts, known as key bindings, for various commands in the CodeWarrior IDE. You can bind a set of keystrokes to virtually any command. To activate the command, type its associated key binding. Use the Customize IDE Commands window to change IDE key bindings.

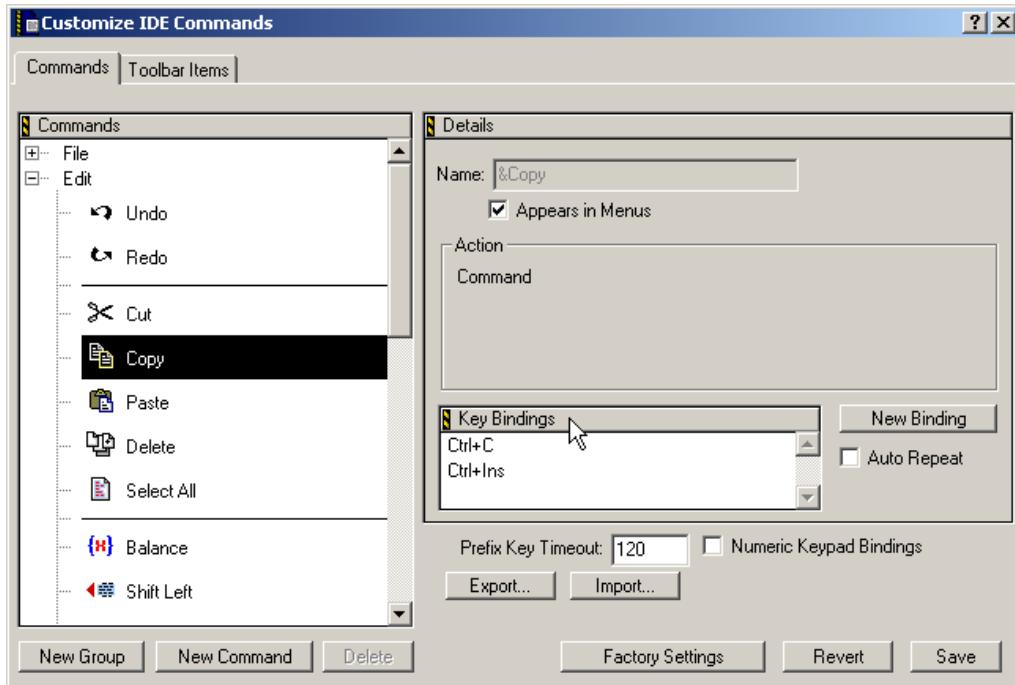
You can also use the Customize IDE Commands window to look up default key bindings for specific commands, as well as change existing key bindings to better suit your needs.

Click the **Commands** tab at the top of the Customize IDE Commands window to display the Commands view. Use this view to configure the key bindings for menu commands, editor actions, and other actions. You can also specify prefix keys.

This section has these topics:

- Modifying key bindings
- Adding key bindings
- Deleting key bindings
- Setting Auto Repeat for key bindings
- Exporting commands and key bindings
- Importing commands and key bindings

Figure 30.10 Customize IDE Commands—Key Bindings



## Adding Key Bindings

Use the Customize IDE Commands window to specify additional key bindings for a particular command.

To add a key binding, follow these steps:

1. From the Commands list, select the command to which you want to add a new key binding.

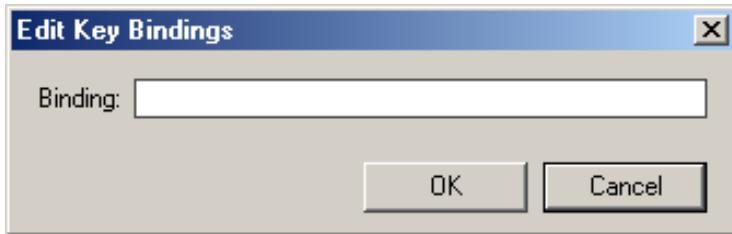
Click the hierarchical controls next to the command categories to expand them as necessary so that you can see individual commands. Select the individual command you wish to modify.

<b>NOTE</b>	If you want to use your keyboard's numeric keypad as part of the new key binding, enable the <b>Numeric Keypad Bindings</b> checkbox in the Customize IDE Commands window.
-------------	--

2. Click **New Binding**.

After clicking this button, the **Edit Key Binding** dialog box appears.

**Figure 30.11 Edit Key Bindings**



3. Create the key combination you would like to use for the selected command.

For example, to add the key combination Ctrl-8, hold down the **Ctrl** key and press the **8** key, then release both keys at the same time.

If you decide against the key combination that you just entered, or if you make a mistake, click **Cancel** in the **Edit Key Binding** dialog box. The IDE discards changes and returns you to the Customize IDE Commands window.

4. Click **OK** in the Edit Key Binding dialog box.

The new key binding appears in the Key Bindings list in the Customize IDE Commands window.

5. Click **Save** in the Customize IDE Commands window to save your changes.

The new key binding is now available for use in the IDE.

## Exporting Commands and Key Bindings

You can export commands and key bindings for later use if you do not want to delete them.

To export a command or key binding, follow these steps:

1. Click **Export** in the Customize IDE Commands window.

After you click this button, a standard **Save** dialog box appears.

2. Select a location in a computer or network folder to save the **Cmds&Bindings** file.
3. Click **Save**.

The IDE exports the current commands and key bindings to a file at that location.

---

**TIP**

Complete the name of the **Cmds&Bindings** file with a **.mkb** file-name extension like this: **MyCmdsAndKBS.mkb**. This naming convention helps you quickly identify the file as a Metrowerks **Key Bindings** file. Furthermore, the Windows-hosted version of the CodeWarrior IDE uses this extension to properly recognize the commands and key bindings file.

---

## Importing Commands and Key Bindings

You can import commands and key bindings from a previously exported file.

To import a command or key binding, follow these steps:

1. Click **Import** in the Customize IDE Commands window.

After you click this button, a standard **Open** dialog box appears.

2. Use the dialog box to find and open the **Cmds&Bindings** file you want to import.

The IDE adds the imported commands and key bindings to the **Commands** list in the Customize IDE Commands window.



# Working with IDE Preferences

---

This chapter explains core CodeWarrior™ IDE preference panels and provides basic information on global- and project-level preference options. Consult the *Targeting* documentation for information on platform-specific preference panels.

This chapter contains these sections:

- [“IDE Preferences Window” on page 349](#)
- [“General Panels” on page 351](#)
- [“Editor Panels” on page 365](#)
- [“Debugger Panels” on page 375](#)
- [“RAD Tools Panels” on page 383](#)

Abbreviated descriptions appear in this chapter. See [“Preference and Target Settings Options” on page 411](#) for more information on preference-panel options.

## IDE Preferences Window

The **IDE Preferences** window lists global IDE preference options. These preferences, unless superseded by a Target Settings option, apply to every open project file.

The IDE Preferences window lists preferences by group:

- **General**—configures overall IDE preferences, such as project builds, recent items, and third-party tools
- **Editor**—configures editor preferences, such as fonts, tabs, and syntax coloring
- **Debugger**—configures debugger preferences, such as window hiding during debugging sessions, low-level interactions, and variable highlighting
- **RAD Tools**—configures Rapid Application Development preferences, such as Layout Editor operation

## Working with IDE Preferences

### IDE Preferences Window

Figure 31.1 IDE Preferences window

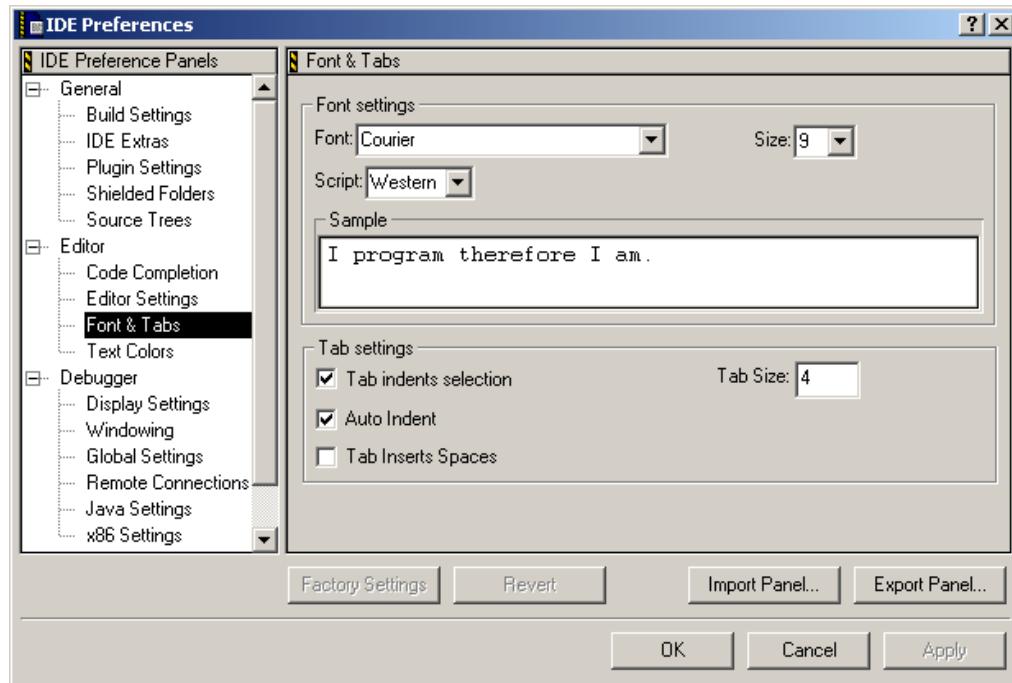


Table 31.1 IDE Preferences window—items

Item	Explanation
IDE Preference Panels list	Lists preference panels, organized by group. Click the hierarchical control next to a group name to show or hide individual preference panels.
Preference panel	Shows the options for the selected item in the <b>IDE Preference Panels</b> list.
<a href="#">Factory Settings</a>	Click to restore the default options for the current preference panel.
<a href="#">Revert Panel</a>	Click to restore the most recently saved options for the current preference panel.
<a href="#">Export Panel</a>	Click to save an XML file that contains options for the current preference panel.
<a href="#">Import Panel</a>	Click to open an XML file that contains options for the current preference panel.

**Table 31.1 IDE Preferences window—items (*continued*)**

Item	Explanation
OK (Windows)	Click to save modifications to all preference panels and close the window.
Cancel (Windows)	Click to discard modifications to all preference panels and close the window.
Apply (Windows)	Click to confirm modifications to all preference panels.
Save (Macintosh, Solaris, and Linux)	Click to save modifications to all preference panels.

---

## Opening the IDE Preferences Window

Use the **IDE Preferences** window to modify global general, editor, debugger, and RAD options.

To open the IDE Preferences window, select **Edit > Preferences**.

# General Panels

The **General** section of the IDE Preference Panels list defines the basic options assigned to a new project.

The General preference panels available on most IDE hosts include:

- [“Build Settings” on page 351](#)
- [“Concurrent Compiles” on page 353](#)
- [“IDE Extras” on page 354](#)
- [“Help Preferences” on page 358](#)
- [“Plugin Settings” on page 358](#)
- [“Shielded Folders” on page 359](#)
- [“Source Trees” on page 361](#)

## Build Settings

The **Build Settings** preference panel provides options for customizing various aspects of project builds, including:

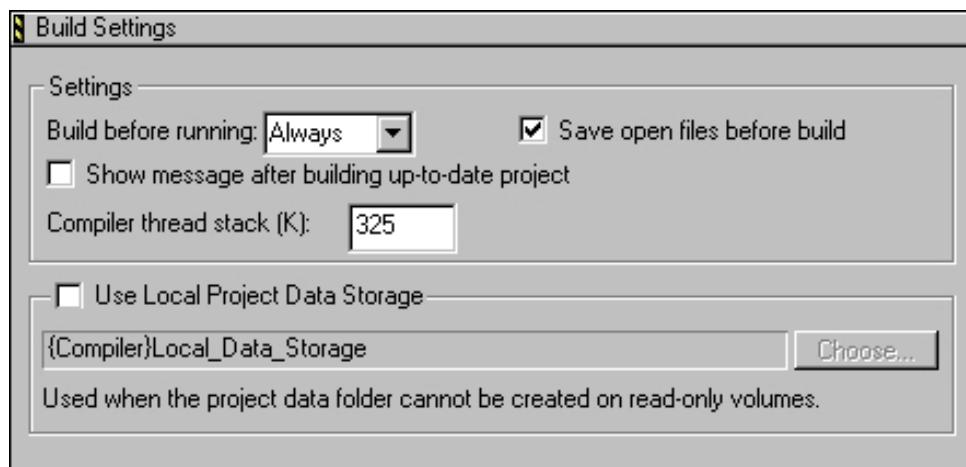
## Working with IDE Preferences

### General Panels

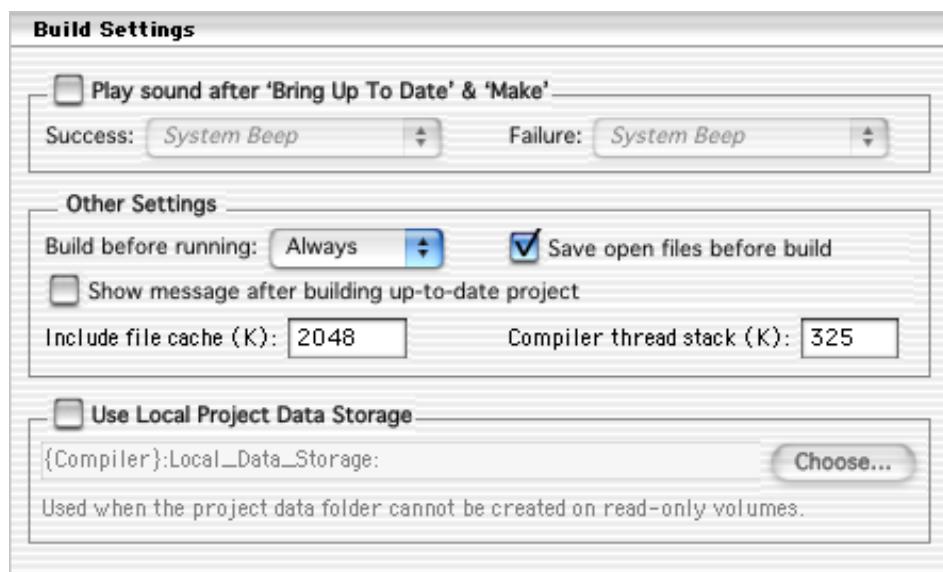
---

- file actions during project builds
- memory usage to accelerate builds
- local data storage of projects stored on read-only volumes

**Figure 31.2 Build Settings preference panel (Windows)**



**Figure 31.3 Build Settings preference panel (Macintosh)**



**Table 31.2 Build Settings preference panel—items**

Item	Explanation
<a href="#">Play sound after 'Bring Up To Date' &amp; 'Make'</a> (Macintosh, Solaris, and Linux)	Select to have the IDE play an alert sound after completing a <b>Bring Up To Date</b> or <b>Make</b> command.
<a href="#">Success</a> (Macintosh, Solaris, and Linux)	Choose the sound the IDE plays after successfully completing a <b>Bring Up To Date</b> or <b>Make</b> command.
<a href="#">Failure</a> (Macintosh, Solaris, and Linux)	Choose the sound the IDE plays after failing to complete a <b>Bring Up To Date</b> or <b>Make</b> command.
<a href="#">Build before running</a>	Choose to always build the project before running it, never build the project before running it, or ask for the desired action.
<a href="#">Save open files before build</a>	Select to automatically save the contents of all editor windows before starting a build.
<a href="#">Show message after building up-to-date project</a>	Select to have the IDE display a message after successfully building a project.
<a href="#">Include file cache</a> (Macintosh)	Enter the kilobytes of memory to allocate to the file cache used for #include files during a project build.
<a href="#">Compiler thread stack</a> (Windows and Macintosh)	Enter the kilobytes of memory to allocate to the stack for execution of the IDE compiler thread. Increase the size when compiling heavily optimized code.
<a href="#">Use Local Project Data Storage</a>	Select to specify a location to save project data if the project is on a read-only volume. Click <b>Choose</b> to select the location.

## Concurrent Compiles

The **Concurrent Compiles** preference panel controls execution of simultaneous IDE compilation processes. The IDE lists this panel in the IDE Preference Panels list when the active compiler supports concurrency.

The IDE uses concurrent compiles to compile code more efficiently. The IDE improves its use of available processor capacity by spawning multiple compile processes, which allow the operating system to perform these tasks as needed:

- optimize resource use
- use overlapped input/output

For those compilers that support concurrency, concurrent compiles improve compile time on both single- and multiple-processor systems.

Figure 31.4 Concurrent Compiles preference panel

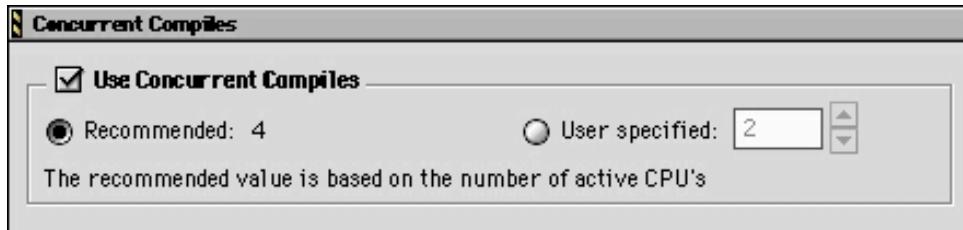


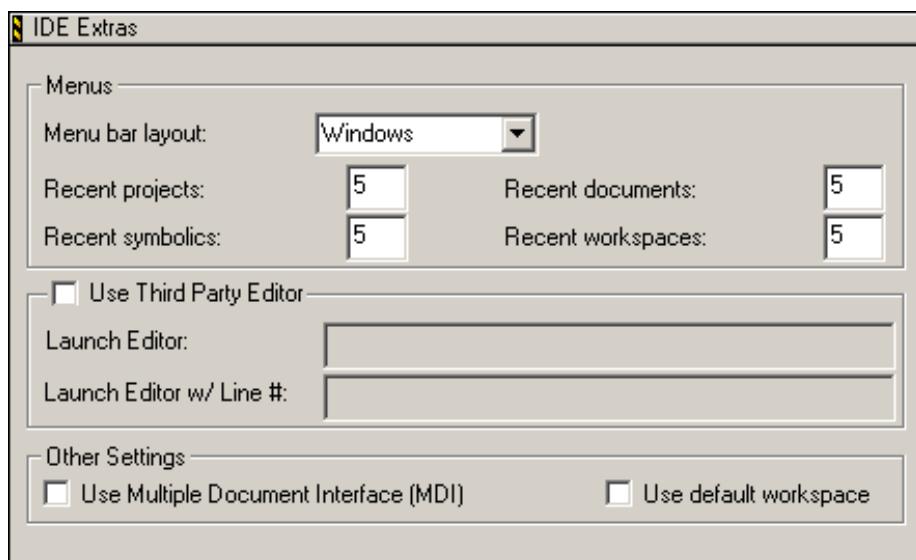
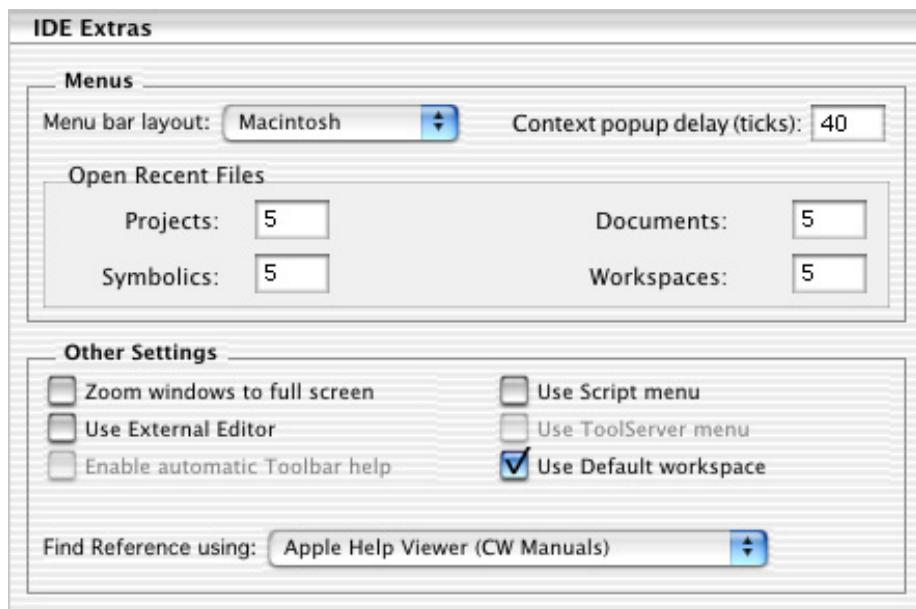
Table 31.3 Concurrent Compiles preference panel—items

Item	Explanation
<a href="#">Use Concurrent Compiles</a>	Select to have the IDE run multiple compilation processes simultaneously.
<a href="#">Recommended</a>	Select to allow the number of concurrent compiles suggested by the IDE.
<a href="#">User Specified</a>	Select to stipulate the number of concurrent compiles.

## IDE Extras

The **IDE Extras** preference panel provides options for customizing various aspects of the IDE, including:

- menu-bar layout
- the number of recent projects, document files, and symbolics files to remember
- use of a third-party editor

**Figure 31.5 IDE Extras preference panel (Windows)****Figure 31.6 IDE Extras preference panel (Macintosh)**

## Working with IDE Preferences

### General Panels

---

**Table 31.4 IDE Extras preference panel—items**

Item	Explanation
<a href="#">Menu bar layout</a>	Choose a layout that organizes IDE menus into a typical host-platform menu bar. Restart the IDE in order for menu-bar layout changes to take effect.
<a href="#">Projects</a>	Enter the number of recently opened projects for the IDE to display in the <b>Open Recent</b> submenu. Enter zero to disable this feature.
<a href="#">Documents</a>	Enter the number of recently opened documents for the IDE to display in the <b>Open Recent</b> submenu. Enter zero to disable this feature.
<a href="#">Symbolics</a>	Enter the number of recently opened symbolics files for the IDE to display in the <b>Open Recent</b> submenu. Enter zero to disable this feature.
<a href="#">Workspaces</a>	Enter the number of recently opened workspaces for the IDE to display in the <b>Open Recent</b> submenu. Enter zero to disable this feature.
<a href="#">Context popup delay</a> (Macintosh, Solaris, and Linux)	Enter the number of ticks to wait before displaying contextual menus. A tick is 1/60 of a second.
<a href="#">Use Third Party Editor</a> (Windows)	Select to use a third-party text editor to edit source files.
<a href="#">Launch Editor</a> (Windows)	Enter a command-line expression that runs the desired third-party text editor.
<a href="#">Launch Editor w/ Line #</a> (Windows)	Enter a command-line expression that runs the desired third-party text editor and passes to that editor an initial line of text to display.
<a href="#">Use Multiple Document Interface</a> (Windows)	Select to have the IDE use the Multiple Document Interface (MDI). Clear to have the IDE use the Floating Document Interface (FDI). Restart the IDE in order for interface changes to take effect.
<a href="#">Zoom windows to full screen</a> (Macintosh, Solaris, and Linux)	Select to have zoomed windows fill the entire screen. Clear to have zoomed windows in a default size.
<a href="#">Use Script menu</a> (Macintosh, Solaris, and Linux)	Select to display the Scripts menu in the menu bar. Clear to remove the Scripts menu from the menu bar.
<a href="#">Use External Editor</a> (Macintosh, Solaris, and Linux)	Select to use a third-party text editor to edit text files in the current project. Clear to use the editor included with the IDE.
<a href="#">Use ToolServer menu</a> (Classic Macintosh)	Select to display the ToolServer menu in the menu bar. Clear to remove the ToolServer menu from the menu bar.

**Table 31.4 IDE Extras preference panel—items (*continued*)**

Item	Explanation
<a href="#">Enable automatic Toolbar help (Classic Macintosh)</a>	Select to display Balloon Help after resting the cursor over a toolbar icon. Clear to prevent Balloon Help from appearing.
<a href="#">Use default workspace</a>	Select this option to have the IDE use the default workspace to save and restore state information. Clear this option to have the IDE always start in the same state.
<a href="#">Find Reference using (Macintosh)</a>	Choose an online browser application to view reference information and definitions.

---

## Using an External Editor on the Macintosh

To use an external editor on the Macintosh, the IDE sends AppleEvents to an alias file that points to the editor application. Manually configure the IDE to use an external editor.

1. Choose **Edit > Preferences**.

The IDE Preferences window appears.

2. Select the **IDE Extras** panel from the **IDE Preference Panels** list.
3. Select the **Use External Editor** option.
4. Click **Save**.

The IDE is now prepared to use an external editor application. To specify the external editor to use:

1. Find and open the **CodeWarrior** folder.
2. Create a folder named **(Helper Apps)** inside the **CodeWarrior** folder (if it does not already exist).
3. Make an alias of the desired editor application.
4. Place the alias file inside the **(Helper Apps)** folder.
5. Rename the alias file **External Editor**.

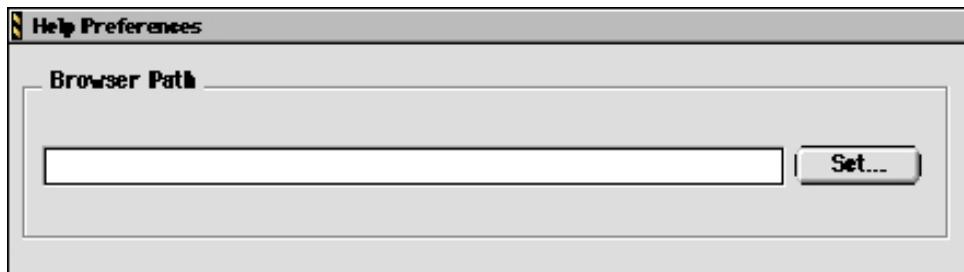
6. Restart the IDE in order for changes to take effect.

The IDE now uses the aliased external editor.

## Help Preferences

The **Help Preferences** panel, available on the Solaris and Linux IDE hosts, specifies the browser used for viewing IDE online help.

**Figure 31.7 Help Preferences panel**

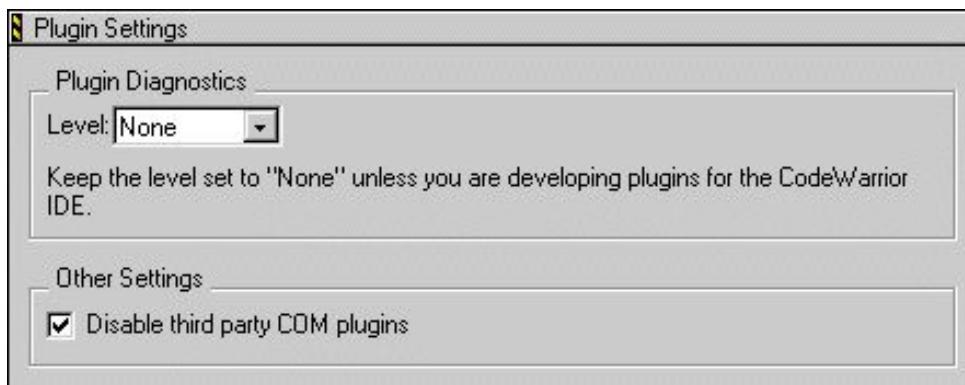


**Table 31.5 Help Preferences panel—items**

Item	Explanation
<a href="#">Browser Path</a>	Enter a path to the browser to use for viewing IDE online help. Alternatively, use the <b>Set</b> button.
Set	Click to select the path to the browser to use for viewing IDE online help.

## Plugin Settings

The **Plugin Settings** preference panel contains options for troubleshooting third-party IDE plug-ins.

**Figure 31.8 Plugin Settings preference panel****Table 31.6 Plugin Settings preference panel—items**

<b>Item</b>	<b>Explanation</b>
<a href="#"><u>Level</u></a>	Choose the plug-in diagnostics level the IDE generates the next time it loads plug-ins. Restart the IDE in order for diagnostic-level changes to take effect.
<a href="#"><u>Disable third party COM plugins</u></a>	Select to prevent the IDE from loading third-party Common Object Model (COM) plug-ins.

## Shielded Folders

The **Shielded Folder** preference panel enables the IDE to ignore specified folders during project operations and find-and-compare operations. The IDE ignores folders based on matching names with regular expressions defined in the preference panel.

---

<b>NOTE</b>	If the <b>Access Paths</b> settings panel in the Target Settings window contains a path to a shielded folder, the IDE overrides the shielding and includes the folder in project operations and find-and-compare operations.
-------------	--

---

Figure 31.9 Shielded Folders preference panel

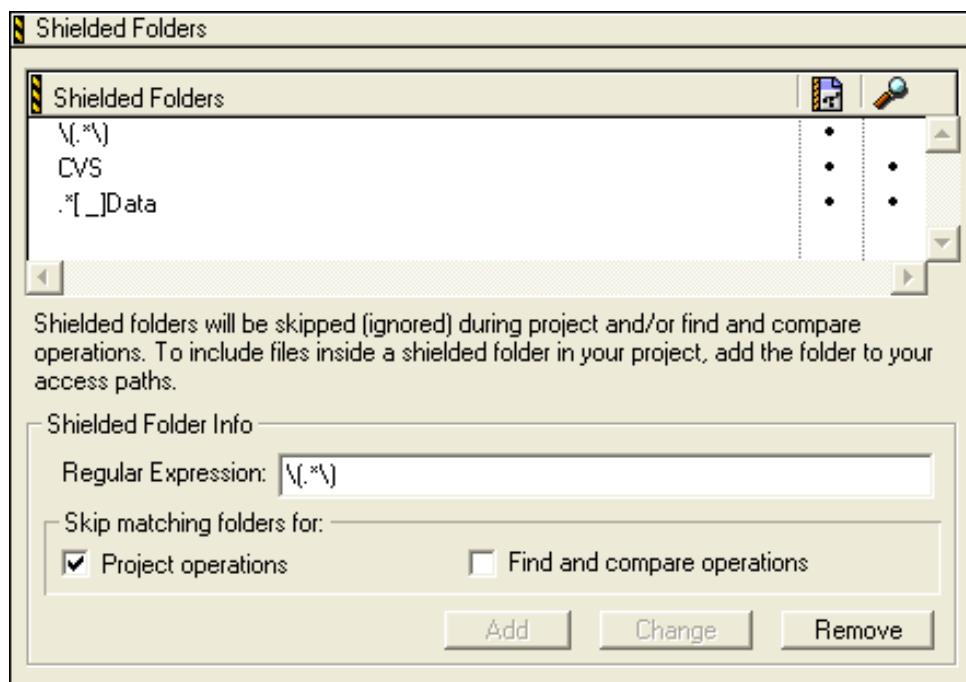


Table 31.7 Shielded Folders preference panel—items

Item	Icon	Explanation
Shielded folder list		Lists folders that match the specified regular expression. The IDE skips these folders during project operations, find-and-compare operations, or both operations.
<a href="#">Regular Expression</a>		Enter the regular expression used to shield folders from selected operations.
<a href="#">Project operations</a>		Select to have the IDE skip folders during project operations. A bullet appears in the corresponding column of the shielded folder list.
<a href="#">Find and compare operations</a>		Select to have the IDE skip folders during find-and-compare operations. A bullet appears in the corresponding column of the shielded folder list.
Add		Click to add the current <b>Regular Expression</b> field entry to the shielded folder list.

**Table 31.7 Shielded Folders preference panel—items (*continued*)**

Item	Icon	Explanation
Change		Click to replace the selected regular expression in the shielded folder list with the current <b>Regular Expression</b> field entry.
Remove		Click to delete the selected regular expression from the shielded folder list.

**Table 31.8 Default regular expressions in Shielded Folders panel**

Regular Expression	Explanation
\(.*\)	Matches folders with names that begin and end with parentheses, such as the (Project Stationery) folder.
CVS	Matches folders named CVS. With this regular expression, the IDE skips Concurrent Versions System (CVS) data files.
. * [__]Data	Matches the names of folders generated by the IDE that store target data information, such as a folder named MyProject_Data.

## Source Trees

Use the **Source Trees** panel to add, modify, and remove source trees (root paths) used in projects. Use source trees to define common access paths and build-target outputs to promote sharing of projects across different hosts. Source trees have these scopes:

- *Global source trees*, defined in the IDE Preferences window, apply to all projects.
- *Project source trees*, defined in the Target Settings window for a particular project, apply only to files in that project. Project source trees always take precedence over global source trees.

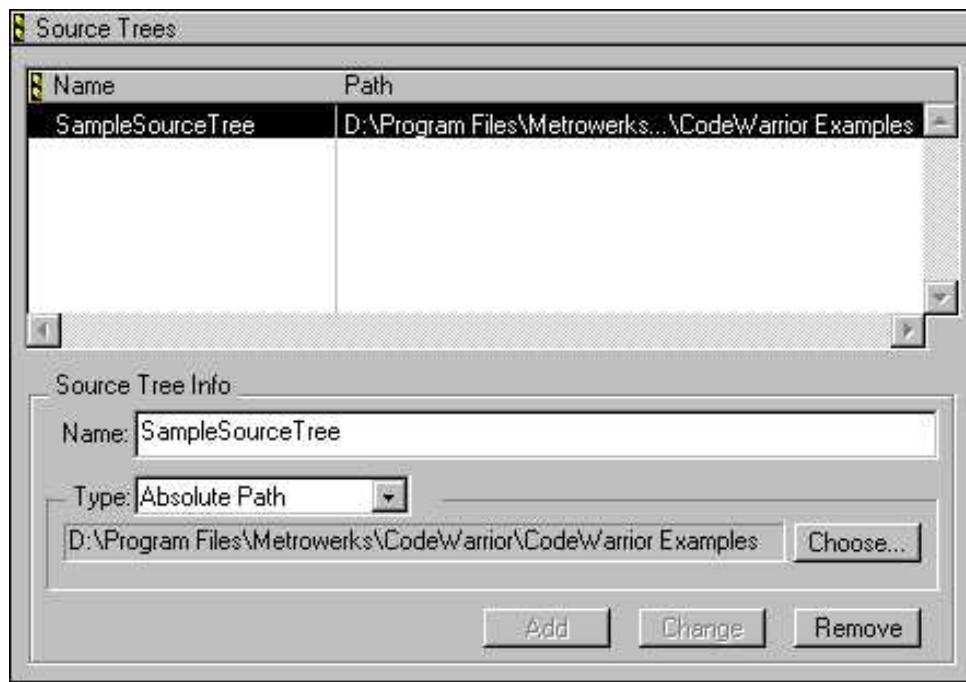
Except for the difference in scope, global and project source trees operate identically.

## Working with IDE Preferences

### General Panels

---

**Figure 31.10 Source Trees panel**



**Table 31.9 Source Trees panel—items**

Item	Explanation
Source Tree list	Contains the <b>Name</b> and <b>Path</b> of currently defined source trees.
Name	Enter in this field a name for a new source tree or modify the name of a selected source tree.
Type	Choose the source-tree path type.
Choose	Click to select or modify a source-tree path.
Add	Click to add a new source-tree path to the Source Tree list.
Change	Click to modify the selected source-tree name or path.
Remove	Click to delete the selected source tree from the Source Tree list.

## Adding Source Trees

Add source trees that define root paths for access paths and build-target output.

1. Choose **Edit > Preferences**.

The IDE Preferences window appears.

2. Select the **Source Trees** panel from the **IDE Preference Panels** list.

3. Enter in the **Name** field a name for the new source tree.

4. Choose the source tree **Type**:

- **Absolute Path**—defines a path from the root level of the hard drive to a desired folder, including all intermediate folders
- **Environment Variable**—(Windows, Solaris, and Linux) defines an environment variable in the operating environment
- **Registry Key**—(Windows) defines a key entry in the operating-environment registry

5. Enter the source-tree definition:

- For **Absolute Path**—Click **Choose** to display a subordinate dialog box. Use the dialog box to select the desired folder. The absolute path to the selected folder appears in the **Source Trees** preference panel.
- For **Environment Variable**—Enter the path to the desired environment variable.
- For **Registry Key**—Enter the path to the desired key entry in the registry.

6. Click **Add**.

The IDE adds the new source tree to the **Source Trees** list.

7. Click **OK**, **Apply**, or **Save**.

The IDE saves the source-tree changes.

## Changing Source Trees

Change a source tree to update path information for a project. The IDE must be able to resolve source trees before building the project.

1. Choose **Edit > Preferences**.

The IDE Preferences window appears.

2. Select the **Source Trees** panel from the **IDE Preference Panels** list.

3. Select the desired source tree in the **Source Trees** list.

4. If needed, enter in the **Name** field a new name for the selected source tree.

5. If needed, choose from the **Type** options a new path type for the selected source tree.

6. Click **Change**.

The IDE updates the source tree and displays changes in the **Source Trees** list. A reminder message to update source-tree references in the project appears.

7. Click **OK**, **Apply**, or **Save**.

The IDE saves the source-tree changes.

---

## Removing Source Trees

Remove source trees that the project no longer uses. The IDE must be able to find the remaining source trees before building the project.

1. Choose **Edit > Preferences**.

The IDE Preferences window appears.

2. Select the **Source Trees** panel from the **IDE Preference Panels** list.

3. Select the obsolete source tree from the **Source Trees** list.

4. Click **Remove**.

The IDE updates the **Source Trees** list. A reminder message to update source-tree references in the project appears.

5. Click **OK**, **Apply**, or **Save**.

The IDE saves the source-tree changes.

---

## Removing Remote Connections

Remove a remote connection that the project no longer uses.

1. Choose **Edit > Preferences**.

The IDE Preferences window appears.

2. Select the **Remote Connections** panel from the **IDE Preference Panels** list.

3. Select from the **Remote Connections** list the obsolete remote connection.

4. Click **Remove**.

The IDE updates the **Remote Connections** list.

5. Click **OK**, **Apply**, or **Save**.

The IDE saves the remote-connection changes.

## Editor Panels

The **Editor** section of the IDE Preference Panels list defines the editor settings assigned to a new project.

The Editor preference panels available on most IDE hosts include:

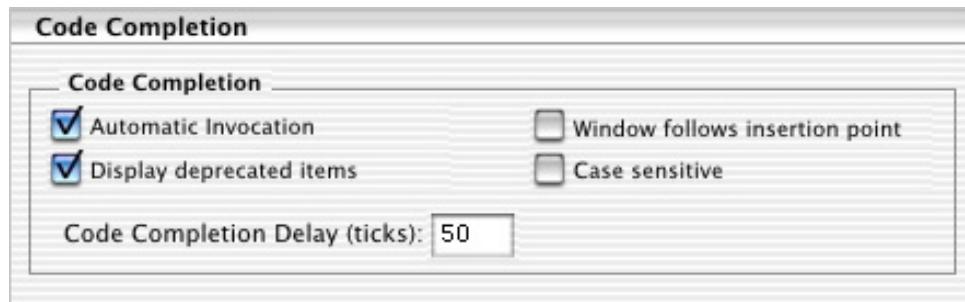
- “[Code Completion](#)” on page 366
- “[Editor Settings](#)” on page 366
- “[Font & Tabs](#)” on page 368
- “[Text Colors](#)” on page 371

## Code Completion

The **Code Completion** preference panel provides options for customizing the IDE code-completion behavior, including:

- automatic invocation and indexing
- window positioning and appearance delay
- case sensitivity

**Figure 31.11** Code Completion preference panel



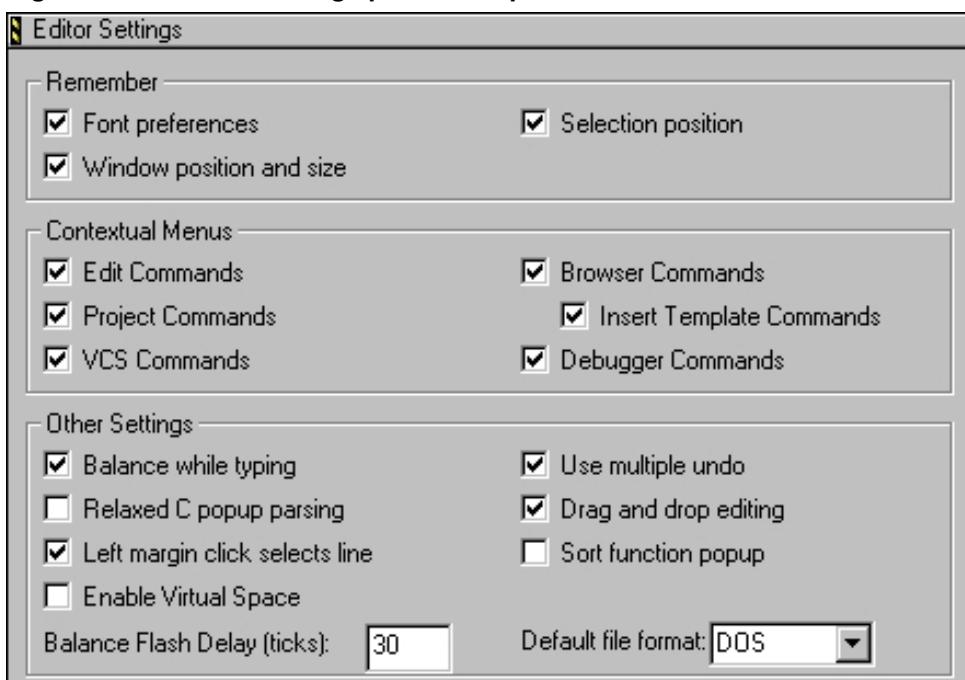
**Table 31.10** Code Completion preference panel—items

Item	Explanation
<a href="#">Automatic Invocation</a>	Select to automatically open the Code Completion window to complete programming-language symbols. Clear to manually open the window.
<a href="#">Window follows insertion point</a>	Select to have the Code Completion window follow the insertion point as you edit text. Clear to leave the window in place.
<a href="#">Display deprecated items</a>	Select to have the Code Completion window display obsolete items in gray text. Clear to have the window hide obsolete items.
<a href="#">Case sensitive</a>	Select to have the IDE consider case when completing code. Clear to have the IDE ignore case.
<a href="#">Code Completion Delay</a> (ticks)	Enter the number of ticks to wait before opening the Code Completion window. A tick is 1/60 of a second.

## Editor Settings

The **Editor Settings** preference panel provides options for customizing the editor, including:

- fonts, window locations, and insertion-point positions
- contextual menus
- additional editor-window features

**Figure 31.12 Editor Settings preference panel****Table 31.11 Editor Settings preference panel—items**

Item	Explanation
<a href="#">Font preferences</a>	Select to retain font settings for each source file. Clear to apply default font settings each time the IDE displays the source file.
<a href="#">Selection position</a>	Select to retain the text-insertion position in each source file.
<a href="#">Window position and size</a>	Select to retain the location and dimensions of each editor window.
<a href="#">Edit Commands</a>	Select to add <b>Edit</b> menu commands to contextual menus.
<a href="#">Browser Commands</a>	Select to add <b>Browser</b> menu commands to contextual menus. Also select in order to use the <b>Insert Template Commands</b> option.

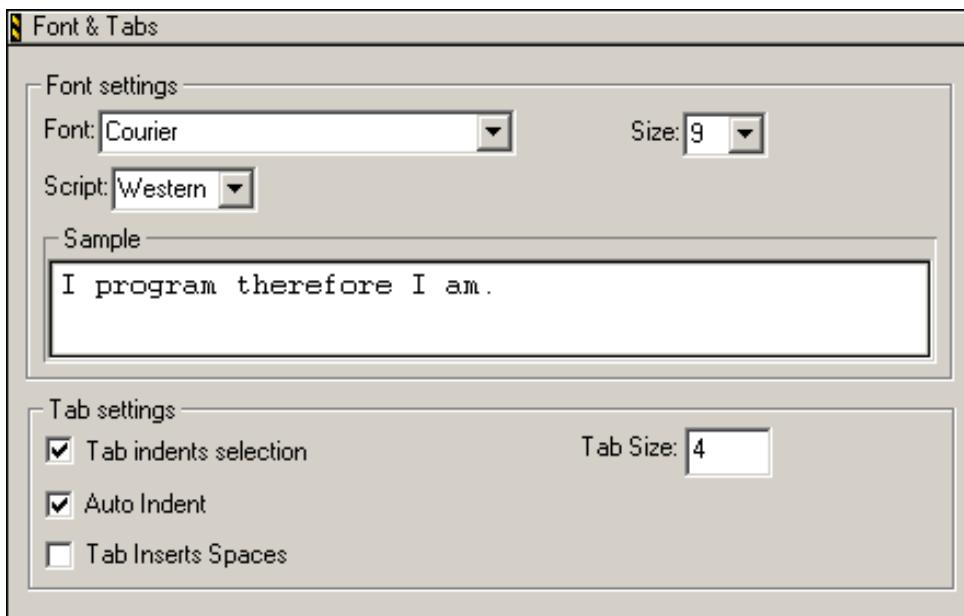
**Table 31.11 Editor Settings preference panel—items (*continued*)**

<b>Item</b>	<b>Explanation</b>
<a href="#">Insert Template Commands</a> (Macintosh)	Select to add the <b>Insert Template</b> submenu to contextual menus. The submenu displays source-defined function templates.
<a href="#">Project Commands</a>	Select to add <b>Project</b> menu commands to contextual menus.
<a href="#">VCS Commands</a>	Select to add <b>VCS</b> (Version Control System) menu commands to contextual menus.
<a href="#">Debugger Commands</a>	Select to add <b>Debug</b> menu commands to contextual menus.
<a href="#">Balance while typing</a>	Select to flash the matching (, [, or { after typing ), ], or } in an editor window.
<a href="#">Use multiple undo</a>	Select to allow multiple undo and redo operations while editing text.
<a href="#">Relaxed C popup parsing</a>	Select to allow the C parser to recognize some non-standard function formats and avoid skipping or misinterpreting some definition styles.
<a href="#">Drag and drop editing</a>	Select to allow drag-and-drop text editing.
<a href="#">Left margin click selects line</a>	Select to allow selection of an entire line of text by clicking in the left margin of the editor window.
<a href="#">Sort function popup</a>	Select to sort function names by alphabetical order in menus. Clear to sort function names by order of appearance in the source file.
<a href="#">Enable Virtual Space</a> (Windows and Macintosh)	Select to allow moving the text-insertion point beyond the end of a source-code line. Entering new text automatically inserts spaces between the former end of the line and the newly entered text.
<a href="#">Balance Flash Delay</a>	Enter the number of ticks to flash a balancing punctuation character. A tick is 1/60 of a second.
<a href="#">Default file format</a>	Choose the default end-of-line format used to save files.

## Font & Tabs

The **Font & Tabs** preference panel provides options for customizing settings used by the editor, including:

- font and font size used in editor windows
- auto indentation and tab size
- tabs on selections and replacing tabs with spaces

**Figure 31.13** Font & Tabs preference panel**Table 31.12** Font & Tabs preference panel—items

Item	Explanation
<a href="#">Font</a>	Choose the typeface displayed in editor windows.
<a href="#">Size</a>	Choose the font size displayed in editor windows.
<a href="#">Script</a> (Windows)	Choose the IDE script system. The script system maps keyboard keys to characters of an alphabet.
<a href="#">Tab indents selection</a>	Select to indent each line of selected text after pressing Tab. Clear to replace selected text with a tab character after pressing Tab.
<a href="#">Tab Size</a>	Enter the number of spaces to substitute in place of a tab character. This number applies to the <b>Tab Inserts Spaces</b> option.
<a href="#">Auto Indent</a>	Select to automatically apply the indentation level from the previous line of text to each new line created by pressing Enter or Return.
<a href="#">Tab Inserts Spaces</a>	Select to insert spaces instead of a tab character into text after pressing Tab. The <b>Tab Size</b> option determines the number of inserted spaces.

## Setting the Text Font

To set the text font, follow these steps:

1. Choose **Edit > Preferences**.
2. Select the **Font & Tabs** panel in the **Editor** selection in the **IDE Preference Panels** list.
3. In the **Font Settings** area of the **IDE Preferences** window, select a font type in the drop-down menu in the **Font** field.
4. Save your font in the **IDE Preferences** window.
  - Windows: Click **OK**.
  - Macintosh: Click **Save**.

The foreground text changes to the new font.

---

## Setting the Text Size

To set the text size, follow these steps:

1. Choose **Edit > Preferences**.
2. Select the **Font & Tabs** panel in the **Editor** selection in the **IDE Preference Panels** list.
3. In the **Font Settings** area of the **IDE Preferences** window, select the **Size** drop-down menu and choose a text point size (from 2 points to 24 points).
4. Save your text size in the **IDE Preferences** window.
  - Windows: Click **OK**.
  - Macintosh: Click **Save**.

The foreground text changes to the new size.

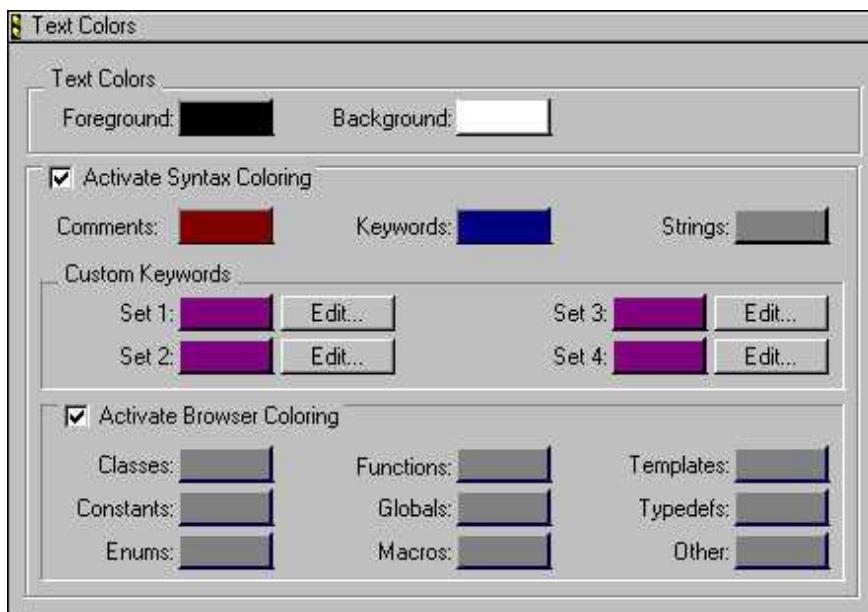
## Text Colors

The **Text Colors** preference panel customizes colors applied to elements of source code displayed in editor windows:

- default foreground and background in editor windows
- standard comments, keywords, and strings in source code
- custom-defined keywords
- browser symbols

Default settings provide a simple scheme of at least four source-code colors. If four colors do not provide sufficient detail, modify this preference panel to create more sophisticated color schemes.

**Figure 31.14** Text Colors preference panel



## Working with IDE Preferences

### Editor Panels

---

**Table 31.13 Text Colors preference panel—items**

Item	Explanation
<a href="#">Foreground</a>	Click the color swatch to display a subordinate dialog box. Use the dialog box to set the foreground color used in editor windows for text.
<a href="#">Background</a>	Click the color swatch to set the background color used in editor windows.
<a href="#">Activate Syntax Coloring</a>	Select to apply customized colors to comments, keywords, strings, and custom keywords in text. Clear to use the <b>Foreground</b> color for all text.
<a href="#">Comments</a>	Click the color swatch to set the color used for source-code comments.
<a href="#">Keywords</a>	Click the color swatch to set the color used for source-code language keywords.
<a href="#">Strings</a>	Click the color swatch to set the color used for source-code string literals.
Set 1, Set 2, Set 3, Set 4	Click a color swatch to set the color used for the corresponding custom-keyword set.
Edit	Click to add, modify, or remove keywords from the corresponding custom-keyword set.
<a href="#">Activate Browser Coloring</a>	Select to apply customized colors to browser symbols in text. Clear to use the <b>Foreground</b> color for all text.
Classes	Click the color swatch to set the color used for source-code classes.
Constants	Click the color swatch to set the color used for source-code constants.
Enums	Click the color swatch to set the color used for source-code enumerations.
Functions	Click the color swatch to set the color used for source-code functions.
Globals	Click the color swatch to set the color used for source-code global variables.
Macros	Click the color swatch to set the color used for source-code macros.
Templates	Click the color swatch to set the color used for source-code templates.

**Table 31.13 Text Colors preference panel—items (continued)**

Item	Explanation
TypeDefs	Click the color swatch to set the color used for source-code type definitions.
Other	Click the color swatch to set the color used for other symbols not specified in the <b>Activate Browser Coloring</b> section.

---

## Setting the Foreground Text Color

Use the **Foreground Color** option to configure the foreground text color displayed in editor windows.

1. Choose **Edit > Preferences**.
2. Select the **Text Colors** panel in the **Editor** selection in the **IDE Preference Panels** list.
3. Click the **Foreground** color box to set the editor's foreground color.
4. Pick color.
5. Click **OK** in the **Color Picker** window.
6. Save your colors in the **IDE Preferences** window.
  - Windows: Click **OK**.
  - Macintosh: Click **Save**.

The foreground text color changes to the new color.

---

## Setting the Background Text Color

Use the **Background Color** option to configure the background color displayed by all editor windows.

1. Choose **Edit > Preferences**.

2. Select the **Text Colors** panel in the **Editor** selection in the **IDE Preference Panels** list.
3. Click the **Background** color box to set the editor's background color.
4. Pick color.
5. Click **OK** in the **Color Picker** window.
6. Save your colors in the **IDE Preferences** window.
  - Windows: Click **OK**.
  - Macintosh: Click **Save**.

The background text color changes to the new color.

---

## Activate Syntax and Browser Coloring

Use the **Activate Syntax Coloring** and **Activate Browser Coloring** options to configure the syntax and browser colors that all your editor windows display.

1. Choose **Edit > Preferences**.
2. Select the **Text Colors** panel in the **Editor** selection in the **IDE Preference Panels** list.
3. Select the checkbox next to the **Activate Syntax Coloring** or the **Activate Browser Coloring** option.
4. Click on the colored box next to the item for which you want to configure color.
5. Pick color.
6. Click **OK** in the **Color Picker** window.
7. Save your colors in the **IDE Preferences** window.
  - Windows: Click **OK**.
  - Macintosh: Click **Save**.

The colors change to the new colors.

# Debugger Panels

The **Debugger** section of the IDE Preference Panels list defines the basic debugger settings assigned to a new project.

The Debugger preference panels available on most IDE hosts include:

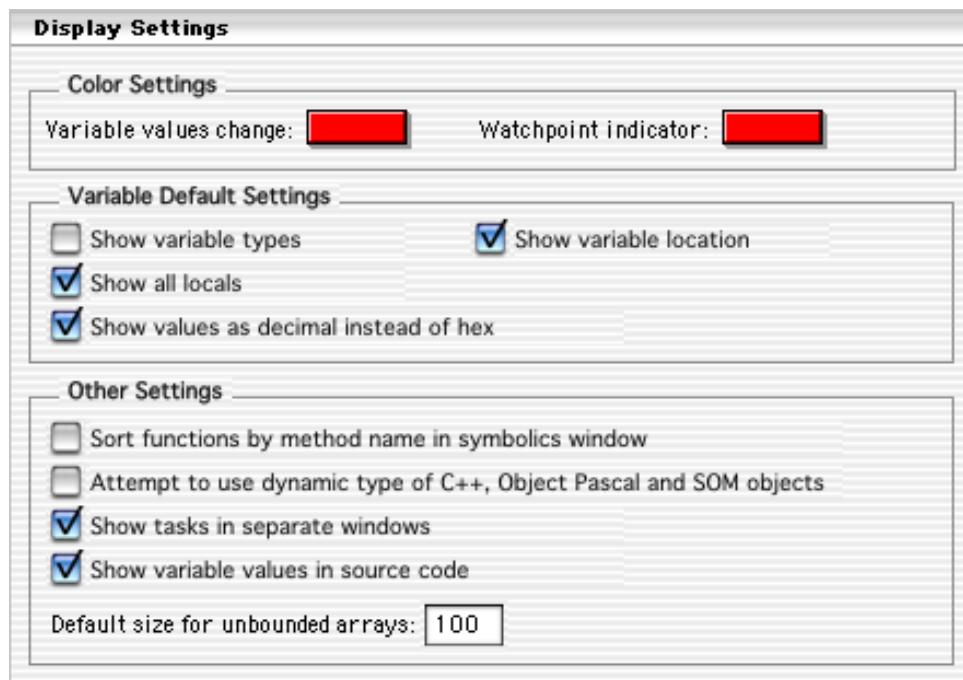
- [“Display Settings” on page 375](#)
- [“Windowing” on page 377](#)
- [“Global Settings” on page 378](#)
- [“Remote Connections” on page 380](#)

## Display Settings

The **Display Settings** preference panel provides options for customizing various aspects of the IDE Debugger, including:

- assignment of colors to changed variables and watchpoints
- view of variable types
- display of local variables
- use of decimal values
- sorting of functions
- use of dynamic objects

[Figure 31.15 on page 376](#) shows the Display Settings preference panel. [Table 31.14 on page 376](#) explains the items in the preference panel.

**Figure 31.15** Display Settings preference panel**Table 31.14** Display Settings preference panel—items

Item	Explanation
<a href="#">Variable values change</a>	Click the color swatch to set the color that indicates a changed variable value.
<a href="#">Watchpoint indicator</a>	Click the color swatch to set the color that indicates a changed watchpoint value.
<a href="#">Show variable types</a>	Select to always show the type associated with each variable.
<a href="#">Show variable location</a>	Select to display the <b>Location</b> column in the Variables pane of the Thread window.
<a href="#">Show all locals</a>	Select to always show all local variables. Clear to have the debugger show only variables near the program counter.
<a href="#">Show values as decimal instead of hex</a>	Select to always show decimal values instead of hexadecimal values.

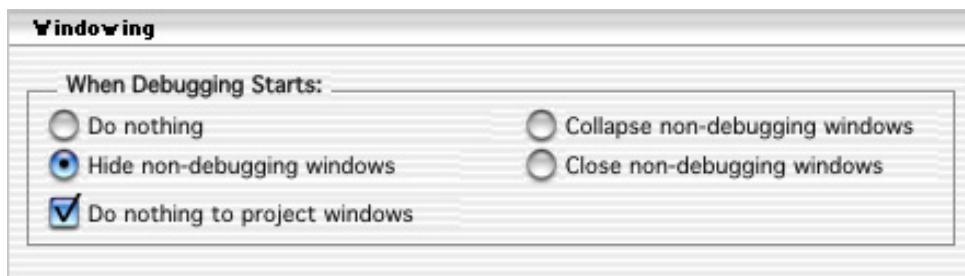
**Table 31.14 Display Settings preference panel—items (continued)**

<b>Item</b>	<b>Explanation</b>
<a href="#">Sort functions by method name in symbolics window</a>	Select to sort functions of the form <code>className::methodName</code> in the Symbolics window by <code>methodName</code> . Clear to sort by <code>className</code> .
<a href="#">Attempt to use dynamic type of C++, Object Pascal and SOM objects</a>	Select to attempt to display the runtime type of the specified language objects. Clear to display the static type.
<a href="#">Show tasks in separate windows</a>	Select to display tasks in separate Thread windows. Clear to shows all tasks in one window, with a list pop-up to choose a task to display. Restart active debugging sessions in order for changes to take effect.
<a href="#">Show variable values in source code</a>	Select to show variable values in contextual menus in the source code.
<a href="#">Default size for unbounded arrays</a>	Enter the default number of unbounded array elements to display in a View Array window.

## Windowing

The **Windowing** preference panel provides options for customizing how the debugger displays windows during debugging sessions, including non-debugging and project windows.

**Figure 31.16 Windowing preference panel**



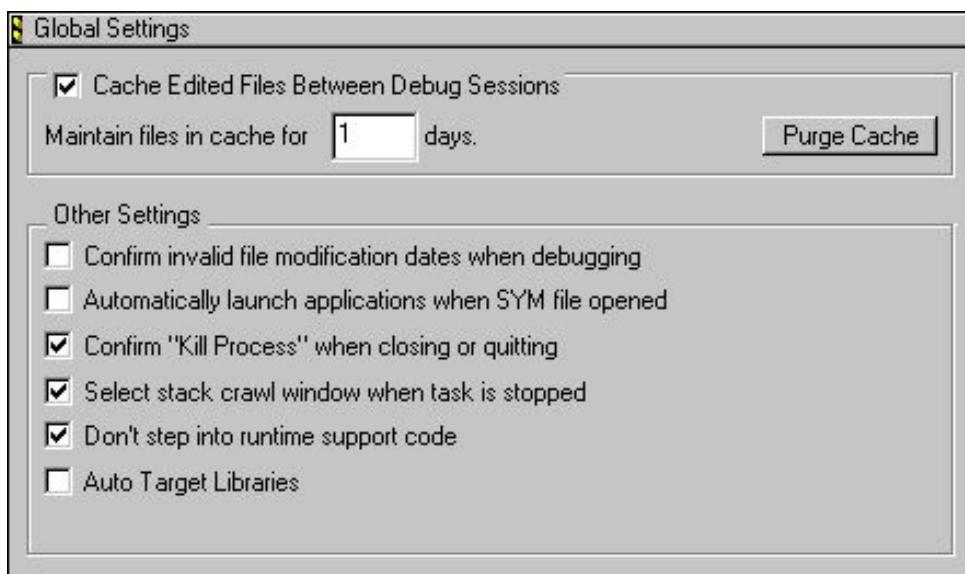
**Table 31.15 Windowing preference panel—items**

<b>Item</b>	<b>Explanation</b>
<a href="#"><u>Do nothing</u></a>	Select to leave all windows in place when starting a debugging session.
<a href="#"><u>Minimize non-debugging windows</u> (Windows)</a>	Select to minimize all non-debugging windows when starting a debugging session.
<a href="#"><u>Collapse non-debugging windows</u> (Macintosh, Solaris, and Linux)</a>	Select to collapse all non-debugging windows when starting a debugging session.
<a href="#"><u>Hide non-debugging windows</u></a>	Select to hide, but not close, all non-debugging windows when starting a debugging session.
<a href="#"><u>Close non-debugging windows</u></a>	Select to close all non-debugging windows, except for the active project window, when starting a debugging session.
<a href="#"><u>Do nothing to project windows</u></a>	Select to prevent the IDE from hiding project windows when starting a debugging session.
<a href="#"><u>Use Debugging Monitor</u> (Classic Macintosh)</a>	Select to use a second monitor during debugging sessions.
<a href="#"><u>Monitor for debugging</u> (Classic Macintosh)</a>	Choose the monitor to display debugging windows. The coordinates in parentheses identify the selected monitor in the QuickDraw® coordinate space.
<a href="#"><u>Move open windows to debugging monitor when debugging starts</u> (Classic Macintosh)</a>	Select to move all open windows to the selected debugging monitor when a debugging session starts.
<a href="#"><u>Open windows on debugging monitor during debugging</u> (Classic Macintosh)</a>	Select to display on the debugging monitor any window opened during a debugging session.

## Global Settings

The **Global Settings** preference panel provides options for customizing various global options for the debugger, including:

- file caching to accelerate debugger sessions
- automatic launch of applications and libraries
- confirmation of attempts to close or quit debugging sessions

**Figure 31.17 Global Settings preference panel****Table 31.16 Global Settings preference panel—items**

<b>Item</b>	<b>Explanation</b>
<a href="#"><u>Cache Edited Files Between Debug Sessions</u></a>	Select to maintain a cache of modified files between debugging sessions. Use this option to debug through the original source code for files modified since the last build.
<a href="#"><u>Maintain files in cache</u></a>	Enter the number of days that the IDE maintains its file cache.
<a href="#"><u>Purge Cache</u></a>	Click to delete the file cache maintained by the IDE, freeing memory and disk space.
<a href="#"><u>Confirm invalid file modification dates when debugging</u></a>	Select to have the IDE display a warning message when debugging a project with mis-matched file modification dates.
<a href="#"><u>Automatically launch applications when SYM file opened</u></a>	Select to automatically launch the application program associated with an opened symbolics file.
<a href="#"><u>Confirm "Kill Process" when closing or quitting</u></a>	Select to prompt for confirmation before killing processes upon quitting a debugging session.
<a href="#"><u>Select stack crawl window when task is stopped</u></a>	Select to bring forward the Stack Crawl window (also known as the Thread window) after the debugger stops tasks.

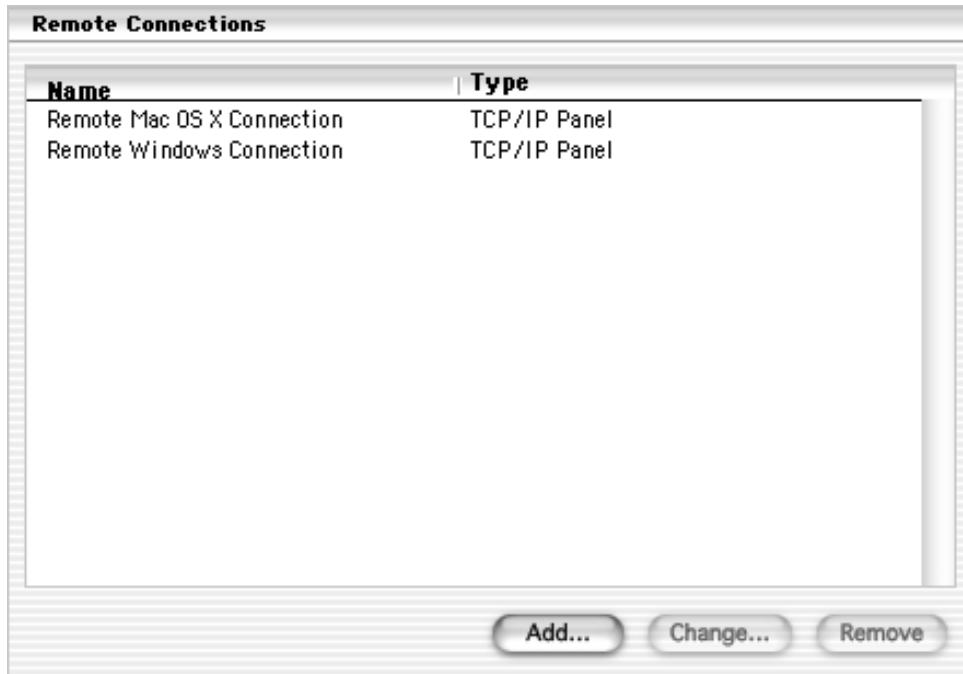
Table 31.16 Global Settings preference panel—items (*continued*)

Item	Explanation
<a href="#">Don't step into runtime support code</a>	Select to have the IDE skip stepping into Metrowerks Standard Library (MSL) runtime support code and instead directly step into your own code.
<a href="#">Auto Target Libraries</a>	Select to have the IDE attempt to debug dynamically linked libraries (DLLs) loaded by the target application.

## Remote Connections

The **Remote Connections** preference panel configures general network settings for remote-debugging connections between the host computer and other computers. Use these general settings as the basis for defining more specific connections for individual projects in conjunction with the **Remote Debugging** settings panel. The Target Settings window contains the **Remote Debugging** settings panel.

Figure 31.18 Remote Connections preference panel



**Table 31.17 Remote Connections preference panel—items**

Item	Explanation
Remote Connection list	Displays the name and connection type of all remote connections currently defined.
Add	Click to add a new remote connection to the Remote Connection list.
Change	Click to change the settings of the selected remote connection.
Remove	Click to remove the selected remote connection from the Remote Connection list.

---

## Adding Remote Connections

Add a remote connection that defines a general network connection between the host computer and a remote computer.

1. Choose **Edit > Preferences**.

The IDE Preferences window appears.

2. Select the **Remote Connections** panel from the **IDE Preference Panels** list.

3. Click **Add**.

The **New Connection** dialog box appears.

4. Enter in the **Name** field the name for the general remote connection.

5. Choose from the **Debugger** pop-up menu the desired debugger for use with the remote connection.

6. Configure the **Browse in processes window** option as desired:

- selected—the IDE filters the **Processes** window list and the list of available debuggers for an opened symbolics file. The filter prevents an unavailable remote connection from appearing in either list.
- cleared—the IDE does not filter the **Processes** window list or the list of available debuggers for an opened symbolics file. Both available and unavailable remote connections appear in the lists.

7. Choose from the **Connection Type** pop-up menu the desired network protocol for the remote connection.
8. Enter in the **IP Address** field the Internet Protocol address of the remote computer.
9. Click **OK**.

The IDE adds the new remote connection to the **Remote Connections** list.

10. Click **OK**, **Apply**, or **Save**.

The IDE saves the remote-connection changes.

---

## Changing Remote Connections

Change a remote connection to update network-connection information between the host computer and a remote computer.

1. Choose **Edit > Preferences**.

The IDE Preferences window appears.

2. Select the **Remote Connections** panel from the **IDE Preference Panels** list.

3. Select from the **Remote Connections** list the remote connection that requires modification.

4. Click **Change**.

A dialog box appears with the current network settings for the selected remote connection.

5. If needed, enter in the **Name** field a new name for the general remote connection.

6. If needed, choose from the **Debugger** pop-up menu a new debugger for use with the remote connection.

7. If needed, toggle the **Browse in processes window** option.

8. If needed, choose from the **Connection Type** pop-up menu a new network protocol for the remote connection.

9. If needed, enter in the **IP Address** field a new Internet Protocol address for the remote computer.

10. Click **OK**.

The IDE updates the remote connection and displays changes in the **Remote Connections** list.

11. Click **OK**, **Apply**, or **Save**.

The IDE saves the remote-connection changes.

---

## Removing Remote Connections

Remove a remote connection that the project no longer uses.

1. Choose **Edit > Preferences**.

The IDE Preferences window appears.

2. Select the **Remote Connections** panel from the **IDE Preference Panels** list.

3. Select from the **Remote Connections** list the obsolete remote connection.

4. Click **Remove**.

The IDE updates the **Remote Connections** list.

5. Click **OK**, **Apply**, or **Save**.

The IDE saves the remote-connection changes.

## RAD Tools Panels

The **RAD Tools** section of the IDE Preference Panels list defines the Rapid Application Development tool settings assigned to a new project.

The RAD Tools preference panels available on most IDE hosts include:

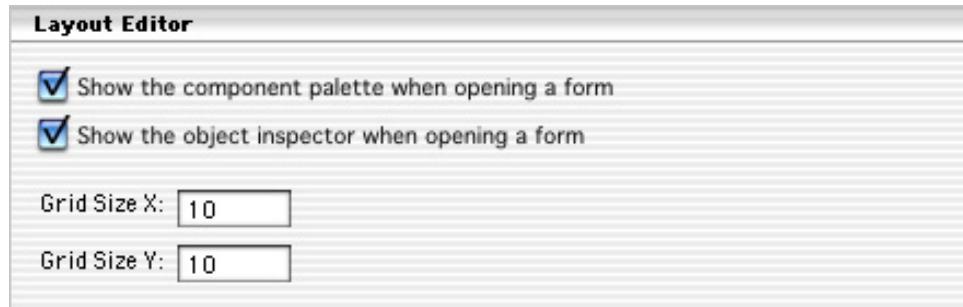
- [“Layout Editor” on page 384](#)

## Layout Editor

The **Layout Editor** preference panel provides options for customizing the Layout Editor tool, including:

- display of the Component Palette and the Object Inspector
- grid size displayed onscreen

**Figure 31.19 Layout Editor preference panel**



**Table 31.18 Layout Editor preference panel—items**

Item	Explanation
<a href="#">Show the component palette when opening a form</a>	Select to automatically display the Component Palette after opening a form.
<a href="#">Show the object inspector when opening a form</a>	Select to automatically display the Object Inspector after opening a form.
<a href="#">Grid Size X</a>	Enter the number of pixels to space between markings on the x axis of the Layout Editor grid.
<a href="#">Grid Size Y</a>	Enter the number of pixels to space between markings on the y axis of the Layout Editor grid.

# Working with Target Settings

---

This chapter explains core CodeWarrior™ IDE target-settings panels and provides basic information on target-settings options for the current project’s build targets. Consult the *Targeting* documentation for information on platform-specific target-settings panels.

This chapter contains these sections:

- [“Target Settings Window” on page 385](#)
- [“Target Panels” on page 387](#)
- [“Code Generation Panels” on page 398](#)
- [“Editor Panels” on page 401](#)
- [“Debugger Panels” on page 403](#)

Abbreviated descriptions appear in this chapter. See [“Preference and Target Settings Options” on page 411](#) for more information on target-settings-panel options.

## Target Settings Window

The **Target Settings** window lists settings for the current project’s build targets. These target settings supersede global IDE preferences defined in the **IDE Preferences** window.

The Target Settings window lists settings by group:

- **Target**—configures overall build-target settings, such as names, browser caching, file mappings, and access paths
- **Language Settings**—configures programming-language settings. Consult the *Targeting* documentation for more information about these settings panels.
- **Code Generation**—configures processor, disassembler, and optimization settings for generating code

## Working with Target Settings

### Target Settings Window

- **Linker**—configures linker settings for transforming object code into a final executable file. Consult the *Targeting* documentation for more information about these settings panels.
- **Editor**—configures custom keyword sets and colors
- **Debugger**—configures settings for executable files, program suspension, and remote debugging

Figure 32.1 Target Settings window

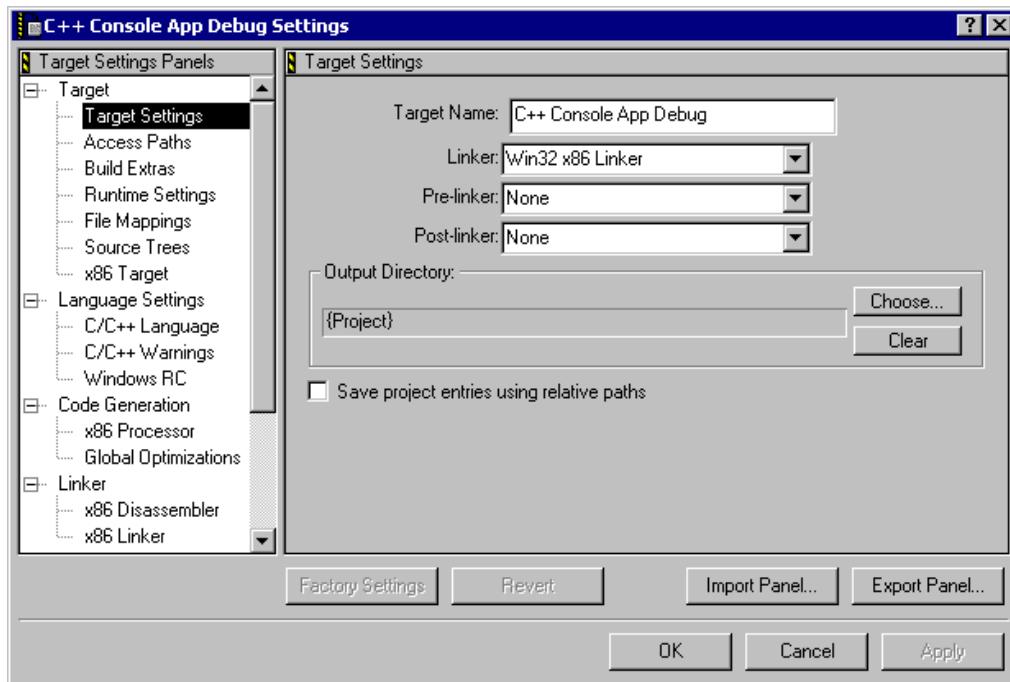


Table 32.1 Target Settings window—items

Item	Explanation
Target Settings Panels list	Lists settings panels, organized by group. Click the hierarchical control next to a group name to show or hide individual settings panels.
Settings panel	Shows the options for the selected item in the <b>Target Settings Panels</b> list.
<a href="#">Factory Settings</a>	Click to restore the default options for the current settings panel.

**Table 32.1 Target Settings window—items (*continued*)**

Item	Explanation
<a href="#">Revert Panel</a>	Click to restore the most recently saved options for the current settings panel.
<a href="#">Export Panel</a>	Click to save an XML file that contains options for the current settings panel.
<a href="#">Import Panel</a>	Click to open an XML file that contains options for the current settings panel.
OK (Windows)	Click to save modifications to all settings panels and close the window.
Cancel (Windows)	Click to discard modifications to all settings panels and close the window.
Apply (Windows)	Click to confirm modifications to all settings panels.
Save (Macintosh, Solaris, and Linux)	Click to save modifications to all settings panels.

---

## Opening the Target Settings Window

Use the **Target Settings** window to modify build-target options for the current project.

- Choose **Edit > targetname Settings**.

The **Target Settings** window appears.

# Target Panels

The **Target** section of the Target Settings Panels list defines the general target settings assigned to a new project.

The Target settings panels available on most IDE hosts include:

- [“Target Settings” on page 388](#)
- [“Access Paths” on page 389](#)
- [“Build Extras” on page 392](#)
- [“Runtime Settings” on page 394](#)

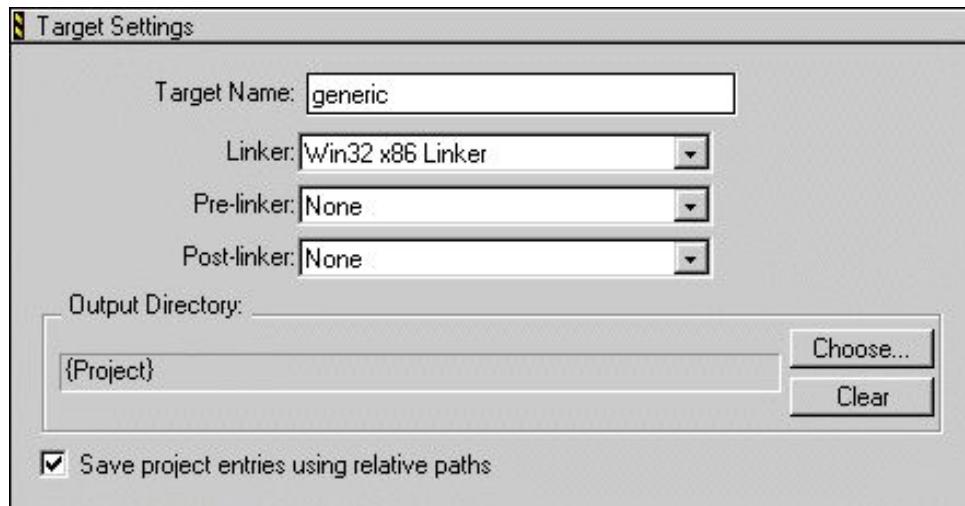
- “[File Mappings](#)” on page 396
- “[Source Trees](#)” on page 398

## Target Settings

The **Target Settings** panel provides options for:

- setting the name of the current build target
- setting the linker, pre-linker, and post-linker for the build target
- specifying the project output directory for the final output file

**Figure 32.2 Target Settings panel**



**Table 32.2 Target Settings panel—items**

Item	Explanation
<a href="#">Target Name</a>	Enter a name (26 or fewer characters) for the selected build target as it will appear in the project window.
<a href="#">Linker</a>	Select the linker to use on the current build target.
<a href="#">Pre-linker</a>	Select the pre-linker to use on the current build target.
<a href="#">Post-linker</a>	Select the post-linker to use on the current build target.

Table 32.2 Target Settings panel—items (*continued*)

Item	Explanation
<a href="#"><b>Output Directory</b></a>	Shows the location where IDE creates the output binary file. Click <b>Choose</b> to change this location.
<b>Choose</b>	Click to select the directory in which the IDE saves the output binary file.
<b>Clear</b>	Click to delete the current <b>Output Directory</b> path.
<a href="#"><b>Save project entries using relative paths</b></a>	Select to save project file entries using a relative path from a defined access path. This option is helpful if the project has multiple files with the same name.

## Access Paths

The **Access Paths** settings panel defines the search paths for locating and accessing a build target's system files and header files.

---

**NOTE** The Windows version of the Access Paths settings panel displays either User Paths or System Paths, depending on the selected radio button. The Macintosh, Solaris, and Linux versions of the Access Paths settings panel display both User Paths and System Paths.

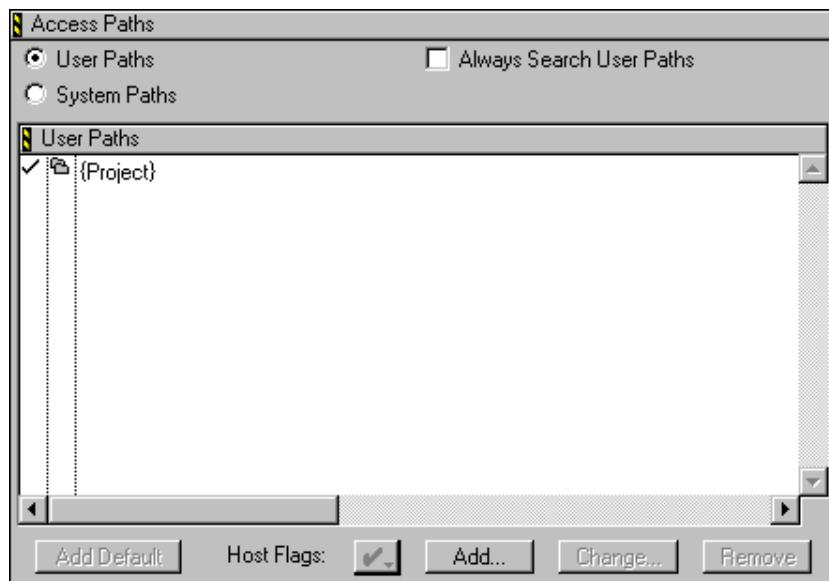
---

## Working with Target Settings

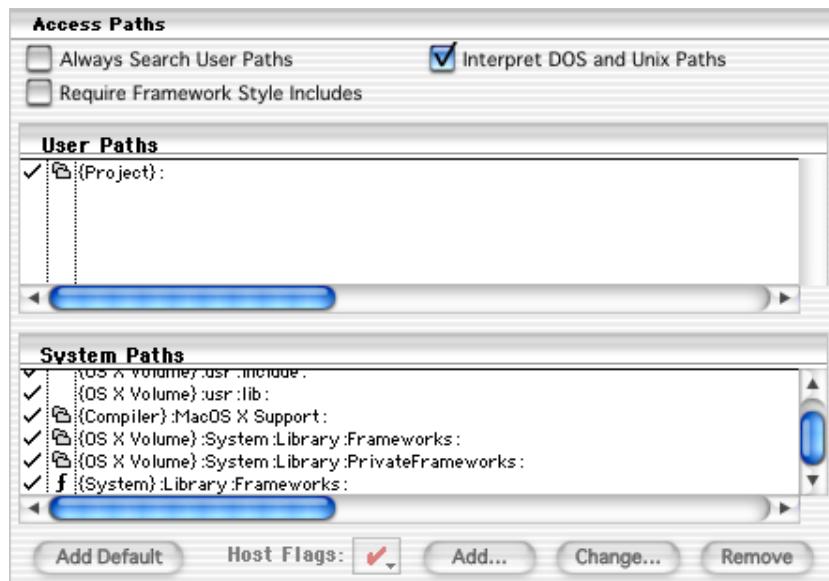
### Target Panels

---

**Figure 32.3 Access Paths settings panel (Windows)**



**Figure 32.4 Access Paths settings panel (Macintosh)**



**Table 32.3 Access Paths settings panel—items**

Item	Explanation
<a href="#"><u>User Paths</u></a>	Click the radio button to display the <b>User Paths</b> list (those paths searched by #include "..." statements).
<a href="#"><u>System Paths</u></a>	Click the radio button to display the <b>System Paths</b> list (those paths searched by #include <...> statements).
<a href="#"><u>Always Search User Paths</u></a>	Select to treat #include <...> statements the same as #include "..." statements.
<a href="#"><u>Interpret DOS and Unix Paths</u> (Macintosh)</a>	Select to treat / and \ as subfolder separator characters. Clear to treat / and \ as ordinary text.
<a href="#"><u>Require Framework Style Includes</u> (Mac OS X)</a>	Select to require #include statements of the form LibraryName/HeaderFile.h. Clear to allow statements of the form HeaderFile.h.
User Paths list	Displays a list of currently defined user-level access paths.
System Paths list	Displays a list of currently defined system-level access paths.
<a href="#"><u>Add Default</u></a>	Click to restore the default user- and system-level access paths.
<a href="#"><u>Host Flags</u> list pop-up</a>	Choose the host platforms that can use the selected access path.
<a href="#"><u>Add</u></a>	Click to add a user- or system-level access path.
<a href="#"><u>Change</u></a>	Click to modify the selected user- or system-level access path.
<a href="#"><u>Remove</u></a>	Click to remove the selected user- or system-level access path.

The **User Paths** and **System Paths** lists display columns with status icons for each access path. Furthermore, there are different types of access paths. [Table 32.4](#) explains these items.

**Table 32.4 User Paths and System Paths list columns**

Name	Icon	Explanation
Search status	✓	A checkmark icon indicates an active access path that the IDE searches.
		No checkmark icon indicates an inactive access path that the IDE does not search.

**Table 32.4 User Paths and System Paths list columns (*continued*)**

Name	Icon	Explanation
Recursive search		A folder icon indicates that the IDE recursively searches the subdirectories of the access path.
		No folder icon indicates that the IDE does not recursively search the access path.
Framework (Mac OS X development)		An <i>f</i> icon indicates that the access path points to a framework. Framework paths are implicitly recursive.
		No <i>f</i> icon indicates that the access path does not point to a framework.
Access path		Shows the full access path to the selected directory. Access paths have these types: <ul style="list-style-type: none"> <li>• Absolute—the complete path, from the root level of the hard drive to the directory, including all intermediate directories</li> <li>• Project—the path from the project file relative to the designated directory</li> <li>• CodeWarrior—the path from the CodeWarrior IDE relative to the designated directory</li> <li>• System—the path from the operating system's base directory relative to the designated directory</li> <li>• Source tree—the path from a user-defined source tree relative to the designated directory</li> </ul>

## Build Extras

The **Build Extras** settings panel contains options that define how the CodeWarrior IDE builds a project.

Figure 32.5 Build Extras settings panel

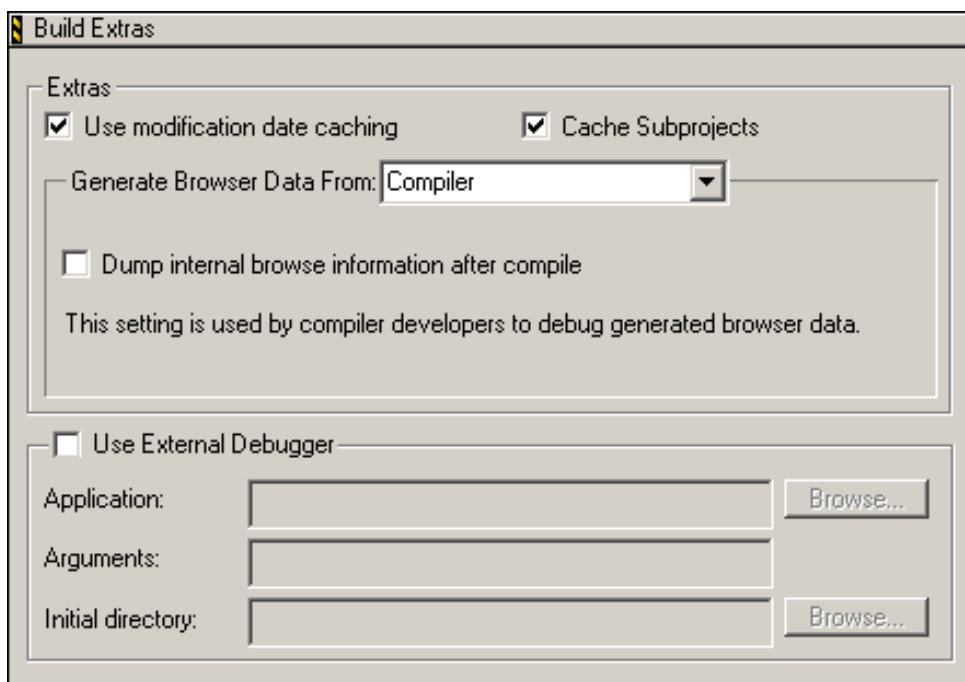


Table 32.5 Build Extras settings panel—items

Item	Explanation
<a href="#">Use modification date caching</a>	Select to have the IDE cache modification-date information and use that information each time it builds a target.
<a href="#">Cache Subprojects</a>	Select to improve multi-project updating and linking speed.
<a href="#">Generate Browser Data From</a>	Choose whether the IDE generates browser data for the project, and the method by which the IDE generates that data.
<a href="#">Dump internal browse information after compile</a>	Select to have the IDE dump raw browser information for viewing. This option appears after selecting <b>Compiler</b> from the <a href="#">Generate Browser Data From</a> pop-up menu.
Prefix file	Enter the path to your project's prefix file. This option appears after selecting <b>Language Parser</b> from the <a href="#">Generate Browser Data From</a> pop-up menu.

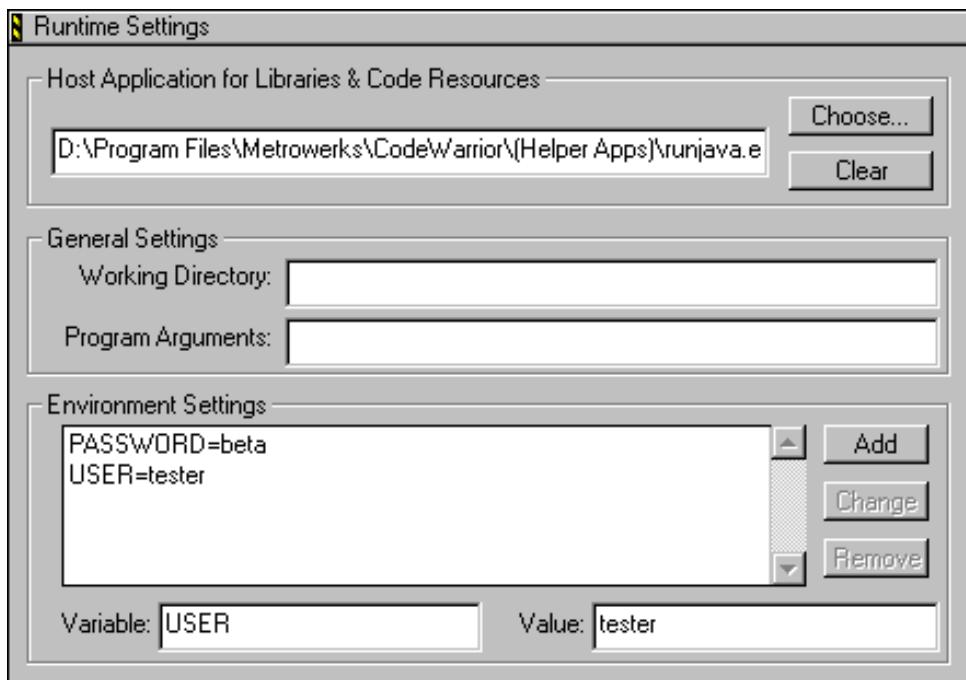
**Table 32.5 Build Extras settings panel—items (*continued*)**

<b>Item</b>	<b>Explanation</b>
Macro file	Enter the path to your project's macro file. This option appears after selecting <b>Language Parser</b> from the <a href="#">Generate Browser Data From</a> pop-up menu.
<a href="#">Use External Debugger</a> (Windows)	Select to use an external debugger instead of the CodeWarrior debugger.
<a href="#">Application</a> (Windows)	Click <b>Browse</b> to select the external debugger application. Alternatively, enter in the field the path to the external debugger.
<a href="#">Arguments</a> (Windows)	Enter in this field any program arguments to pass to the external debugger when the IDE transfers control.
<a href="#">Initial directory</a> (Windows)	Click <b>Browse</b> to select an initial directory for the external debugger. Alternatively, enter in the field the path to the initial directory.

## Runtime Settings

The **Runtime Settings** panel specifies a debugging application for non-executable files. Dynamic linked libraries (DLLs), shared libraries, and code resources are sample non-executable files. For example, use this panel to specify a debugging application for Adobe® Photoshop® plug-ins.

**Figure 32.6 Runtime Settings panel**



**Table 32.6 Runtime Settings panel—items**

Item	Explanation
<a href="#">Host Application for Libraries &amp; Code Resources</a>	Click <b>Choose</b> to select the application program for debugging non-executable files. Alternatively, enter in the field the path to the application program. Click <b>Clear</b> to delete the current field entry.
<a href="#">Working Directory</a>	Enter the path to a directory used for debugging the non-executable files. Leave this field blank to use the same directory that contains the non-executable files.
<a href="#">Program Arguments</a>	Enter a command line of program arguments to pass to the host application when the IDE transfers control.
<a href="#">Environment Settings</a>	Lists the environment variables that have been added to the build target.
Add	Click to add the current <b>Variable</b> and <b>Value</b> pair to the Environment Settings list.

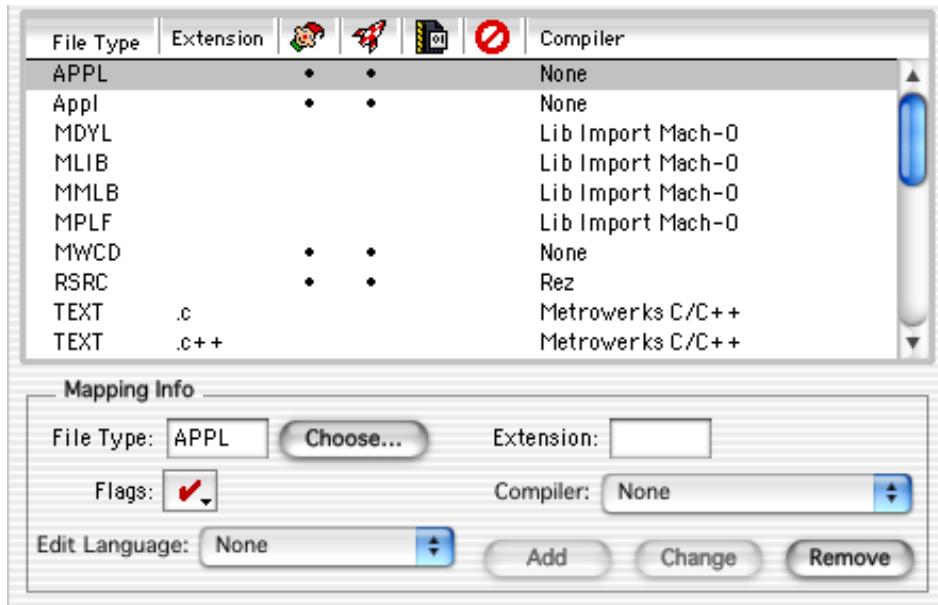
Table 32.6 Runtime Settings panel—items (*continued*)

Item	Explanation
Change	Click to replace the selected entry in the Environment Settings list with the current <b>Variable</b> and <b>Value</b> pair.
Remove	Click to delete the selected environment variable from the Environment Settings list.
<u>Variable</u>	Enter a name for the environment variable. This name pairs with the information in the <b>Value</b> field.
<u>Value</u>	Enter a value for the environment variable. This value pairs with the information in the <b>Variable</b> field.

## File Mappings

The **File Mappings** settings panel associates file-name extensions with a CodeWarrior plug-in compiler. These associations determine whether the IDE recognizes a source file by its filename extension or file type. Use the settings panel to add, change, and remove file mappings.

Figure 32.7 File Mappings settings panel



**Table 32.7 File Mappings settings panel—items**

Item	Icon	Explanation
File Mappings list		Displays a list of currently defined mappings between file-name extensions and plug-in compilers.
<a href="#">File Type</a>		Enter in this field a file type (such as TEXT) for the file mapping. Alternatively, click <b>Choose</b> to set the file type by selecting an example file. This file type also appears in the corresponding column of the File Mappings list.
<a href="#">Extension</a>		Enter in this field the file-name extension (such as .cpp) for the file mapping. This file-name extension also appears in the corresponding column of the File Mappings list.
Resource File flag		A bullet in this column denotes a resource file. The IDE includes these resource files when building the final output file. Use the <b>Flags</b> pop-up menu to toggle this flag.
Launchable flag		A bullet in this column denotes a launchable file. The IDE opens launchable files with the application that created them. Double-click launchable files from the Project window. Use the <b>Flags</b> pop-up menu to toggle this flag.
Precompiled File flag		A bullet in this column denotes a precompiled file. The IDE builds precompiled files before building other files. Use the <b>Flags</b> pop-up menu to toggle this flag.
Ignored By Make flag		A bullet in this column denotes a file ignored by the compiler during builds. For example, use this option to ignore text (.txt) files or document (.doc) files. Use the <b>Flags</b> pop-up menu to toggle this flag.
<a href="#">Compiler</a>		Choose from this list pop-up the plug-in compiler to associate with the selected file mapping. This compiler selection also appears in the corresponding column of the File Mappings list.
Flags		Choose from this pop-up menu the desired flags for the selected file mapping. A checkmark indicates an active flag. Bullets appear in the corresponding columns of the File Mappings list to reflect flag states.
<a href="#">Edit Language</a>		Choose from this list pop-up the desired language to associate with the selected file mapping. The IDE applies the appropriate syntax coloring for the selected language.
Add		Click to add the current <b>File Type</b> , <b>Extension</b> , <b>Flags</b> , <b>Compiler</b> , and <b>Edit Language</b> entries to the File Mappings list.

**Table 32.7 File Mappings settings panel—items (continued)**

Item	Icon	Explanation
Change		Click to change the selected item in the File Mappings list to reflect the current <b>File Type</b> , <b>Extension</b> , <b>Flags</b> , <b>Compiler</b> , and <b>Edit Language</b> entries.
Remove		Click to remove the selected item in the File Mappings list.

## Source Trees

The **Source Trees** settings panel in the Target Settings window defines project-specific root paths. These project-specific paths supersede the global root paths defined in the **Source Trees** preference panel of the IDE Preferences window.

## Code Generation Panels

The **Code Generation** section of the Target Settings Panels list provides a single core panel for configuring optimization routines. Consult the *Targeting* documentation for more information about platform-specific settings panels.

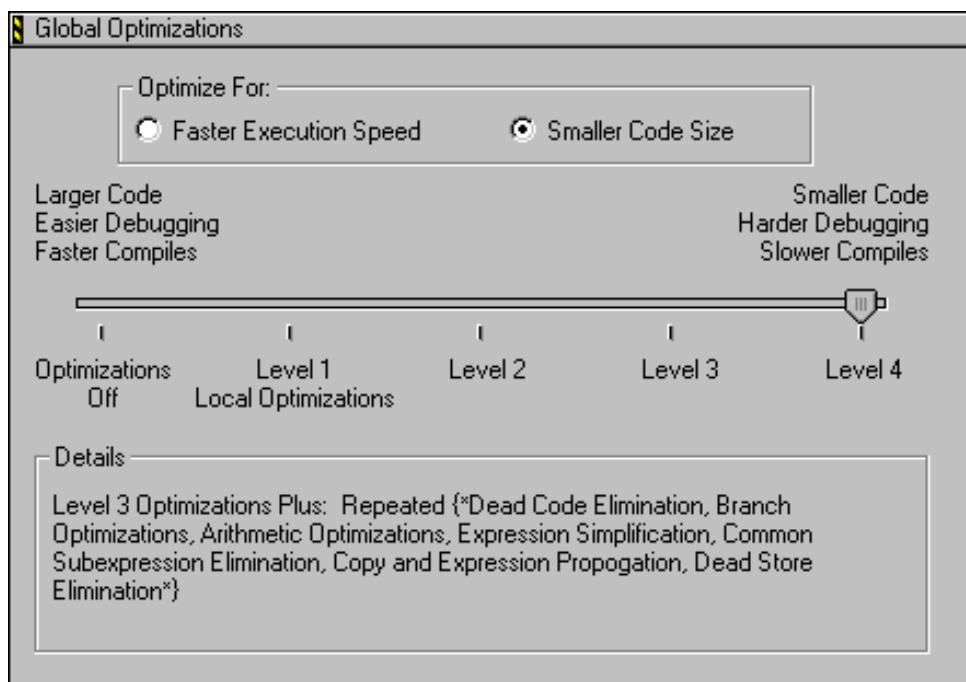
## Global Optimizations

The **Global Optimizations** settings panel configures how the compiler optimizes object code. All optimization routines rearrange object code without affecting its logical execution sequence.

---

**NOTE** Always debug programs with optimization routines disabled. The IDE does not provide source views of optimized code.

---

**Figure 32.8 Global Optimizations settings panel****Table 32.8 Global Optimizations settings panel—items**

Item	Explanation
Faster Execution Speed	Select to favor optimization routines that increase the execution speed of the final object code, at the expense of larger code size.
Smaller Code Size	Select to favor optimization routines that reduce the size of the final object code, at the expense of slower execution speed.
Optimization Level slider	Move to the desired optimization level. The IDE applies more optimization routines at higher optimization levels. The <b>Details</b> field lists the active optimization routines.

The **Details** field lists individual optimization routines applied at the selected optimization level. [Table 32.9 on page 400](#) explains these optimizations and their availability at certain optimization levels.

**Table 32.9 Optimization routines**

<b>Optimization Routine</b>	<b>Explanation</b>	<b>Optimization Level</b>
Global Register Allocation or Global Register Allocation Only for Temporary Values	Stores working values of heavily used variables in registers instead of memory.	1, 2, 3, 4
Dead Code Elimination	Removes statements never logically executed or referred to by other statements.	1, 2, 3, 4
Branch Optimizations	Merges and restructures portions of the intermediate code translation in order to reduce branch instructions.	1, 2, 3, 4
Arithmetic Operations	Replaces intensive computational instructions with faster equivalent instructions that produce the same result.	1, 2, 3, 4
Expression Simplification	Replaces complex arithmetic expressions with simplified equivalent expressions.	1, 2, 3, 4
Common Subexpression Elimination	Replaces redundant expressions with a single expression.	2, 3, 4
Copy Propagation or Copy and Expression Propagation	Replaces multiple occurrences of one variable with a single occurrence.	2, 3, 4
Peephole Optimization	Applies local optimization routines to small sections of code.	2, 3, 4
Dead Store Elimination	Removes assignments to a variable that goes unused before being reassigned again.	3, 4
Live Range Splitting	Reduces variable lifetimes to achieve optimal allocation. Shorter variable lifetimes reduce register spilling.	3, 4
Loop-Invariant Code Motion	Moves static computations outside of a loop	3, 4
Strength Reduction	Inside loops, replaces multiplication instructions with addition instructions.	3, 4
Loop Transformations	Reorganizes loop object code in order to reduce setup and completion-test overhead.	3, 4

**Table 32.9 Optimization routines (*continued*)**

Optimization Routine	Explanation	Optimization Level
Loop Unrolling or Loop Unrolling (Opt for Speed Only)	Duplicates code inside a loop in order to spread over more operations branch and completion-test overhead.	3, 4
Vectorization	For processors that support vector optimizations, translates computations with code-loop arrays into equivalent vector instructions.	3, 4
Lifetime Based Register Allocation or Register Coloring	In a particular routine, uses the same processor register to store different variables, as long as no statement uses those variables simultaneously.	3, 4
Instruction Scheduling	Rearranges the instruction sequence to reduce conflicts among registers and processor resources.	3, 4
Repeated	Iterates the optimization routines listed between { * and * }.	4

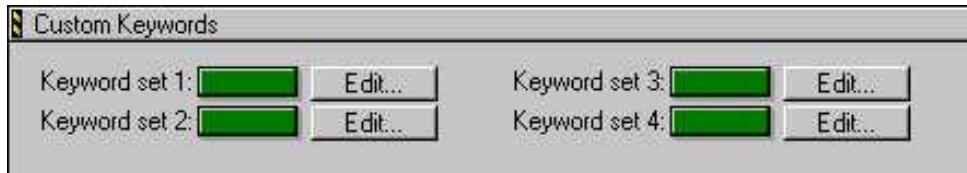
## Editor Panels

The **Editor** section of the Target Settings Panels list provides a single core panel for configuring custom keywords within a project.

## Custom Keywords

The **Custom Keywords** settings panel configures as many as four keyword sets, each with a list of keywords, and syntax coloring for a project. These project-specific settings supersede the global settings defined in the **Text Colors** preference panel of the IDE Preferences window.

**Figure 32.9 Custom Keywords settings panel**



**Table 32.10 Custom Keywords settings panel—items**

Item	Explanation
Keyword set 1, Keyword set 2, Keyword set 3, Keyword set 4	Click a color swatch to set the color used for the corresponding custom-keyword set.
Edit	Click to add, modify, or remove keywords from the corresponding custom-keyword set.

---

## Adding a Keyword to a Keyword Set

To add a keyword to a keyword set, follow these steps:

1. Click **Edit** next to the desired keyword set.

A dialog box appears. This dialog box lists the current collection of keywords in the keyword set.

2. Enter the new keyword into the field at the top of the dialog box.

3. Click **Add**.

The new keyword appears in the keyword list.

4. Select **Case Sensitive** as desired.

When selected, the IDE treats the case of each keyword in the keyword set as significant. When cleared, the IDE ignores the case of each keyword in the keyword set.

5. Click **Done**.

The IDE saves the modified keyword set.

## Removing a Keyword from a Keyword Set

To remove a keyword from a keyword set, follow these steps:

1. Click **Edit** next to the desired keyword set.

A dialog box appears. This dialog box lists the current collection of keywords in the keyword set.

2. Select the obsolete keyword in the Custom Keywords list.
3. Press the delete key for your platform.
  - Windows, Solaris, and Linux: Backspace
  - Macintosh: Delete
4. Click **Done**.

The IDE saves the modified keyword set.

## Debugger Panels

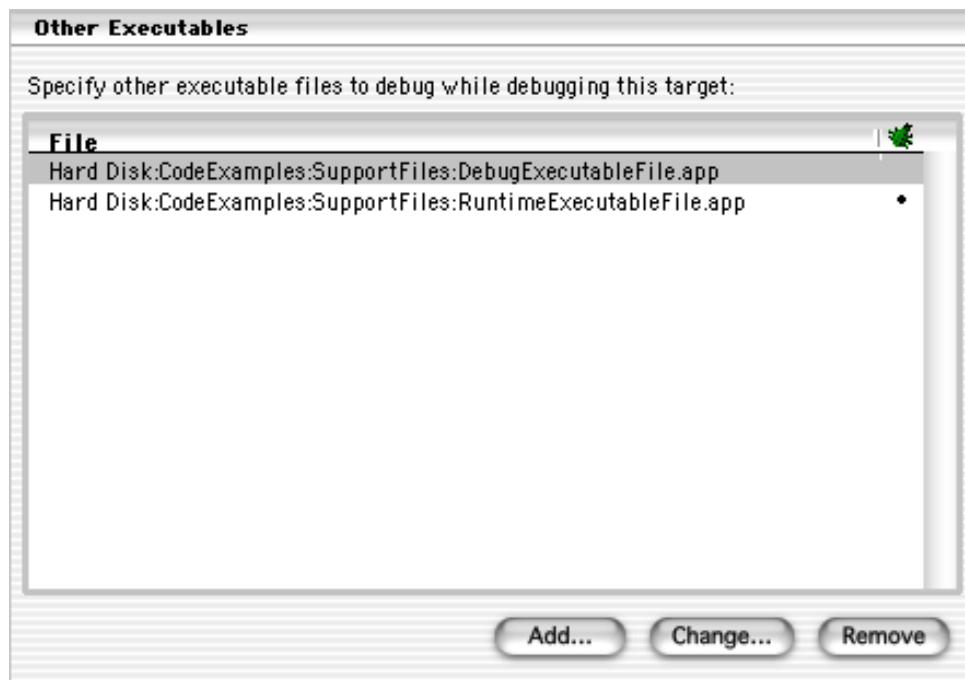
The **Debugger** section of the Target Settings Panels list defines general debugger settings for the project. Consult the *Targeting* documentation for more information about platform-specific settings panels.

The Debugger settings panels available on most IDE hosts include:

- [“Other Executables” on page 403](#)
- [“Debugger Settings” on page 406](#)
- [“Remote Debugging” on page 408](#)

## Other Executables

The **Other Executables** settings panel configures additional executable files for the IDE to debug together with the current build target.

**Figure 32.10 Other Executables settings panel****Table 32.11 Other Executables settings panel—items**

Item	Icon	Explanation
File list		Lists the executable files that the IDE can debug together with the current build target.
Debug column		Click in this column to toggle debugging of the corresponding executable file.
Add		Click to select an executable file to add to the File list.
Change		Click to change the selected entry in the File list.
Remove		Click to remove the selected entry in the File list.

## Adding an Executable File to the File List

To add an executable file to the File list, follow these steps:

1. Click **Add**.

The **Debug Additional Executable** dialog box appears.

2. Enter in the **File Location** field the path to the executable file.

Alternatively, click **Choose** to display a dialog box. Use the dialog box to select the executable file. The path to the selected executable file appears in the **File Location** field.

3. Select **Download file during remote debugging** as desired.

When selected, the IDE downloads the executable file from a remote computer during the debugging session. Enter in the corresponding field the path to the remote file. Alternatively, click **Choose** to select the file. Click **Clear** to delete the current entry.

4. Select **Debug merged executable** as desired.

When selected, the IDE debugs an executable file that merged with the project output. Enter in the corresponding field the path to the original executable file (prior to merging). Alternatively, click **Choose** to select the file. Click **Clear** to delete the current entry.

5. Click **Done**.

The IDE adds the executable file to the File list.

---

## Changing an Executable File in the File List

To change an executable file in the File list, follow these steps:

1. Select the desired path.

2. Click **Change**.

The **Debug Additional Executable** dialog box appears.

3. Modify the **File Location** field as desired.
4. Modify the **Download file during remote debugging** option as desired.
5. Modify the **Debug merged executable** option as desired.
6. Click **Done**.

The IDE modifies the executable file.

---

## **Removing an Executable File from the File List**

To remove an executable file from the File list, follow these steps:

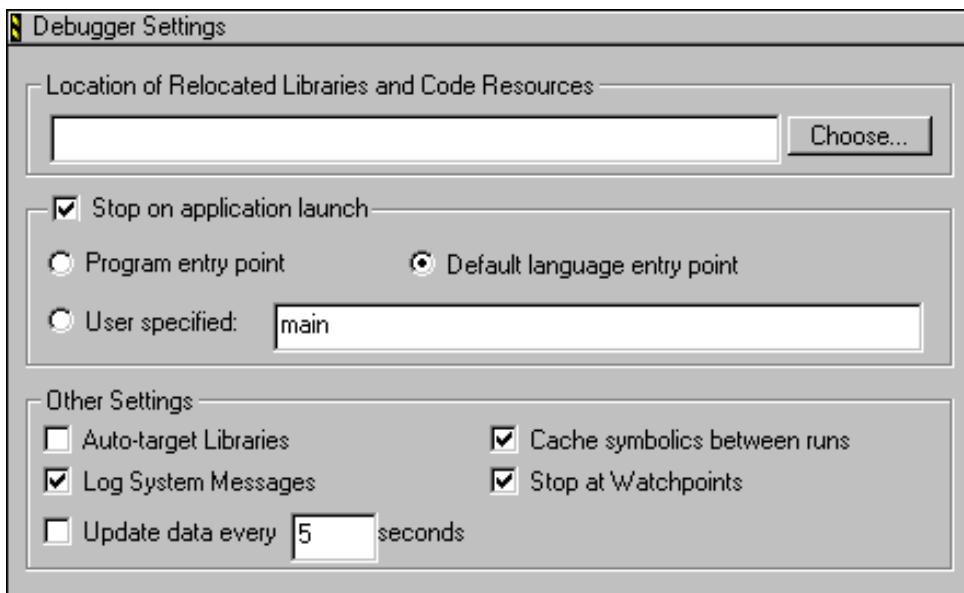
1. Select the obsolete path.
2. Click **Remove**.

The IDE removes the executable file from the File list.

## **Debugger Settings**

The **Debugger Settings** panel configures activity logs, data-update intervals, and other debugger-related options.

**Figure 32.11 Debugger Settings panel**



**Table 32.12 Debugger Settings panel—items**

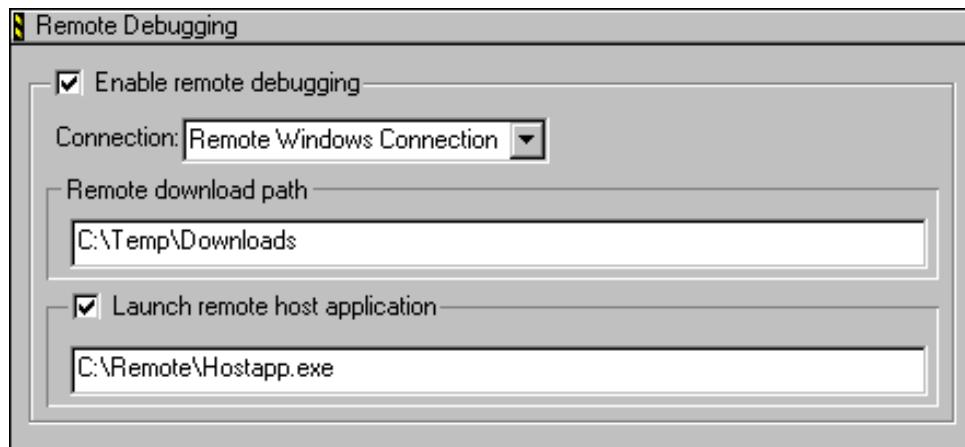
Item	Explanation
<a href="#">Location of Relocated Libraries and Code Resources</a>	Enter in this field the path to code resources or relocated libraries required for debugging the project. Alternatively, click <b>Choose</b> to select the required files.
<a href="#">Stop on application launch</a>	Select to halt program execution at the beginning of a debugging session. Select the desired stop point: <b>Program entry point</b> , <b>Default language entry point</b> , or <b>User specified</b> .
<a href="#">Program entry point</a>	Select to halt program execution upon entering the program.
<a href="#">Default language entry point</a>	Select to halt program execution upon entering a default point defined by the programming language.
User specified	Select to halt program execution at a specified function. Enter the desired function name in the corresponding field.
<a href="#">Auto-target Libraries</a>	Select to debug dynamically linked libraries (DLLs) loaded by the target application, at the expense of slower performance.
<a href="#">Cache symbolics between runs</a>	Select to have the IDE cache the symbolics information it generates for a project. Clear to have the IDE discard the information after each debugging session ends.

**Table 32.12 Debugger Settings panel—items (*continued*)**

<b>Item</b>	<b>Explanation</b>
<a href="#">Log System Messages</a>	Select to log all system messages to a Log window.
<a href="#">Stop at Watchpoints</a>	Select to halt program execution at every watchpoint. Clear to halt program execution at watchpoints with changed values.
<a href="#">Update data every n seconds</a>	Enter the number of seconds <i>n</i> to wait before updating the data displayed in debugging-session windows.

## Remote Debugging

The **Remote Debugging** settings panel configures target-specific network settings for remote-debugging connections between the host computer and other computers. Use this target-specific panel to build on the general connections defined in the **Remote Connections** preference panel of the IDE Preferences window.

**Figure 32.12 Remote Debugging settings panel****Table 32.13 Remote Debugging settings panel—items**

<b>Item</b>	<b>Explanation</b>
<a href="#">Enable remote debugging</a>	Select to define for the current build target a remote-debugging connection in terms of a general connection. Choose from the <b>Connection</b> pop-up menu the desired general connection.

**Table 32.13 Remote Debugging settings panel—items (*continued*)**

Item	Explanation
<a href="#">Remote download path</a>	Enter the path to the directory in which to store downloaded files.
<a href="#">Launch remote host application</a>	Select to launch an application on the remote computer to serve as a host application. Enter in the corresponding field the path to the remote application.

## **Working with Target Settings**

*Debugger Panels*

---

# Preference and Target Settings Options

---

Use this chapter to look up CodeWarrior™ IDE preference-panel or target-setting options and learn more about their capabilities. Option names are arranged in alphabetical order.

---

**NOTE** This chapter covers options for the core IDE preference or target-setting panels described in this manual. See the *Targeting* documentation for additional panel information.

---

## A

### Activate Browser Coloring

Select this option to activate coloring of browser symbols in editor windows. Clear the option to apply the default text color to all symbols. Click the color swatch next to a symbol to modify its color.

### Activate Syntax Coloring

Select this option to activate coloring of Comments, Keywords, Strings, and Custom Keyword Sets symbols in editor windows. Clear the option to apply the default text color to all the symbols. Click the color swatch next to a symbol to modify its color.

### Add Default

Click this button to restore the default user path or system path to the Access Paths panel.

## Always Search User Paths

This option controls the search criteria the IDE uses to find system and user files.

When

- selected—the IDE treats searching for system files (such as `#include <...>`) the same as user files (`#include "..."`).
- disabled—the IDE treats system paths differently from user paths.

## Application

Enter in this field the path to the external debugger that the IDE uses in place of the CodeWarrior debugger. Alternatively, click **Browse** to open a dialog box. Use the dialog box to select the external debugger.

## Arguments

Enter in this field command-line arguments to pass to the external debugger at the beginning of a debugging session.

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

Select this option to display runtime types for C++, Object Pascal, and SOM objects. Clear the option to display static types.

## Auto Indent

Select this option to apply automatically the same indentation as the previous line for each new line of text created by pressing Enter or Return. Clear the option to always return to the left margin for each new line of text.

## Auto-target Libraries

## Auto Target Libraries

Select this option to have the IDE attempt to debug dynamically linked libraries (DLLs) loaded by the target application. The IDE debugs the DLLs that have symbolics information.

---

This option applies to non-project debugging sessions, such as debugging an attached process.

---

<b>NOTE</b>	Selecting this option may slow IDE performance. Clear the option to improve speed.
-------------	--

---

## Automatic Invocation

Select this option to have the Code Completion window automatically open after typing specific programming-language characters in the active editor window. Clear the option to manually open the Code Completion window.

The specific characters that trigger opening of the Code Completion window depend on the programming language that you use. For example, typing a period after a Java class triggers opening of the Code Completion window, allowing you to complete the class invocation.

You can change the time it takes for the Code Completion window to appear after you type a trigger character. If you perform any activity during this delay time, you cancel opening the Code Completion window.

See also:

- [“Code Completion Delay” on page 416](#)

## Automatically launch applications when SYM file opened

Select this option to launch the application associated with an opened symbolics file. The IDE sets an implicit breakpoint at the main entry point of the application. Clear the option to open the symbolics file without launching the associated application.

[Table 33.1](#) explains how to skip launching the target program

**Table 33.1 Bypass launching the target program**

On this host...	Do this...
Windows	Press Alt while the IDE opens the symbolics file.
Macintosh	Press Option while the IDE opens the symbolics file.

**Table 33.1 Bypass launching the target program (*continued*)**

On this host...	Do this...
Solaris	Press Alt while the IDE opens the symbolics file.
Linux	Press Alt while the IDE opens the symbolics file.

---

## B

### Background

Click this color swatch to configure the background color of editor windows.

### Balance Flash Delay

Enter in this field the time, in ticks, to highlight a matching punctuation character during a **Balance while typing** check. Each *tick* represents 1/60th of a second (16.67 milliseconds).

Sample tick values include:

- 0 (zero)—disables balance flashing
- 30—the default flash value (1/2 of a second)
- 999—the maximum-flash delay value

### Balance while typing

Select this option to have the editor check for balanced parentheses, brackets, and braces in editor windows. For each closing parenthesis, bracket, or brace, the editor attempts to find the opening counterpart.

The IDE behaves differently, depending on whether it finds the counterpart:

- Found—the editor window scrolls to display the matching character, then returns to the insertion point. The **Balance Flash Delay** option determines how long the editor displays the matching character.
- Not found—the IDE beeps.

## Browser Commands

Select this option to add **Browser** menu commands to contextual menus. Clear the option to remove the commands from the contextual menus.

## Browser Path

Enter in this field a path to the browser to use for viewing IDE online help. The Netscape Navigator® browser is the default application. The PATH environment variable specifies the path to the browser.

To change the default setting, or if the IDE cannot find Netscape Navigator, enter in the **Browser Path** field a path to an alternate browser. Alternatively, click **Set** to select the path.

## Build before running

Choose from this pop-up menu the way in which the IDE handles project builds before running the compiled application:

- Always—always build projects before running them.
- Never—never build projects before running them.
- Ask—ask each time how to proceed.

---

## C

## Cache Edited Files Between Debug Sessions

Select this option to maintain a cache of edited files between debugging sessions. Use this option to debug through the original source code for files modified since the last build.

Enter in the **Maintain files in cache** field the number of days to keep the cached files. Click **Purge Cache** to delete the current cache.

See also:

- [“Maintain files in cache” on page 431](#)
- [“Purge Cache” on page 434](#)

## Cache Subprojects

Use this option to improve multi-project updating and linking. When

- selected—the IDE increases its memory requirements in order to generate symbolics information for both the build targets and the subprojects within each build target.
- cleared—the IDE does not increase its memory requirements and does not generate symbolics information.

## Cache symbolics between runs

Select this option to have the IDE maintain a cache of symbolics information generated for the project. The IDE refers to this cached symbolics information during subsequent debugging sessions. The cache improves IDE performance. Clear the option to force the IDE to discard the symbolics information at the end of each debugging session.

## Case sensitive

Select this option to have the IDE consider case when completing code. Clear the option to have the IDE ignore case.

The IDE can determine possible symbol matches according to case. For example, if you clear the **Case sensitive** option and type `str` in the active editor window, the IDE displays both `string` and `String` as possible matches. Selecting the option causes the IDE to display only `string` as a possible match.

## Close non-debugging windows

Select this option to close non-debugging windows, except for the active project window, when starting a debugging session. At the end of the debugging session, the IDE automatically re-opens the closed windows.

## Code Completion Delay

Enter in this field the number of ticks to have the IDE wait from the time you type a trigger character to the time the Code Completion window opens. A tick is 1/60 of a second.

Performing any activity during this delay time cancels opening of the Code Completion window.

See also:

- [“Automatic Invocation” on page 413](#)

## Collapse non-debugging windows

Select this option to collapse non-debugging windows when starting a debugging session. At the end of the debugging session, the IDE automatically restores the collapsed windows.

## Comments

Select the **Activate Syntax Coloring** option in order to configure this option. Use this option to configure the color of C, C++, and Java comments displayed in editor windows. The IDE then uses the chosen color for comments placed between /\* and \*/ or from // to the end of a line.

Click the color swatch next to Comments to set the color.

## Compiler

Choose from this list pop-up the desired compiler for the selected **File Type** in the **File Mappings** list. Select **None** to not associate the selected file type with any compiler.

## Compiler thread stack

Enter in this field the maximum kilobytes of stack size for the IDE to allocate to compiling and linking thread support.

The IDE threads all build processes, with compiling and linking occurring on a thread separate from the main application thread. This setting controls the compiler-thread stack size.

To avoid frequent compiler crashes, such as when building very large or complex projects, increase the default compiler-thread-stack size.

## Confirm invalid file modification dates when debugging

Select this option to keep track of source-file modification dates in a project. The IDE displays a warning message if the modification dates do not match. The message

warns of possible discrepancies between object code and source code. Clear the option to prevent the IDE from displaying the warning message.

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

Select the **Confirm “Kill Process” when closing or quitting** option to have the IDE prompt for confirmation before killing processes upon closing the Thread window or quitting the IDE. Clear the option to kill processes without prompting.

## **Context popup delay**

Enter in this field the minimum time, in ticks, to hold down the mouse button before IDE contextual menus appear. Each *tick* represents 1/60 of a second (16.67 milliseconds).

Sample tick values include:

- 0 (zero)—disables appearance of contextual menus
- 40—default popup delay value (2/3 of a second)
- 240—maximum popup delay value

---

## D

## **Debugger Commands**

Select this option to add **Debug** menu commands to IDE contextual menus. Clear the option to remove the commands from the contextual menus.

## **Default file format**

Choose from this list pop-up the default end-of-line (EOL) conventions used by the IDE to save files:

- Macintosh: <CR>
- DOS: <LF><CR>
- UNIX: <LF>

## Default language entry point

Select this option to halt program execution upon entering a default point defined by the programming language. For example, C++ defines the `main()` function as the default point.

## Default size for unbounded arrays

Enter in this field the default number of elements to display in **View Array** windows for unbounded arrays.

## Disable third party COM plugins

Select this option to prevent the IDE from loading third-party Component Object Model (COM) plugins. Clear the option to have the IDE load the plugins at start-up time.

Use this option to help troubleshoot problems with the IDE. If the problem goes away after disabling the plug-ins, then a conflict exists between the third-party plugins and the IDE plugins.

## Display deprecated items

Select this option to have the Code Completion window display obsolete programming-language items. Clear the option to have the window hide the obsolete items.

Deprecated items appear in gray text in the Code Completion window.

## Do nothing

Select this option to leave all windows in place during a debugging session.

## Do nothing to project windows

Select this option to prevent the IDE from manipulating project windows when starting a debugging session. Use this option to help debug multiple build targets or multiple projects.

## Documents

Enter in this field the number of recent documents to display in the **Open Recent** submenu.

### Don't step into runtime support code

Select this option to have the IDE bypass stepping into the Metrowerks Standard Library (MSL) runtime support code and instead directly step into your own code. Clear the option to have the IDE step into the MSL runtime setup code, then step into your own code.

### Drag and drop editing

Select this option to allow dragging and dropping of text in editor windows. Clear the option to disable drag-and-drop text editing.

### Dump internal browse information after compile

Select this option to view the raw browser information that a plug-in compiler or linker provides for the IDE. Use this option to help develop plug-ins for use with the IDE.

---

<b>NOTE</b>	After enabling the <b>Dump internal browse information after compile</b> option, compile only single files or small files. Compiling an entire project can create huge internal browser information for the IDE to display.
-------------	---

---

## E

### Edit Commands

Select this option to add **Edit** menu commands to IDE contextual menus. Clear the option to remove the commands from the contextual menus.

## Edit Language

Choose from this pop-up menu the programming language to associate with the selected file mapping. The selected language determines the syntax-color scheme. For example, choose **C/C++** to apply the appropriate syntax-color scheme for C or C++ programming-language components.

## Enable automatic Toolbar help

Select this option to display Balloon Help after resting the cursor over a toolbar button. Clear the option to prevent Balloon Help from appearing.

## Enable remote debugging

Select this option to define a remote-debugging connection specific to the current build target. Choose from the **Connection** pop-up menu the general connection to use as the basis for the target-specific connection.

## Enable Virtual Space

Use this option to configure the editor for handling spaces in different ways. When

- selected—the editor allows moving the text-insertion point past the end of a line of text, using either the arrow keys or the mouse. After moving to the desired position, begin entering text. The editor automatically inserts spaces between the former end of the line and the newly entered text.
- cleared—the editor requires manual insertion of spaces to move past the end of a line of text.

## Environment Settings

Use this section to specify environment variables to pass to your program as part of the environment parameter in your program's `main()` function, or as part of environment calls. These environment variables are only available to the target program. When your program terminates, the settings are no longer available.

---

**NOTE**

The **Environment Settings** section appears only when you develop code for a Windows build target. The section does not appear for any other build target.

---

## Export Panel

Click this button to save to an Extensible Markup Language (XML) file the current state of the active preference or settings panel.

## Extension

Enter in this field a file-name extension, such as the .c or .h , for a selected File Type in the File Mappings list. [Table 33.2 on page 422](#) lists default file-name extensions.

**Table 33.2 Default file-name extensions**

Type	Extension	Explanation
Minimum CodeWarrior Installation	.iSYM	CodeWarrior Intel® Symbols
	.mch	CodeWarrior Precompiled Header
	.mcp	CodeWarrior Project File
	.SYM	CodeWarrior Mac OS 68K Debug Symbols
	.xSYM	CodeWarrior Mac OS PPC Debug Symbols
	.dbg	CodeWarrior Debug Preferences
	.exp	Exported Symbol File
	.iMAP	CodeWarrior Link Map
	.MAP	CodeWarrior Link Map
Assembly	.a	Assembly Source File (Windows and Macintosh)
	.asm	Assembly Source File
	.dump	CodeWarrior Disassembled File

**Table 33.2 Default file-name extensions (*continued*)**

Type	Extension	Explanation
C and C++	.c++	C++ Source File
	.cc	C++ Source File
	.hh	C++ Header File
	.hpp	C++ Header File
	.i	C Inline Source File
	.icc	C++ Inline Source File
	.m	Object C Source File
	.mm	Object C++ Source File
Default C and C++	.c	C Source File
	.cp	C++ Source File
	.cpp	C++ Source File
	.h	C and C++ Header File
Default Java	.class	Java Class File
	.jar	Java Archive File
	.jav	Java Source File
	.java	Java Source File
Java	.JMAP	Java Import Mapping Dump
	.jpob	Java Constructor File
	.mf	Java Manifest File

**Table 33.2 Default file-name extensions (*continued*)**

Type	Extension	Explanation
Library	.a	(Static) Archive Library (Solaris and Linux)
	.lib	Library File
	.o	Object File (Windows and Macintosh)
	.o	Object (Relocatable) Library or Kernel Module (Solaris and Linux)
	.obj	Object File
	.pch	Precompiled Header Source File
	.pch++	Precompiled Header Source File
	.so	Shared Library (Linux)
Script	.sh	Shell Script (Linux)
	.psh	Precompile Shell Script (Linux)
	.pl	Perl Script (Linux)
Mac OS X	.dylib	Mach-O Dynamic Library
	.a	Mach-O Static Library
	.o	Mach-O Object File
	.plist	Property List

**F****Factory Settings**

Click this button to change all modified options to their default values in the current preference or settings panel.

## Failure

Choose from this pop-up menu a sound to play after a **Bring Up To Date** or **Make** operation fails.

## File Type

Enter in this field the four-character file type for the selected file mapping in the **File Mappings** list.

## Find and compare operations



A bullet in the **Find and compare operations** column, whose label appears at left, indicates that the IDE ignores matching folders for find-and-compare operations. Such operations include dragging a folder into fields in the **Find** window, or comparing folder contents.

## Find Reference using

Choose from the **Find Reference using** options an online browser application to look up references and definitions.

For example, use this option to look up documentation for language keywords:

1. Select an online browser application, such as THINK Reference, with the **Find Reference using** option.
2. Select a language keyword, such as `boolean`, in the source code.
3. Choose the **Find Reference** menu command. The IDE looks up reference information for the `boolean` keyword in the THINK Reference documentation.

Although they are not included with the CodeWarrior product, the IDE supports these online browser formats:

- Apple Help Viewer (CW manuals)
- Apple Help Viewer (Mac OS X API Ref)
- PalmQuest Reference (Palm Pilot)
- QuickView—such as Macintosh Programmer’s Toolbox Assistant (MPTA)
- THINK Reference

## Font

Choose from the **Font** options the typeface to use for displaying text in editor windows. This setting behaves in two different ways, depending on the current IDE state:

- No editor windows open—the setting modifies the default font. All editor windows take on the default font.
- Editor windows open—the setting modifies the font displayed in the frontmost editor window only. Other editor windows remain unaffected. The default font remains unchanged.

## Font preferences

Select the **Font preferences** option to remember font settings for each file in a project. Clear the option to use the default font settings every time the IDE opens each file. The **Font & Tabs** preference panel defines the default settings.

## Foreground

Use the **Foreground** option to configure the color of any text not affected by the **Activate Syntax Coloring** or **Activate Browser Coloring** options.

Click the color swatch to change the current color.

---

## G-I

## Generate Browser Data From

Choose from this pop-up menu whether the IDE generates browser data, and from what source it generates that data.

Choose from these possibilities:

- **None**—Disable browser-data generation. Certain IDE features that use browser data will be unable to work with the project, but the project's size will be smaller.
- **Compiler**—Have the IDE use the compiler to generate the browser data. If you choose this option, you must Make the project in order to generate the browser data. The IDE uses the compiler assigned to the project to generate the browser data during the build process.

- **Language Parser**—Have the IDE use the language parser to generate the browser data. Certain IDE features, such as C/C++ Code Completion, function more effectively if you choose this option. The IDE uses the language parser assigned to the project to generate the browser data.

<b>NOTE</b>	If you choose the <b>Language Parser</b> option, you can also have the IDE take into account your custom macro definitions. To do so, enter the path to your prefix file in the <b>Prefix file</b> field and the path to your macro file in the <b>Macro file</b> field.
-------------	--

## Grid Size X

Enter in the **Grid Size X** field the number of pixels to space between markings on the x-axis of the Layout Editor grid.

## Grid Size Y

Enter in the **Grid Size Y** field the number of pixels to space between markings on the y-axis of the Layout Editor grid.

## Hide non-debugging windows

Select the **Hide non-debugging windows** option to hide, but not close, non-debugging windows when starting a debugging session.

To reveal the hidden windows, do one of these tasks:

- Use the **Window** menu, or
- Double-click the names of the hidden files in the Project window, or
- Perform lookups for symbols within the hidden windows.

At the end of the debugging session, the IDE automatically reveals the hidden windows.

## Host Application for Libraries & Code Resources

The **Host Application for Libraries & Code Resources** field lets you specify a host application to use when debugging a non-executable file, such as a shared library, dynamic link library (DLL), or code resource. The application that you specify in this field is not the debugger application, but rather the application with which the non-executable file interacts.

## Host Flags

The **Host Flags** list pop-up defines the host platforms which can use the selected access path. The settings include:

- **None**—no host can use this access path.
- **All**—all hosts can use this access path.
- **Windows**—only use this path for Windows build targets.
- **Mac OS**—only use this path for Mac OS build targets.

---

**NOTE**      Multiple hosts can be selected.

---

## Import Panel

Click **Import Panel** to load the contents of a previously saved Extensible Markup Language (XML) file into the active preference or settings panel.

## Include file cache

Use the **Include file cache** option to specify the upper limit of kilobytes of memory used by the IDE for caching `#include` files and precompiled headers. The larger the value entered, the more memory the IDE uses to accelerate builds.

## Initial directory

Enter in this field the initial directory for use with the external debugger.

Alternatively, click **Browse** to open a dialog box. Use the dialog box to select the initial directory.

## Insert Template Commands

Select the **Insert Template Commands** option to display the **Insert Template** submenu in contextual menus. The submenu displays source-defined function templates. Clear to remove the submenu from the contextual menus.

---

**NOTE**      Select the **Browser Commands** option in order to select the **Insert Template Commands** option. Otherwise, the **Insert Template Commands** state has no effect.

---

## Interpret DOS and Unix Paths

This option determines how the IDE treats filenames for interface files:

- Selected—the IDE treats the backslash (\) and the forward slash (/) characters as subfolder separator characters. In the example

```
#include "sys/socks.h"
```

the IDE searches for a subfolder called `sys` that contains a `socks.h` file.

- Cleared—the IDE treats both the backslash and forward slash characters as part of the filename. Using the same example, the IDE now searches for a `sys/socks.h` filename.

---

## K-L

### Keywords

Use the **Keywords** option to configure the color of C, C++, and Java programming language's keywords displayed in editor windows when the **Activate Syntax Coloring** option is enabled. Coloring does not include macros, types, variables defined by system interface files, or variables defined in source code. Click the color swatch next to Keywords to set the color.

### Launch Editor

Enter in the **Launch Editor** field a command-line expression that specifies the third-party text editor that the CodeWarrior IDE runs to edit text files.

The IDE expands the `%file` variable of the command-line expression into the full file path. For example, to run the Emacs text editor to edit text files, enter this command-line expression:

```
runemacs %file
```

Consult the documentation provided with the third-party text editor for more information about using command lines.

## Launch Editor w/ Line #

Enter in the **Launch Editor w/ Line #** field a command-line expression that specifies the third-party text editor that the IDE runs to edit text files, and an initial line of text that the third-party editor displays upon running.

The IDE expands the %line variable of the command-line expression into an initial line of text for the third-party text editor to display. For example, to run the Emacs text editor to edit a text file, and to have the Emacs editor display the line provided to it by the IDE, enter this command-line expression:

```
runemacs %file %line
```

Consult the documentation provided with the third-party text editor for more information about using command lines.

## Launch remote host application

Select this option to launch an application on the remote computer to serve as a host application. Enter in the corresponding field the path to the remote host application.

## Left margin click selects line

-  Select the **Left margin click selects line** option to use a right-pointing cursor, shown at left, to select entire lines of text from the left margin. Clear the option to disable use of the right-pointing cursor.

With the right-pointing cursor active, click in the left margin to select the current line, or click and drag along the left margin to select multiple lines.

## Level

Choose from the **Level** options the amount of information reported for IDE plug-ins in development. This information is useful for diagnosing plug-in behavior or for viewing information about the properties of installed plug-ins.

Choose one of these levels of plug-in diagnostic information:

- **None** (default)—The IDE does not activate plug-in diagnostics or produce output.
- **Errors Only**—The IDE reports problems encountered while loading plug-ins. These problems appear in a new text file after the IDE starts up
- **All Info**—The IDE reports information for each installed plug-in, such as problems with plug-in loading, optional plug-in information, and plug-in

properties. This information appears in a new text file after the IDE starts up. The text file also contains a complete list of installed plug-ins and their associated preference panels, compilers, and linkers.

The IDE allows saving and printing the text file. Use the file as an error reference for troubleshooting plug-ins. The text file also provides suggestions for correcting general plug-in errors.

## Linker

Use the **Linker** option menu to select the linker to use with the project. The choices available are always dependent on the plug-in linkers that available to the CodeWarrior IDE.

To learn more about the linkers, see the appropriate *Targeting* manual.

## Location of Relocated Libraries and Code Resources

Enter in this field the path to the relocated libraries and code-resource files required for debugging the project. Alternatively, click **Choose** to display a dialog box. Use the dialog box to select the required files.

## Log System Messages

Select this option to have the IDE maintain a log of all system messages generated during the debugging session. The Log window displays this platform-specific information. Clear the option to disable the log.

---

## M

### Maintain files in cache

Enter in the **Maintain files in cache** text box the number of days that the IDE maintains files in the file cache.

### Menu bar layout

Choose from the **Menu bar layout** options the desired configuration of menus listed in the IDE:

---

- **Windows**—organizes the menu bar according to a typical Microsoft® Windows® arrangement
- **Macintosh**—organizes the menu bar according to a typical Apple® Mac® OS arrangement

## Minimize non-debugging windows

Select the **Minimize non-debugging windows** option to minimize non-debugging windows to a reduced size when a debugging session starts. At the end of the debugging session, the IDE automatically restores the minimized windows.

---

**NOTE**      The **Minimize non-debugging windows** option is only available in MDI mode.

---

See also:

- [“Use Multiple Document Interface” on page 444](#)

## Monitor for debugging

Choose from the **Monitor for debugging** options the specific monitor to use during debugging sessions. The IDE displays debugging windows in the selected monitor. The coordinates in parentheses identify the selected monitor in QuickDraw space.

## Move open windows to debugging monitor when debugging starts

Select the **Move open windows to debugging monitor when debugging starts** option to move all open windows to the selected debugging monitor after a debugging session starts. At the end of the debugging session, the IDE restores the moved windows to their original positions.

---

**O**

## **Open windows on debugging monitor during debugging**

Select the **Open windows on debugging monitor during debugging** option to display on the debugging monitor any window that opens during the debugging session.

The IDE does not save the positions of windows closed on the debugging monitor during the debugging session. This behavior prevents window positions from gravitating to the debugging monitor.

## **Output Directory**

Use the **Output Directory** caption to show the location the IDE places a final linked output file. The default location is the directory that contains your project file. Use the **Choose** control to specify the location path.

---

**P**

## **Play sound after ‘Bring Up To Date’ & ‘Make’**

Select the **Play sound after ‘Bring Up To Date’ & ‘Make’** option to play a sound after a build operation completes. Choose different sounds for successful and unsuccessful builds using the **Success** and **Failure** pop-up options, respectively.

See also:

- “[Failure](#)” on page 425
- “[Success](#)” on page 440

## **Post-linker**

Use the **Post-linker** option to select a post-linker that performs additional work (such as format conversion) on the final executable file.

---

For more information see the appropriate *Targeting* manual.

## **Pre-linker**

Use the **Pre-linker** option to select a pre-linker that performs additional work on the object code in a project. This work takes place before the IDE links the object code into the final executable file.

For more information about the pre-linkers available, see the build targets *Targeting* manual.

## **Program Arguments**

Use the **Program Arguments** field to enter command-line arguments to pass to the project at the beginning of a debugging session. Your program receives these arguments after you choose **Project > Run**.

## **Program entry point**

Select this option to halt program execution upon entering the program.

## **Projects**

Enter in this field the number of recent projects to display in the **Open Recent** submenu.

## **Project Commands**

Select the **Project Commands** option to add **Project** menu commands to contextual menus. Clear the option to remove the commands from the contextual menus.

## **Project operations**



A bullet in the **Project operations** column, whose label appears at left, indicates that the IDE ignores matching folders for project operations. Such operations include dragging a folder into the Project window, building a project, or searching access paths after choosing **File > Open**.

## **Purge Cache**

Click **Purge Cache** to delete the contents of the current file cache.

## R

### Recommended

Select the **Recommended** option to allow the number of concurrent compiles suggested by the IDE. This suggestion takes into account the number of active Central Processing Units (CPUs) on the host computer.

### Regular Expression

Enter in the **Regular Expression** field a text pattern to match against folder names. The IDE excludes matching folders and their contents from selected project operations or find-and-compare operations.

### Relaxed C popup parsing

Use the **Relaxed C popup parsing** option to control the strictness of C coding conventions:

- Select the option to have the IDE recognize some non-standard functions that interfere with Kernighan-and-Ritchie conventions. The IDE displays the non-standard functions in the **Routine** list pop-up.
- Clear the option to have the IDE recognize only functions that conform to Kernighan-and-Ritchie conventions. The IDE displays only the standard functions in the **Routine** list pop-up.

For more information, refer to “Reference Manual,” of *The C Programming Language, Second Edition*, by Kernighan and Ritchie, published by Prentice Hall.

---

<b>NOTE</b>	Toggle the <b>Relaxed C popup parsing</b> option to maximize recognition of functions, macros, and routine names in the source code.
-------------	--

---

### Remote download path

Enter in this field the path to the directory in which to store files downloaded from the remote host application.

## Require Framework Style Includes

This option determines the strictness with which the IDE treats `#include` statements for frameworks:

- selected—the IDE requires the framework in addition to the referenced header file. In the example

```
#include <Cocoa/CocoaHeaders.h>
```

the IDE requires the presence of `Cocoa/` in order to find the `CocoaHeaders.h` file.

- cleared—the IDE requires only the referenced header file. Using the same example, `Cocoa/` becomes optional.

## Revert Panel

Click **Revert Panel** to revert all modified options in the current preference or settings panel to the values present when the panel was originally opened.

---

## S

### Save open files before build

Select the **Save open files before build** option to automatically save files during project operations:

- Preprocess
- Precompile
- Compile
- Disassemble
- Bring Up To Date
- Make
- Run

### Save project entries using relative paths

Use the **Save project entries using relative paths** option to store the location of a file using a relative path from one of the access paths. The settings include:

- **enabled**—the IDE stores extra location information to distinctly identify different source files with the same name. The IDE remembers the location information even if it needs to re-search for files in the access paths.
- **disabled**—the IDE remembers project entries only by name. This setting can cause unexpected results if two or more files share the same name. In this case, re-searching for files could cause the IDE to find the project entry in a different access path.

## Script

Choose from the **Scripts** options the script system (language) used to display text in editor windows. This setting behaves in two different ways, depending on the current IDE state:

- No editor windows open—the setting modifies the default script system. All editor windows take on the default script system.
- Editor windows open—the setting modifies the script system displayed in the frontmost editor window only. Other editor windows remain unaffected. The default script system remains unchanged.

## Select stack crawl window when task is stopped

Select the **Select stack crawl window when task is stopped** option to automatically bring the Thread window to the foreground after the debugger stops a task. Clear the option to leave the Thread window in its previous position.

This option is useful for watching variable values change in multiple Variable windows as the debugger steps through code.

## Selection position

Select the **Selection position** option to remember these items for each editor window:

- visible text
- insertion-point location
- selected text

Clear the option to open each editor window according to default settings and place the insertion point at the first line of text.

<b>NOTE</b>	The IDE must be able to write to the file in order to remember selection position.
-------------	--

## Show all locals

Select the **Show all locals** option to display all local variables in Variable windows. Clear the option to show only variables near the program counter.

The Variables pane uses these display settings:

- **Variables: All**—shows all local variables in the code.
- **Variables: Auto**—only shows the local variables of the routine to which the current-statement arrow currently points.
- **Variables: None**—does not show variables. Use this setting to improve stepping performance for slow remote connections.

## Show message after building up-to-date project

Select the **Show message after building up-to-date project** option to have the IDE display a message after building an up-to-date project.

## Show tasks in separate windows

Select the **Show tasks in separate windows** option to open a separate Thread window for each task. Clear the option to use one Thread window to display multiple tasks. When cleared, use the Task list pop-up at the top of the Thread window to choose a task to display.

## Show the component palette when opening a form

Select the **Show the component palette when opening a form** option to automatically display the Component Palette after opening a form in the Layout Editor. Clear the option to require manual opening of the Component Palette.

## Show the object inspector when opening a form

Select the **Show the object inspector when opening a form** option to automatically open an Object Inspector window when opening a layout in the Layout Editor. Clear the option to require manual opening of the Object Inspector.

## Show values as decimal instead of hex

Select the **Show values as decimal instead of hex** option to display variable values in decimal form. Clear the option to display the values in hexadecimal form.

## Show variable location

Select the **Show variable location** option to display the **Location** column in the Variables pane of the Thread window. Clear the option to hide the **Location** column.

## Show variable types

Select the **Show variable types** option to display the type associated with each variable in Variable windows. Clear the option to hide the variable types.

## Show variable values in source code

Select the **Show variable values in source code** option to show current values for variable names displayed in contextual menus. Clear the option to show variable names only.

## Size

Choose from the **Size** options the font size used to display text in editor windows. This setting behaves in two different ways, depending on the current IDE state:

- No editor windows open—the setting modifies the default font size. All editor windows take on the default font size.
- Editor windows open—the setting modifies the font size displayed in the frontmost editor window only. Other editor windows remain unaffected. The default font size remains unchanged.

## Sort functions by method name in symbolics window

Select the **Sort functions by method name in symbolics window** option to alphabetically sort functions by method name. Clear the option to alphabetically sort by class name. The sorting affects functions of the form `className::methodName` that appear in the Symbolics window.

Since most C++ and Java source files contain methods that belong to the same class, select the option to simplify selection of functions by typing method names.

## Stop at Watchpoints

Select this option to halt program execution at every watchpoint, regardless of whether the watchpoint value changed. Clear the option to halt execution at watchpoints with changed values.

## Stop on application launch

Select this option to halt program execution at a specified point each time a debugging session begins.

## Strings

Use the **Strings** option to configure the color of anything that is not a comment, keyword, or custom keyword and displayed in editor windows when the **Activate Syntax Coloring** option is enabled. Sample strings include literal values, variable names, routine names, and type names.

Click the color swatch next to Strings to set the color.

## Sort function popup

Select the **Sort function popup** option to sort function names by alphabetical order in list pop-ups. Clear the option to sort function names by order of appearance in the source file.

## Success

Choose from the **Success** options a sound to play after a **Bring Up To Date** or **Make** operation succeeds.

## Symbolics

Enter in this field the number of recent symbolics files to display in the **Open Recent** submenu.

## System Paths

Click the **System Paths** radio button to display the System Paths pane in the Access Paths preference panel.

Supported hosts:

- Windows: available.
  - Macintosh: not available.
- 

## T

### Tab indents selection

Use the **Tab indents selection** option to control how the editor inserts tabs into the currently selected lines of text:

- Select the option so that pressing Tab causes the editor to insert tab characters in front of each selected line of text. The editor thereby indents the selected text.
- Clear the option so that pressing Tab causes the editor to replace selected text with a tab character. The editor thereby overwrites the selected text.

### Tab Inserts Spaces

Select the **Tab Inserts Spaces** option to have the editor insert spaces instead of tab characters into text. Clear the option to have the editor use tab characters.

The **Tab Size** option determines the number of spaces inserted by the editor.

### Tab Size

Enter in the **Tab Size** field the number of spaces to substitute in place of a tab character in text. This number applies to the **Tab Inserts Spaces** option.

### Target Name

Use the **Target Name** text box to set or modify the name of the current build target. This name appears in the Targets view in the Project window. This name is not the name assigned to the final output file, that is set in the Linker panel for the build target.

### Type

Choose from the **Type** options the desired source-tree path type:

- **Absolute Path**—This source-tree type is based on a file path.
-

- **Environment Variable**—This source-tree type is based on an existing environment-variable definition. The Macintosh-hosted IDE cannot create or modify this source-tree type.
  - **Registry Key**—This source-tree type is based on an existing Windows registry key entry.
- 

## U

### Update data every *n* seconds

Select this option to update the information displayed in debugging-session windows after a specified time interval. Enter in the field the number of seconds *n* to elapse before the next update. Clear this option to prevent data updates and keep the same window information throughout the debugging session.

### Use Concurrent Compiles

Select the **Use Concurrent Compiles** option to run more than one compiling processes at a time. Concurrent compiling makes better use of available processor capacity by allowing the operating system to optimize resource utilization, such as taking advantage of over-lapped input/output.

Both single- and multi-processor systems benefit from enabling concurrent compiles. On multiprocessor systems, the speed-up is significant.

### Use Debugging Monitor

Select the **Use Debugging Monitor** option to view debugging windows on a second monitor after a debugging session starts. This option only appears when the second monitor is connected to the computer.

### Use default workspace

Select this option to have the IDE use the default workspace. The IDE uses the default workspace to save and restore window and debugging states from one session to the next.

For example, if you select this option and close the IDE with a project window visible onscreen, that project window reappears the next time you start the IDE.

Clear this option to have the IDE start with the same default state for each new session: no windows visible onscreen.

For example, if you clear this option and close the IDE with a project window visible onscreen, that project window does not appear the next time you start the IDE. Instead, the IDE always starts without opening any windows.

## Use External Debugger

Select this option to have the IDE use an external debugger application in place of the CodeWarrior debugger.

## Use External Editor

Select the **Use External Editor** option to use an external text editor to modify text files in the current project. Clear the option to use the text editor included with the IDE.

## Use Local Project Data Storage

Select the **Use Local Project Data Storage** option to store on the host computer data associated with a project file on a read-only volume. Clear the option to store project data inside the same folder as the project file itself.

After loading a project file, the IDE creates or updates an associated project data folder. The IDE stores intermediate project data in this folder. When building or closing a project, the IDE uses the information in the project data folder to update the project file.

By default, the IDE places the project data folder within the same folder as the project file. However, the IDE cannot create or update a project data folder in a location that grants read-only privileges.

## Use modification date caching

Use the **Use modification date caching** option to determine whether the IDE checks the modification date of each project file prior to making the project. The settings include:

- **enabled**—the IDE caches the modification dates of the files in a project. At compilation time, the IDE refers to this cache to determine whether a specific file should be recompiled. This can shorten compilation time significantly for large projects.

- **disabled**—the IDE checks every file at each recompile of the project. Use this setting if using third-party editors to ensure that the IDE checks every file at compilation time.

## **Use Multiple Document Interface**

Configure the **Use Multiple Document Interface** option to change the interface style used by the IDE. The choices include:

- **MDI** (Multiple Document Interface)—In this interface style, the IDE uses a main application window. Multiple IDE document windows appear inside the main application window. Select the option to use this style.
- **FDI** (Floating Document Interface)—In this interface style, the IDE does not use a main application window. Multiple IDE document windows appear above the desktop. Clear the option to use this interface style.

## **Use multiple undo**

Select the **Use multiple undo** option to remember several undo and redo operations in editor windows. Clear the option to remember only the most recent undo or redo action.

The IDE stores undo and redo actions on a stack in first-in last-out (FILO) order, however, the stack size and capability are limited. For example, assume there are five undo actions on the stack (ABCDE). If the IDE redoes two actions (ABC), then performs a new action (ABC<sub>F</sub>), the undo events (DE) are no longer available.

## **Use Script menu**

Select the **Use Script menu** option to display the **Scripts** menu in the IDE menu bar. Clear the option to remove the Scripts menu from the menu bar. The Scripts menu provides convenient access to IDE scripts.

For more information about scripting the IDE, refer to the *CodeWarrior Scripting Reference*.

## **Use Third Party Editor**

Select the **Use Third Party Editor** option to use a third-party text editor to modify text files. Clear the option to use the text editor included with the IDE.

Enter in the **Launch Editor** and **Launch Editor w/ Line #** fields command-line expressions that specify information that the IDE passes to the third-party editor.

Consult the documentation provided with the third-party text editor for more information about using command lines.

See also:

- [“Launch Editor” on page 429](#)
- [“Launch Editor w/ Line #” on page 430](#)

## Use ToolServer menu

Select the **Use ToolServer menu** option to display the **ToolServer** menu in the IDE menu bar. Clear the option to remove the ToolServer menu from the menu bar.

## User Paths

Click this radio button to display the **User Paths** pane in the **Access Paths** preference panel.

## User Specified

Select the **User Specified** option to stipulate the number of concurrent compiles to allow in the IDE. Enter the desired number in the text box beside the option.

---

<b>NOTE</b>	The IDE accommodates a maximum of 1024 concurrent compiles. However, there is a point where the host system becomes compute-bound, and allowing more processes only adds overhead. For a single-processor system, the practical limit is approximately 12 concurrent compiles.
-------------	--

---

---

## V

## Value

The **Value** text box defines the value of the variable defined in the **Variable** text box that will be passed to a host application when control is transferred to it by the IDE.

## Variable

The **Variable** text box defines the name of a variable to be passed to a host application when control is transferred to it by the IDE.

## Variable values change

Use the **Variable values change** option to configure the color of changed variables that appear in debugger windows. Click the color swatch to change the current color.

## VCS Commands

Select the **VCS Commands** option to add VCS menu commands to contextual menus. Clear the option to remove the commands from the contextual menus.

Refer to the documentation that came with the version control system to learn about using it with the CodeWarrior IDE.

---

## W-Z

### Watchpoint indicator

Use the **Watchpoint indicator** option to configure the color of watchpoints that appear in debugger windows. Click the color swatch to change the current color.

### Window follows insertion point

Select this option to have the Code Completion window follow the insertion point as you edit text in the active editor window. Clear the option to leave the Code Completion window in place.

### Window position and size

Select the **Window position and size** option to remember the location and dimensions of each editor window. Clear the option to open each editor window according to default settings.

<b>NOTE</b>	The IDE must be able to write to the file in order to remember window position and size.
-------------	--

## Working Directory

Enter in this field the path to the default directory to which the current project has access. Debugging occurs in this location. If this field is blank, debugging occurs in the same directory as the executable file.

## Workspaces

Enter in this field the number of recent workspace files to display in the **Open Recent** submenu.

## Zoom windows to full screen

Use the **Zoom windows to full screen** option to configure the behavior of the zoom box in the upper right-hand corner of all editor windows:

- Select the option to have the IDE resize a zoomed window to fill the entire screen.
- Clear the option to have the IDE resize a zoomed window to its default size.



# Menus

---

This section contains these chapters:

- [IDE Menus](#)
- [Menu Commands](#)



# IDE Menus

---

This chapter provides an overview of CodeWarrior™ IDE menus and their commands. The IDE provides two different arrangements of IDE menus, configurable in the **IDE Extras** preference panel:

- **Windows** menu layout
- **Macintosh** menu layout

This chapter lists the IDE menus under each menu layout. For each menu, a table shows this information:

- **Menu command**—the name of each command in the menu.
- **Description**—a short description of each command.

This chapter has these sections:

- [“Windows Menu Layout” on page 451](#)
- [“Macintosh Menu Layout” on page 464](#)
- [“Rapid Application Development \(RAD\) Menus” on page 478](#)

## Windows Menu Layout

This section provides an overview of the menus and menu commands available in the **Windows** menu layout.

### File Menu

The **File** menu contains commands for opening, creating, saving, closing, and printing source files and projects. The File menu also provides different methods for saving edited files.

**Table 34.1 File menu commands**

<b>Menu command</b>	<b>Explanation</b>
<a href="#"><u>New</u></a>	Creates new projects using the New Project wizard or from project stationery files.
<a href="#"><u>Open</u></a>	Opens source and project files for editing and project modification operations.
<a href="#"><u>Find and Open File</u></a>	Opens the file specified in the Find and Open File dialog or from the selected text in the active window.  When using the Windows menu layout on a Macintosh host, hold down the Option key to change this command to <a href="#"><u>Find</u></a> .
<a href="#"><u>Close</u></a>	Closes the active window.
<a href="#"><u>Save</u></a>	Saves the active file using the editor window's filename.  When using the Windows menu layout on a Macintosh host, hold down the Option key to change this command to <a href="#"><u>Save All</u></a> .
<a href="#"><u>Save All</u></a>	Saves all open editor windows.  When using the Windows menu layout on a Macintosh host, hold down the Option key to substitute this command for the <a href="#"><u>Save</u></a> command.
<a href="#"><u>Save As</u></a>	Saves a copy of the active file under a new name and closes the original file.
<a href="#"><u>Save A Copy As</u></a>	Saves a copy of the active file without closing the file.
<a href="#"><u>Revert</u></a>	Discards all changes made to the active file since the last save operation.
<a href="#"><u>Open Workspace</u></a>	Opens a workspace that you previously saved.
<a href="#"><u>Close Workspace</u></a>	Closes the current workspace. (You cannot close the default workspace.)
<a href="#"><u>Save Workspace</u></a>	Saves the current state of onscreen windows, recent items, and debugging.
<a href="#"><u>Save Workspace As</u></a>	Saves an existing workspace under a different name.
<a href="#"><u>Import Components</u></a>	Imports the components from another catalog into the current catalog.
<a href="#"><u>Close Catalog</u></a>	Closes the current catalog and its associated Catalog Components window and Component Palette.

**Table 34.1 File menu commands (*continued*)**

Menu command	Explanation
<a href="#">Import Project</a>	Imports a project file previously saved in extensible markup language format (XML) and converts it into project file format.
<a href="#">Export Project</a>	Exports the active project file to disk in extensible markup language (XML) format.
<a href="#">Page Setup</a>	Displays the Page Setup dialog for setting paper size, orientation, and other printer options.
<a href="#">Print</a>	Displays the Print dialog for printing active files, and the contents of Project, Message, and Errors & Warning window contents.
<a href="#">Open Recent</a>	Displays a submenu of recently opened files and projects that can be chosen to open in the IDE.
<a href="#">Exit</a>	Quits the CodeWarrior IDE.  When using the Windows menu layout on a Macintosh host, this command does not appear. Instead, use the <a href="#">Quit</a> command in the File menu or the <a href="#">Quit CodeWarrior</a> command in the CodeWarrior menu.

## Edit Menu

The **Edit** menu contains all the customary editing commands, along with some CodeWarrior additions. This menu also includes the commands that open the Preferences and Target Settings windows.

**Table 34.2 Edit menu commands**

Menu command	Explanation
<a href="#">Undo</a>	Undoes the action of the last cut, paste, clear, or typing operation.  If you cannot undo the action, this command changes to <b>Can't Undo</b> .
<a href="#">Redo</a>	Redoes the action of the last Undo operation.  If you cannot redo the action, this command changes to <b>Can't Redo</b> .
<a href="#">Cut</a>	Removes the selected text and places a copy of it on the Clipboard.
<a href="#">Copy</a>	Copies the selected text and places a copy of it on the Clipboard.

**Table 34.2 Edit menu commands (*continued*)**

<b>Menu command</b>	<b>Explanation</b>
<a href="#">Paste</a>	Places the contents of the Clipboard at current insertion point or replaces the selected text.
<a href="#">Delete</a>	Removes the selected text without placing a copy on the Clipboard.  When using the Windows menu layout on a Macintosh host, this command does not appear. Instead, use the <a href="#">Clear</a> command.
<a href="#">This command saves a copy of an existing workspace. Use this command to save the workspace under a different name.</a> <a href="#">Select All</a>	Selects all the text in the current editor window or text box for cut, copy, paste, clear, or typing operations.
<a href="#">Balance</a>	Selects the text between the nearest set of parenthesis, braces, or brackets.
<a href="#">Shift Left</a>	Moves the selected text one tab stop to the left.
<a href="#">Shift Right</a>	Moves the selected text one tab stop to the right.
<a href="#">Get Previous Completion</a>	Shortcut for selecting the previous item that appears in the Code Completion window.
<a href="#">Get Next Completion</a>	Shortcut for selecting the next item that appears in the Code Completion window.
<a href="#">Complete Code</a>	Opens the Code Completion window.
<a href="#">Preferences</a>	Opens the IDE Preferences window where you can set general IDE, editor, debugger, and layout options.
<a href="#">Target Settings</a> (the name changes, based on the name of the active build target)	Opens the project's Target Settings window where you can set target, language, code generation, linker, editor, and debugger options.
<a href="#">Version Control Settings</a>	Opens the VCS Settings window to enable the activation of a version control system and its relevant settings.
<a href="#">Commands &amp; Key Bindings</a>	Opens the Customize IDE Commands window where you can create, modify, remove menus, menu commands, and key bindings.

## View Menu

The **View** menu contains commands for viewing toolbars, RAD windows, the class browser, the Message window, and debugging windows.

**Table 34.3 View menu commands**

Menu command	Explanation
<a href="#">Toolbars</a>	Use the Toolbars menu to show, hide, reset, and clear window and main toolbars.
<a href="#">Component Catalog</a>	Opens or brings to the front a Component Catalog window.
<a href="#">Component Palette</a>	Opens or brings to the front a Component Palette.
<a href="#">Object Inspector</a>	Opens or brings to the front an Object Inspector window. Use to view or modify object's properties.
<a href="#">Project Inspector</a>	Opens or brings to the front a Project Inspector window.
<a href="#">Browser Contents</a>	Opens or brings to the front a Browser Contents window.
<a href="#">Class Browser</a>	Opens or brings to the front a New Class Browser window.
<a href="#">Class Hierarchy</a>	Opens or brings to the front a Class Hierarchy window.
<a href="#">Build Progress</a>	Opens the Build Progress window.
<a href="#">Errors &amp; Warnings</a>	Opens or brings to the front an Errors & Warnings window.
<a href="#">Symbolics</a>	Opens the Symbolics window.
<a href="#">Processes</a>	Opens or brings to the front a Processes window.
<a href="#">Breakpoints</a>	Opens or brings to the front a Breakpoints window. Use to view, create, modify, and remove breakpoints.
<a href="#">Watchpoints</a>	Opens or brings to the front a Watchpoints window. Use to view, create, modify, and remove watchpoints.
<a href="#">Registers</a>	Opens or brings to the front a Register window.
<a href="#">Expressions</a>	Opens or brings to the front an Expressions window. Use to view, create, modify, and remove expressions.
<a href="#">Global Variables</a>	Opens or brings to the front a Global Variables window.

## Search Menu

The **Search** menu contains commands for finding text, replacing text, comparing files, and navigating code.

**Table 34.4 Search menu commands**

<b>Menu command</b>	<b>Explanation</b>
<a href="#">Find</a>	Opens the Find and Replace window for performing searches in the active editor window.
<a href="#">Replace</a>	Opens the Find and Replace window for replacing text in the active editor window.
<a href="#">Find in Files</a>	Opens the Find in Files window for performing searches in the active editor window.
<a href="#">Find Next</a>	Finds the next occurrence of the find string in the active editor window.  When using the Windows menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#">Find Previous</a> .
<a href="#">Find In Next File</a>	Finds the next occurrence of the find string in the next file listed in the Find window's File Set.  When using the Windows menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#">Find In Previous File</a> .
<a href="#">Enter Find String</a>	Replaces the Find text box string with the selected text.  When using the Windows menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#">Enter Replace String</a> .
<a href="#">Find Selection</a>	Finds the next occurrence of the selected text in the active editor window.  When using the Windows menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#">Find Previous Selection</a> .
<a href="#">Replace Selection</a>	Replaces the replace string in the Replace text box with the selected text.
<a href="#">Replace and Find Next</a>	Replaces the selected text with the Replace text box string, then performs a Find Next operation.  When using the Windows menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#">Replace and Find Previous</a> .

**Table 34.4 Search menu commands (*continued*)**

Menu command	Explanation
<a href="#">Replace All</a>	Finds all matches of the Find text box string and replaces them with the Replace text box string.
<a href="#">Find Definition</a>	Searches for the definition of the routine name selected in the active editor window using the project's source files.  When using the Windows menu layout on a Macintosh host, hold down the Option key to change this command to <a href="#">Find Reference</a> .
<a href="#">Go Back</a>	Returns to the previous CodeWarrior browser view.
<a href="#">Go Forward</a>	Moves to the next CodeWarrior browser view.
<a href="#">Go to Line</a>	Opens the Go To Line dialog where you can specify by line number where to position the text insertion point.
<a href="#">Compare Files</a>	Opens the Compare Files Setup window where you can choose to compare folders or files and merge their contents.
<a href="#">Apply Difference</a>	Adds, removes, or changes the selected text in the destination file to match the selected text in the source file.
<a href="#">Unapply Difference</a>	Reverses the modifications made to the destination file by the Apply Difference command.

## Project Menu

The **Project** menu contains commands for manipulating files, handling libraries, compiling projects, building projects, and linking projects.

**Table 34.5 Project menu commands**

Menu command	Explanation
<a href="#">Add Window</a>	Adds the active window to the project.
<a href="#">Add Files</a>	Opens a dialog box that you can use to add multiple files to the active project.
<a href="#">Create Group</a>	Opens the Create Group dialog box that you can use to add a new file group to the active project. The new file group appears below the selected file or group.
<a href="#">Create Target</a>	Opens the Create Target dialog box that you can use to add a new build target to the active project. The new build target appears below the selected build target.

**Table 34.5 Project menu commands (*continued*)**

Menu command	Explanation
<a href="#">Create Segment</a> or <a href="#">Create Overlay</a>	Opens the Create Segment/Overlay dialog box that you can use to add a new segment or overlay to the active project. The new segment or overlay appears below the selected one.
<a href="#">Create Design</a>	Opens the Create New Design dialog box that you can use to add a design to the active project. The new design appears in the <b>Design</b> tab of the project window.
<a href="#">Export Project as GNU Makefile</a>	Exports the current project to a GNU makefile.  When using the Windows menu layout on a Macintosh host, this command does not appear.
<a href="#">Check Syntax</a>	Checks the active editor window or selected files in the project window for compilation errors.
<a href="#">Preprocess</a>	Preprocesses the active editor window or selected files in the project window and displays the results in a new editor window.
<a href="#">Precompile</a>	Precompiles the active editor window or selected files in the project window and stores the results in a new header file.
<a href="#">Compile</a>	Compiles the active editor window or selected files in the project window.
<a href="#">Disassemble</a>	Disassembles the active editor window or selected files in the project window and displays the results in a new editor window.
<a href="#">Bring Up To Date</a>	Compiles all marked or modified files in the current build target of the active project.
<a href="#">Make</a>	Compiles and links all marked or modified files in the current build target of the active project, saving the executable file.
<a href="#">Stop Build</a>	Stops the current compile and linking operation and cancels the remainder of the build process.
<a href="#">Remove Object Code</a>	Removes the object code from one or more build targets in the project.  When using the Windows menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#">Remove Object Code &amp; Compact</a> .
<a href="#">Re-search for Files</a>	Resets the cached locations of source files using the project access paths, and storing them for faster builds and project operations.
<a href="#">Reset Project Entry Paths</a>	Resets the location of all source files in the active project using the project access paths.

**Table 34.5 Project menu commands (*continued*)**

Menu command	Explanation
<a href="#">Synchronize Modification Dates</a>	Updates the modification dates of all source files in the active project.
<a href="#">Debug</a> or <a href="#">Resume</a>	Compiles and links all marked or modified files in the current build target of the active window, then runs the built executable file.
<a href="#">Run</a>	Compiles and links all marked or modified files in the current build target of the active window, then runs the built executable file.
<a href="#">Set Default Project</a>	Uses the Set Default Project menu to choose the default project when more than one project is open in the IDE.
<a href="#">Set Default Target</a>	Uses the Set Default Target menu to choose the default build target when more than one build target is present in the project file.

## Debug Menu

The **Debug** menu contains commands for managing program execution.

**Table 34.6 Debug menu commands**

Menu command	Explanation
<a href="#">Break</a>	Pauses execution of the program in a debugging session to enable examination of register and variable contents
<a href="#">Kill</a>	Terminates the current debugging session returning control to the IDE.
<a href="#">Restart</a>	Terminates the current debugging session, then restarts the program from the beginning.
<a href="#">Step Over</a>	Executes each source line in the program, treating routine calls as a single statement and stopping the program at the next line of code.
<a href="#">Step Into</a>	Executes each source line in the program, following any subroutine calls.
<a href="#">Step Out</a>	Executes each source line in the subroutine and stops the program when the routine returns to its caller.
<a href="#">Run to Cursor</a>	Sets a temporary breakpoint on the source line containing the insertion point.

**Table 34.6 Debug menu commands (*continued*)**

<b>Menu command</b>	<b>Explanation</b>
<a href="#"><u>Change Program Counter</u></a>	Opens the Change Program Counter dialog box that you can use to move the current statement arrow to an address or symbol.
<a href="#"><u>Set Breakpoint</u></a> or <a href="#"><u>Clear Breakpoint</u></a>	Sets a breakpoint on the source line containing the insertion point. Clears the breakpoint on the source line containing the insertion point.
<a href="#"><u>Set Eventpoint</u></a>	Sets an eventpoint on the source line containing the insertion point.
<a href="#"><u>Clear Eventpoint</u></a>	Clears the breakpoint on the source line containing the insertion point.
<a href="#"><u>Set/Clear Breakpoint</u></a>	Opens the Set/Clear Breakpoint dialog box that you can use for setting or clearing breakpoints by address or symbol.
<a href="#"><u>Enable Breakpoint</u></a> or <a href="#"><u>Disable Breakpoint</u></a>	Activates the disabled breakpoint on the source line containing the insertion point. De-activates the breakpoint on the source line containing the insertion point.
<a href="#"><u>Clear All Breakpoints</u></a>	Clears all breakpoints currently set in the default build target of the active project.
<a href="#"><u>Show Breakpoints</u></a> or <a href="#"><u>Hide Breakpoints</u></a>	Adds a Breakpoint Column to all project editor windows where you can set, view, or clear breakpoints. Removes the Breakpoint Column from all project editor windows.
<a href="#"><u>Set Watchpoint</u></a> or <a href="#"><u>Clear Watchpoint</u></a>	Sets a watchpoint on the source line containing the insertion point. Clears the watchpoint on the source line containing the insertion point.
<a href="#"><u>Enable Watchpoint</u></a> or <a href="#"><u>Disable Watchpoint</u></a>	Activates the disabled watchpoint on the source line containing the insertion point. De-activates the watchpoint on the source line containing the insertion point.

**Table 34.6 Debug menu commands (*continued*)**

Menu command	Explanation
<a href="#">Clear All Watchpoints</a>	Clears all watchpoints currently set in the default build target of the active project.
<a href="#">Break on C++ Exception</a>	Configures the debugger to break at <code>__throw()</code> each time a C++ exception occurs.
<a href="#">Break on Java Exceptions</a>	Use this menu to select the Java exceptions on which the debugger breaks.
<a href="#">Connect</a>	Establishes communication with an embedded device to start a debugging session.  When using the Windows menu layout on a Macintosh host, this command does not appear.

## Data Menu

The **Data** menu contains commands that control how the CodeWarrior debugger displays data values. This menu appears only during a debugging session.

**Table 34.7 Data menu commands**

Menu command	Explanation
<a href="#">Show Types</a>	Toggles the appearance of the data type on local and global variables displayed in Variable panes and Variable windows.
<a href="#">Refresh All Data</a>	Updates data displays.
<a href="#">New Expression</a>	Creates a new expression entry in the Expressions window.
<a href="#">Copy to Expression</a>	Copies the selected variable to the Expressions window.
<a href="#">View As</a>	Displays the View As dialog where the data type of the selected variable can be specified.
<a href="#">View Variable</a>	Displays the selected variable in a new Variables window.
<a href="#">View Array</a>	Displays the selected array variable in a new Arrays window.
<a href="#">View Memory</a>	Displays the selected variable in a new Memory window.
<a href="#">View Memory As</a>	Displays the View As dialog where the data type of the selected variable can be specified, then shown in a new Memory window.
<a href="#">Cycle View</a>	Toggles the data view among <a href="#">View Source</a> , <a href="#">View Disassembly</a> , <a href="#">View Mixed</a> , and <a href="#">View Raw Data</a> .

**Table 34.7 Data menu commands (*continued*)**

<b>Menu command</b>	<b>Explanation</b>
<a href="#"><u>View Source</u></a>	View data as source code.
<a href="#"><u>View Disassembly</u></a>	View data as language disassembly.
<a href="#"><u>View Mixed</u></a>	View data as source code and its disassembly.
<a href="#"><u>View Raw Data</u></a>	View data without applied formatting.
<a href="#"><u>View As Default</u></a>	Views the selected variable in the default value format.
<a href="#"><u>View As Binary</u></a>	Views the selected variable as a binary value.
<a href="#"><u>View As Signed Decimal</u></a>	Views the selected variable as a signed decimal value.
<a href="#"><u>View As Unsigned Decimal</u></a>	Views the selected variable as an unsigned decimal value.
<a href="#"><u>View As Hexadecimal</u></a>	Views the selected variable as a hexadecimal value.
<a href="#"><u>View As Character</u></a>	Views the selected variable as a character value.
<a href="#"><u>View As C String</u></a>	Views the selected variable as a C string.
<a href="#"><u>View As Pascal String</u></a>	Views the selected variable as a Pascal string.
<a href="#"><u>View As Unicode String</u></a>	Views the selected variable as a Unicode string.
<a href="#"><u>View As Floating Point</u></a>	Views the selected variable as a floating point value.
<a href="#"><u>View As Enumeration</u></a>	Views the selected variable as an enumerated value.
<a href="#"><u>View As Fixed</u></a>	Views the selected variable as a 32-bit fixed value.

## Window Menu

The **Window** menu contains commands that manipulate IDE windows.

The menu lists the names of all open file and project windows. A checkmark appears beside the active window, and an underline indicates a modified and unsaved file.

**Table 34.8 Window menu commands**

Menu command	Explanation
<a href="#">Close</a>	Closes the active window.  When using the Windows menu layout on a Macintosh host, hold down the Option key to change this command to <a href="#">Close All</a> .
<a href="#">Close All</a>	Closes all non-project windows.  When using the Windows menu layout on a Macintosh host, hold down the Option key to substitute this command for the <a href="#">Close</a> command.
<a href="#">Cascade</a>	Arranges all editor windows so that only the title bar is visible.
<a href="#">Tile Horizontally</a>	Tiles all editor windows horizontally on the screen so none overlap.
<a href="#">Tile Vertically</a>	Tiles all editor windows vertically on the screen so none overlap.
<a href="#">Save Default Window</a>	Saves the active browser windows settings and applies it to other browser windows as they are opened.

## Help Menu

The **Help** menu contains commands for accessing the IDE's online help.

**Table 34.9 Help menu commands**

Menu command	Explanation
<a href="#">CodeWarrior Help</a>	Launches a help viewer to display the CodeWarrior IDE online help. Click on a link to view a specific IDE topic.
<a href="#">Index</a>	Launches a help viewer to display a glossary of common terms used in the CodeWarrior help and manuals.
<a href="#">Search</a>	Launches a help viewer to a page for searching the CodeWarrior help and manuals.
<a href="#">Online Manuals</a>	Launches a help viewer to display a list of CodeWarrior manuals. Click on a link to view a specific manuals contents.
<a href="#">Metrowerks Website</a>	Launches a browser and automatically points you to the Metrowerks web site.
<a href="#">About Metrowerks CodeWarrior</a>	Displays the CodeWarrior IDE version and build number information.

# Macintosh Menu Layout

This section provides an overview of the menus and menu commands available in the Macintosh menu layout.

## Apple Menu

The **Apple** menu (Mac OS 9.x.x and earlier) provides access to the CodeWarrior About box, shows system applications, and lists additional items.

Select [About Metrowerks CodeWarrior](#) to display the IDE version and build-number information.

When using the Macintosh menu layout on a Windows host, this menu does not appear.

## CodeWarrior Menu

The **CodeWarrior Menu** (visible in Mac OS X only) provides access to the CodeWarrior About box, IDE preferences, and the command that quits the IDE.

When using the Macintosh menu layout on a Windows host, this menu does not appear.

**Table 34.10 Apple menu commands**

Menu command	Explanation
<a href="#">About Metrowerks CodeWarrior</a>	Displays the CodeWarrior IDE version and build number information.
<a href="#">Preferences</a>	Opens the IDE Preferences window where you can set general IDE, editor, debugger, and layout options.
<a href="#">Quit</a>	Quits the CodeWarrior IDE.

## File Menu

The **File** menu contains commands for opening, creating, saving, closing, and printing source files and projects. The File menu also provides different methods for saving edited files.

**Table 34.11 File menu commands**

Menu command	Explanation
<a href="#">New Text File</a>	Creates a new text file and displays it in a new editor window.
<a href="#">New</a>	Creates new projects using the New Project wizard or from project stationery files.
<a href="#">Open</a>	Opens source and project files for editing and project modification operations.
<a href="#">Open Recent</a>	Displays a submenu of recently opened files and projects that can be chosen to open in the IDE.
<a href="#">Find and Open File</a>	Opens the file specified in the Find and Open File dialog or from the selected text in the active window.
<a href="#">Close</a>	Closes the active window.  When using the Macintosh menu layout on a Macintosh host, hold down the Option key to change this command to <a href="#">Close All</a> .
<a href="#">Save</a>	Saves the active file using the editor window's filename.  When using the Macintosh menu layout on a Macintosh host, hold down the Option key to change this command to <a href="#">Save All</a> .
<a href="#">Save As</a>	Saves a copy of the active file under a new name and closes the original file.
<a href="#">Save A Copy As</a>	Saves a copy of the active file without closing the file.
<a href="#">Revert</a>	Discards all changes made to the active file since the last save operation.
<a href="#">Open Workspace</a>	Opens a workspace that you previously saved.
<a href="#">Close Workspace</a>	Closes the current workspace. (You cannot close the default workspace.)
<a href="#">Save Workspace</a>	Saves the current state of onscreen windows, recent items, and debugging.
<a href="#">Save Workspace As</a>	Saves an existing workspace under a different name.
<a href="#">Import Components</a>	Imports the components from another catalog into the current catalog.
<a href="#">Close Catalog</a>	Closes the current catalog and its associated Catalog Components window and Component Palette.

**Table 34.11 File menu commands (*continued*)**

Menu command	Explanation
<a href="#">Import Project</a>	Imports a project file previously saved in extensible markup language format (XML) and converts it into project file format.
<a href="#">Export Project</a>	Exports the active project file to disk in extensible markup language (XML) format.
<a href="#">Page Setup</a>	Displays the Page Setup dialog for setting paper size, orientation, and other printer options.
<a href="#">Print</a>	Displays the Print dialog for printing active files, and the contents of Project, Message, and Errors & Warning window contents.
<a href="#">Quit</a> (Classic Mac OS)	Quits the CodeWarrior IDE.

## Edit Menu

The **Edit** menu contains all of the customary editing commands, along with some CodeWarrior additions. This menu also includes the commands that open the Preferences and Target Settings windows.

**Table 34.12 Edit menu commands**

Menu command	Explanation
<a href="#">Undo</a>	Undoes the action of the last cut, paste, clear or typing operation.  If you cannot undo the action, this command changes to <b>Can't Undo</b> .
<a href="#">Redo</a>	Redoes the action of the last Undo operation.  If you cannot redo the action, this command changes to <b>Can't Redo</b> .
<a href="#">Cut</a>	Removes the selected text and places a copy of it on the Clipboard.
<a href="#">Copy</a>	Copies the selected text and places a copy of it on the Clipboard.
<a href="#">Paste</a>	Places the contents of the Clipboard at current insertion point or replaces the selected text.
<a href="#">Clear</a>	Removes the selected text without placing a copy on the Clipboard.  When using the Macintosh menu layout on a Windows host, this command does not appear. Instead, use the <a href="#">Delete</a> command.

**Table 34.12 Edit menu commands (*continued*)**

Menu command	Explanation
<a href="#">This command saves a copy of an existing workspace. Use this command to save the workspace under a different name.</a> <a href="#">Select All</a>	Selects all the text in the current editor window or text box for cut, copy, paste, clear, or typing operations.
<a href="#">Balance</a>	Selects the text between the nearest set of parenthesis, braces, or brackets.
<a href="#">Shift Left</a>	Moves the selected text one tab stop to the left.
<a href="#">Shift Right</a>	Moves the selected text one tab stop to the right.
<a href="#">Get Previous Completion</a>	Shortcut for selecting the previous item that appears in the Code Completion window.
<a href="#">Get Next Completion</a>	Shortcut for selecting the next item that appears in the Code Completion window.
<a href="#">Complete Code</a>	Opens the Code Completion window.
<a href="#">Insert Reference Template</a>	Inserts a routine template corresponding to the selected Mac OS Toolbox call in the active window.  When using the Macintosh menu layout on a Windows host, this command does not appear.
<a href="#">Preferences</a>	Opens the IDE Preferences window where you can set general IDE, editor, debugger, and layout options.
<a href="#">Target Settings</a> (the name changes, based on the name of the active build target)	Opens the project's Target Settings window where you can set target, language, code generation, linker, editor, and debugger options.
<a href="#">Version Control Settings</a>	Opens the VCS Settings window to enable the activation of a version control system and its relevant settings
<a href="#">Commands &amp; Key Bindings</a>	Opens the Customize IDE Commands window where you can create, modify, remove menus, menu commands, and key bindings.

## Search Menu

The **Search** menu contains commands for finding text, replacing text, comparing files, navigating code, and finding routine definitions.

**Table 34.13 Search menu commands**

<b>Menu command</b>	<b>Explanation</b>
<a href="#">Find and Replace</a>	Opens the Find and Replace window for performing find and replace operations on the active editor window.
<a href="#">Find in Files</a>	Opens the Find in Files window for performing searches in the active editor window.
<a href="#">Find Next</a>	Finds the next occurrence of the find string in the active editor window.  When using the Macintosh menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#"><b>Find Previous</b></a> .
<a href="#">Find In Next File</a>	Finds the next occurrence of the find string in the next file listed in the Find window's File Set.  When using the Macintosh menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#"><b>Find In Previous File</b></a> .
<a href="#">Enter Find String</a>	Replaces the Find text box string with the selected text.  When using the Macintosh menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#"><b>Enter Replace String</b></a> .
<a href="#">Find Selection</a>	Finds the next occurrence of the selected text in the active editor window.  When using the Macintosh menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#"><b>Find Previous Selection</b></a> .
<a href="#">Replace Selection</a>	Replaces the replace string in the Replace text box with the selected text.
<a href="#">Replace and Find Next</a>	Replaces the selected text with the Replace text box string, then performs a Find Next operation.  When using the Macintosh menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#"><b>Replace and Find Previous</b></a> .
<a href="#">Replace All</a>	Finds all matches of the Find text box string and replaces them with the Replace text box string.

**Table 34.13 Search menu commands (*continued*)**

Menu command	Explanation
<a href="#">Find Definition</a>	Searches for the definition of the routine name selected in the active editor window using the project's source files.  When using the Macintosh menu layout on a Macintosh host, hold down the Shift key to change this command to <a href="#">Find Definition &amp; Reference</a> .
<a href="#">Find Reference</a>	Searches for the definition of the routine name selected in the active editor window using the specified online help system.  When using the Macintosh menu layout on a Windows host, this command does not appear.
<a href="#">Go Back</a>	Returns to the previous CodeWarrior browser view.
<a href="#">Go Forward</a>	Moves to the next CodeWarrior browser view.
<a href="#">Go to Line</a>	Opens the Go To Line dialog where you can specify by line number where to position the text insertion point.
<a href="#">Compare Files</a>	Opens the Compare Files Setup window where you can choose to compare folders or files and merge their contents.
<a href="#">Apply Difference</a>	Adds, removes, or changes the selected text in the destination file to match the selected text in the source file.
<a href="#">Unapply Difference</a>	Reverses the modifications made to the destination file by the Apply Difference command.

## Project Menu

The **Project** menu contains commands for manipulating files, handling libraries, compiling projects, building projects, and linking projects.

**Table 34.14 Project menu commands**

Menu command	Explanation
<a href="#">Add Window</a> (the name changes, based on the name of the selected item)	Adds the active window to the project.
<a href="#">Add Files</a>	Opens a dialog that you can use to add multiple files to the active project.

**Table 34.14 Project menu commands (*continued*)**

Menu command	Explanation
<a href="#">Create Group</a> or <a href="#">Create Target</a>	Displays the Create Group dialog where you can add a new file group to the active project immediately after the selected file or group. Displays the Create Target dialog where you can add a new build target to the active project immediately after the selected build target.
<a href="#">Create Overlay</a> or <a href="#">Create Segment</a>	Displays the Create Overlay dialog where you can add a new memory overlay to the active project immediately after the selected overlay. Displays the Create Segment dialog where you can add a new segment to the active project immediately after the selected segment.
<a href="#">Create Design</a>	Opens the Create New Design dialog box that you can use to add a design to the active project. The new design appears in the <b>Design</b> tab of the project window.
<a href="#">Check Syntax</a>	Checks the active editor window or selected files in the project window for compilation errors.
<a href="#">Preprocess</a>	Preprocesses the active editor window or selected files in the project window and displays the results in a new editor window.
<a href="#">Precompile</a>	Precompiles the active editor window or selected files in the project window and stores the results in a new header file.
<a href="#">Compile</a>	Compiles the active editor window or selected files in the project window.
<a href="#">Disassemble</a>	Disassembles the active editor window or selected files in the project window and displays the results in a new editor window.
<a href="#">Bring Up To Date</a>	Compiles all marked or modified files in the current build target of the active project.
<a href="#">Make</a>	Compiles and links all marked or modified files in the current build target of the active project, saving the executable. file
<a href="#">Stop Build</a>	Stops the current compile and linking operation and cancels the remainder of the build process.

**Table 34.14 Project menu commands (*continued*)**

Menu command	Explanation
<a href="#">Remove Object Code</a>	Removes the object code from one or more build targets in the project.  When using the Macintosh menu layout on a Macintosh host, hold down the Option key to change this command to <a href="#"><b>Remove Object Code &amp; Compact</b></a> .
<a href="#">Re-search for Files</a>	Resets the cached locations of source files using the project access paths, and storing them for faster builds and project operations.
<a href="#">Reset Project Entry Paths</a>	Resets the location of all source files in the active project using the project access paths.
<a href="#">Synchronize Modification Dates</a>	Updates the modification dates of all source files in the active project.
<a href="#">Debug</a> or <a href="#">Resume</a>	Compiles and links all marked or modified files in the current build target of the active window, then runs the built executable file.
<a href="#">Run</a>	Compiles and links all marked or modified files in the current build target of the active window, then runs the built executable file.
<a href="#">Set Default Project</a>	Uses the Set Default Project menu to choose the default project when more than one project is open in the IDE.
<a href="#">Set Default Target</a>	Uses the Set Default Target menu to choose the default build target when more than one build target is present in the project file.

## Debug Menu

The **Debug** menu contains commands for managing program execution.

**Table 34.15 Debug menu commands**

Menu command	Explanation
<a href="#">Kill</a>	Terminates the current debugging session returning control to the IDE.
<a href="#">Restart</a>	Terminates the current debugging session, then restarts the program from the beginning.

**Table 34.15 Debug menu commands (*continued*)**

<b>Menu command</b>	<b>Explanation</b>
<a href="#"><u>Step Over</u></a>	Executes each source line in the program, treating routine calls as a single statement and stopping the program at the next line of code.
<a href="#"><u>Step Into</u></a>	Executes each source line in the program, following any subroutine calls.
<a href="#"><u>Step Out</u></a>	Executes each source line in the subroutine and stops the program when the routine returns to its caller.
<a href="#"><u>Stop</u></a>	Pauses execution of the program in a debugging session to enable examination of register and variable contents.
<a href="#"><u>Set Breakpoint</u></a> or <a href="#"><u>Clear Breakpoint</u></a>	Sets a breakpoint on the source line containing the insertion point.  Clears the breakpoint on the source line containing the insertion point.
<a href="#"><u>Set Eventpoint</u></a>	Sets an eventpoint on the source line containing the insertion point.
<a href="#"><u>Clear Eventpoint</u></a>	Clears the breakpoint on the source line containing the insertion point.
<a href="#"><u>Set/Clear Breakpoint</u></a>	Displays the Set/Clear Breakpoint dialog for setting or clearing breakpoints by address or symbol.
<a href="#"><u>Enable Breakpoint</u></a> or <a href="#"><u>Disable Breakpoint</u></a>	Activates the disabled breakpoint on the source line containing the insertion point.  De-activates the breakpoint on the source line containing the insertion point.
<a href="#"><u>Clear All Breakpoints</u></a>	Clears all breakpoints currently set in the default build target of the active project.
<a href="#"><u>Show Breakpoints</u></a> or <a href="#"><u>Hide Breakpoints</u></a>	Adds a Breakpoint Column to all project editor windows where breakpoints can be set, viewed, and cleared.  Removes the Breakpoint Column from all project editor windows.
<a href="#"><u>Set Watchpoint</u></a> or <a href="#"><u>Clear Watchpoint</u></a>	Sets a watchpoint on the source line containing the insertion point.  Clears the watchpoint on the source line containing the insertion point.

**Table 34.15 Debug menu commands (*continued*)**

Menu command	Explanation
<a href="#">Enable Watchpoint</a> or <a href="#">Disable Watchpoint</a>	Activates the disabled watchpoint on the source line containing the insertion point.  De-activates the watchpoint on the source line containing the insertion point.
<a href="#">Clear All Watchpoints</a>	Clears all watchpoints currently set in the default build target of the active project.
<a href="#">Run to Cursor</a>	Sets a temporary breakpoint on the source line containing the insertion point.
<a href="#">Change Program Counter</a>	Displays the Change Program Counter dialog where you can move the current statement arrow to an address or symbol.
<a href="#">Break on C++ Exception</a>	Configures the debugger to break at <code>__throw()</code> each time a C++ exception occurs.
<a href="#">Break on Java Exceptions</a>	Use this menu to select which Java exceptions the debugger should break on.
<a href="#">Switch to Monitor</a>	Configures the IDE to use an external debugger instead of the CodeWarrior debugger.  When using the Macintosh menu layout on a Windows host, this command does not appear.

## Data Menu

The **Data** menu contains commands that control how the CodeWarrior debugger displays data values. This menu only appears during a debugging session.

**Table 34.16 Data menu commands**

Menu command	Description
<a href="#">Show Types</a>	Toggles the appearance of the data type on local and global variables displayed in Variable panes and Variable windows.
<a href="#">Refresh All Data</a>	Updates data displays.
<a href="#">New Expression</a>	Creates a new expression entry in the Expressions window.
<a href="#">Copy to Expression</a>	Copies the selected variable to the Expressions window.

**Table 34.16 Data menu commands (*continued*)**

<b>Menu command</b>	<b>Description</b>
<a href="#">View As</a>	Displays the View As dialog where the data type of the selected variable can be specified.
<a href="#">View Variable</a>	Displays the selected variable in a new Variables window.
<a href="#">View Array</a>	Displays the selected array variable in a new Arrays window.
<a href="#">View Memory</a>	Displays the selected variable in a new Memory window.
<a href="#">View Memory As</a>	Displays the View As dialog where the data type of the selected variable can be specified, then shown in a new Memory window.
<a href="#">Cycle View</a>	Toggles the data view among <a href="#">View Source</a> , <a href="#">View Disassembly</a> , <a href="#">View Mixed</a> , and <a href="#">View Raw Data</a> .
<a href="#">View Source</a>	View data as source code.
<a href="#">View Disassembly</a>	View data as language disassembly.
<a href="#">View Mixed</a>	View data as source code and its disassembly.
<a href="#">View Raw Data</a>	View data without applied formatting.
<a href="#">View As Default</a>	Views the selected variable in the default value format.
<a href="#">View As Binary</a>	Views the selected variable as a binary value.
<a href="#">View As Signed Decimal</a>	Views the selected variable as a signed decimal value.
<a href="#">View As Unsigned Decimal</a>	Views the selected variable as an unsigned decimal value.
<a href="#">View As Hexadecimal</a>	Views the selected variable as a hexadecimal value.
<a href="#">View As Character</a>	Views the selected variable as a character value.
<a href="#">View As C String</a>	Views the selected variable as a C string.
<a href="#">View As Pascal String</a>	Views the selected variable as a Pascal string.
<a href="#">View As Unicode String</a>	Views the selected variable as a Unicode string.
<a href="#">View As Floating Point</a>	Views the selected variable as a floating point value.
<a href="#">View As Enumeration</a>	Views the selected variable as an enumerated value.
<a href="#">View As Fixed</a>	Views the selected variable as a 32-bit fixed value.
<a href="#">View As Fract</a>	Views the selected variable as a fract value.  When using the Macintosh menu layout on a Windows host, this command does not appear.

## Window Menu

The **Window** menu contains commands that manipulate IDE windows. The **Window** menu is divided into several sections:

- window commands to stack, tile, zoom, collapse, and save window positions.
- toolbar submenu for showing, hiding, resetting, and clearing window and floating toolbars.
- commands to open specific browser, IDE, and debugger windows.
- names of all open file and project windows.

A check mark appears beside the active window, and an underline denotes a modified and unsaved file.

**Table 34.17 Window menu commands**

Menu command	Description
<a href="#">Stack Editor Windows</a>	Arranges all editor windows so that only the title bar is visible.
<a href="#">Tile Editor Windows</a>	Tiles all editor windows horizontally on the screen so none overlap.
<a href="#">Tile Editor Windows Vertically</a>	Tiles all editor windows vertically on the screen so none overlap.
<a href="#">Zoom Window</a>	Restores the active editor window to its previous size and position.
<a href="#">Collapse Window (Minimize Window)</a>	Collapses the active editor window so that only its title bar is visible.  or
<a href="#">Expand Window (Maximize Window)</a>	Expands the collapsed editor window to its previous size and position.
<a href="#">Save Default Window</a>	Saves the current browser-window settings for later re-use.
<a href="#">Toolbars</a>	Use the Toolbars submenu to show, hide, reset, and clear window, main, and floating toolbars.
<a href="#">Browser Contents</a>	Opens or brings to the front a Browser Contents window.
<a href="#">Class Hierarchy Window</a>	Opens or brings to the front a Class Hierarchy window.
<a href="#">New Class Browser</a>	Opens or brings to the front a New Class Browser window.
<a href="#">Build Progress Window</a>	Opens or brings to the front a Build Progress window.
<a href="#">Errors &amp; Warnings Window</a>	Opens or brings to the front an Errors & Warnings window.
<a href="#">Project Inspector</a>	Opens or brings to the front a Project Inspector window.

**Table 34.17 Window menu commands (*continued*)**

Menu command	Description
<a href="#">ToolServer Worksheet</a>	Opens or brings to the front a ToolServer Worksheet window.  When using the Macintosh menu layout on a Windows host, this command does not appear.
<a href="#">Symbolics Window</a>	Opens or brings to the front a Symbolics window.
<a href="#">Processes Window</a>	Opens or brings to the front a Processes window.
<a href="#">Expressions Window</a>	Opens or brings to the front an Expressions window. Use to view, create, modify, and remove expressions.
<a href="#">Global Variables Window</a>	Opens or brings to the front a Global Variables window.
<a href="#">Breakpoints Window</a>	Opens or brings to the front a Breakpoints window. Use to view, create, modify, and remove breakpoints.
<a href="#">Watchpoints Window</a>	Opens or brings to the front a Watchpoints window. Use to view, create, modify, and remove watchpoints.
<a href="#">Register Window</a>	Opens or brings to the front a Register window.
<a href="#">Component Catalog</a>	Opens or brings to the front a Component Catalog window.
<a href="#">Component Palette</a>	Opens or brings to the front a Component Palette.
<a href="#">Object Inspector</a>	Opens or brings to the front an Object Inspector window. Use to view or modify object's properties.

## VCS Menu

The VCS (Version Control System) menu appears in the IDE's menu bar when the **Use Version Control** option is enabled. The CodeWarrior IDE can operate with many difference version control systems including CVS, Visual SourceSafe, and others.



This icon represents the VCS menu in the Macintosh-hosted IDE menu bar.

Refer to the documentation that came with the version control system to learn about using it with the CodeWarrior IDE.

## Tools Menu

The Tools menu appears in the IDE's menu bar after you enable the **Use ToolServer menu** checkbox in the [IDE Extras](#) preference panel. The Tools menu contains

commands for controlling Apple® ToolServer™ and Macintosh Programmer’s Workbench (MPW).



Macintosh: This icon represents the Tools menu in the Macintosh-hosted IDE menu bar.

Refer to *Targeting Mac OS* to learn about using ToolServer and MPW with projects.

## Scripts Menu

The **Scripts** menu appears in the IDE’s menu bar after you enable the **Use Scripts menu** checkbox in the [IDE Extras](#) preference panel and creates a (Scripts) folder in the Metrowerks CodeWarrior folder of your CodeWarrior installation.

The Scripts menu uses the directory structure of the (Scripts) folder to create a hierarchical menu. This hierarchical menu lists all of the scripts in the (Scripts) folder. Open a script-editing utility or a text editor to learn more about the scripts that might already exist in the (Scripts) folder.

Refer to the *CodeWarrior Scripting Reference* to learn more about scripting the CodeWarrior IDE.

## Help Menu

The **Help** menu contains commands for accessing the IDE’s online help.

**Table 34.18 Help menu commands**

Menu command	Description
<a href="#">CodeWarrior Help</a>	Launches a help viewer to display the CodeWarrior IDE online help. Click on a link to view a specific IDE topic.
Index (Windows)	Opens the online help to the Index tab.
Search (Windows)	Opens the online help to the Search tab.
<a href="#">Online Manuals</a>	Launches a help viewer to display a list of CodeWarrior manuals. Click on a link to view a specific manuals contents.
<a href="#">Metrowerks Website</a>	Launches a browser and automatically points you to the Metrowerks website.
<a href="#">About Metrowerks CodeWarrior (Windows)</a>	Displays the CodeWarrior IDE version and build number information.

<b>NOTE</b>	Classic Macintosh: The Help menu contains additional menu commands for working with Balloon Help and accessing Apple's online help.
-------------	---

# Rapid Application Development (RAD) Menus

These menus are common across all platforms and appear when the RAD layout editor is active.

## Align Menu

The **Align** menu contains commands to align selected components by component edge or center properties. The Align menu is only visible when editing components within a RAD project's layout window.

**Table 34.19 Align menu commands**

Menu command	Description
<a href="#">To Grid</a>	Aligns the selected components to the grid in the layout window.
<a href="#">Left Edges</a>	Aligns the selected components by the left edges.
<a href="#">Vertical Center</a>	Aligns the selected components by their vertical centers.
<a href="#">Right Edges</a>	Aligns the selected components by the right edges.
<a href="#">Top Edges</a>	Aligns the selected components by the top edges.
<a href="#">Horizontal Center</a>	Aligns the selected components by their horizontal centers.
<a href="#">Bottom Edges</a>	Aligns the selected components by the bottom edges.

## Layout Menu

The **Layout** menu contains commands that manipulate objects in a Rapid Application Development (RAD) layout window. This menu only appears when a RAD project is open.

**Table 34.20 Layout menu commands**

Menu command	Description
<a href="#">Align</a>	Use this submenu to align selected components by component edge or center properties.
<a href="#">Resize</a>	Use this submenu to resize selected components by different heights and widths.
<a href="#">Display Grid</a>	Toggles the visibility of the grid in layout windows.
<a href="#">Snap To Grid</a>	Toggles the alignment of components with the grid in layout windows.
<a href="#">Group</a>	Combines the selected components into a single grouped object for ease of manipulation.
<a href="#">Ungroup</a>	Separates an object of grouped components into the individual components.
<a href="#">Customize</a>	Opens an editor for the selected component if one is available.
<a href="#">Properties</a>	Opens or brings to the front the Object Inspector window to enable editing of the selected component's properties.
New Wire	Opens the Create Wire wizard. Use this wizard to create a new wire in the layout.
Edit Wire	Opens the Create Wire wizard for the selected wire. Use this wizard to modify the wire details.
Show All Wires	Toggles visibility of all wires in the layout.
Show Incoming Wires	Select a component in the layout, then select this command to view all wires that define the selected component as their destination component.
Show Outgoing Wires	Select a component in the layout, then select this command to view all wires that define the selected component as their source component.

## Resize Menu

The **Resize** menu contains commands for resizing components in a layout editor to set sizes. The Resize menu is only visible when editing components within a RAD project's layout window.

**Table 34.21 Resize menu commands**

<b>Menu command</b>	<b>Description</b>
<a href="#">To Smallest Width</a>	Resizes the selected components to match the width of the component with the smallest width.
<a href="#">To Largest Width</a>	Resizes the selected components to match the width of the component with the largest width.
<a href="#">To Smallest Height</a>	Resizes the selected components to match the height of the component with the smallest height.
<a href="#">To Largest Height</a>	Resizes the selected components to match the height of the component with the largest height.

# Menu Commands

---

This section presents an alphabetical listing of all available menu commands in the CodeWarrior™ IDE. Menu commands that appear only on certain host platforms are documented. A menu command that has no host information is available on all hosts.

Use this listing as a reference to find information about a specific menu command.

---

## A

### About Metrowerks CodeWarrior

This command displays the CodeWarrior IDE version and build number information.

---

<b>TIP</b>	Click the <b>Installed Products</b> button in this window to view and save information about installed products and plug-ins for the CodeWarrior IDE. You can also use this window to enable or disable plug-in diagnostics.
------------	--

---

### Add Files

The **Add Files** command opens a dialog which allows one or more files to be added to the project.

### Add Window

The **Add Window** command adds the file in the active Editor window to the open project. The name of the menu command changes, based on the name of the active window. For example, if the name of the active window is `MyFile`, the name of the menu command changes to **Add MyFile to Project**.

## Align

Reveals the **Align** submenu with component alignment commands like Right Edges, Vertical Centers, and others.

See also:

- [“Bottom Edges” on page 483](#)
- [“Horizontal Center” on page 498](#)
- [“Left Edges” on page 499](#)
- [“Right Edges” on page 508](#)
- [“To Grid” on page 515](#)
- [“Top Edges” on page 516](#)
- [“Vertical Center” on page 517](#)

## All Exceptions

The **All Exceptions** command of the **Java** submenu to tell the debugger to break every time an exception occurs. This behavior includes exceptions thrown by the virtual machine, your own classes, the debugger, classes in `classes.zip`, and so on. Java programs throw many exceptions in the normal course of execution, so catching all exceptions causes the debugger to break often.

## Anchor Floating Toolbar

The **Anchor Floating Toolbar** command attaches the floating toolbar beneath the menu bar. Once attached, the anchored toolbar can not be moved again until it is unanchored.

See also:

- [“Unanchor Floating Toolbar” on page 516](#)

## Apply Difference

The **Apply Difference** command applies the selected difference from the source file into the destination file.

**B**

## Balance

The **Balance** command selects all the text starting at the current insertion point and enclosed in parentheses (), brackets [], or braces {},

## Bottom Edges

The **Bottom Edges** command of the **Align** submenu aligns the bottom edges of the selected components.

## Break

The **Break** command temporarily suspends execution of the target program and returns control to the debugger.

See also [“Stop” on page 513](#).

## Break on C++ Exception

The **Break on C++ Exception** command tells the debugger to break at `__throw()` each time a C++ exception occurs.

## Break on Java Exceptions

The **Break on Java Exceptions** command reveals the Java Exceptions submenu.

See also:

- [“Exceptions in Targeted Classes” on page 492](#)
- [“Uncaught Exceptions Only” on page 517](#).

## Breakpoints

### Breakpoints Window

These commands open the Breakpoints window.

## Bring To Front

The **Bring To Front** command moves the selected objects so that they are displayed in front of all other objects.

## Bring Up To Date

The **Bring Up To Date** command updates the current build target in the active project by compiling all of the build target's modified and touched files.

## Browser Contents

The **Browser Contents** command opens the Browser Contents window. This command is not available if the Enable Browser option is not activated.

## Build Progress

### Build Progress Window

These commands open the Build Progress window. Use it to monitor the IDE's status as it compiles a project.

---

## C

### Cascade

The **Cascade** command arranges open editor windows one on top of another, with their window titles visible.

### Change Program Counter

The **Change Program Counter** command opens a window that lets you move the current-statement arrow to a particular address or symbol.

---

## Check Syntax

The **Check Syntax** command checks the syntax of the source file in the active Editor window or the selected files in the open project window. If the IDE detects one or more errors, a Message window appears and shows information about the errors.

The **Check Syntax** command is not available if:

- the active Editor window is empty.
- no project file is open.

**Check Syntax** does not generate object code.

[Table 35.1](#) explains how to abort the syntax-checking process:

**Table 35.1 Aborting the syntax-checking process**

On this host...	Do this...
Windows	Press Esc.
Macintosh	Press Command-. (Command-Period).
Solaris	Press Esc.
Linux	Press Esc.

## Class Browser

The **Class Browser** command opens a Class Browser window. This command is unavailable if the **Enable Browser** option is not enabled.

## Class Hierarchy

### Class Hierarchy Window

These commands open a Multi-Class Browser window. This command is unavailable if the **Enable Browser** option is not enabled.

## Clear

The **Clear** command removes the selected text. This menu command is equivalent to pressing the Backspace or Delete key.

## Clear All Breakpoints

The **Clear All Breakpoints** command clears all breakpoints in all source files belonging to the target program.

## Clear All Watchpoints

The **Clear All Watchpoints** command clears all watchpoints in the current program.

## Clear Breakpoint

The **Clear Breakpoint** command clears the breakpoint at the currently selected line. If the **Show Breakpoints** option is enabled, the marker in the Breakpoints column of the Editor window disappears.

## Clear Eventpoint

This command opens a submenu that lets you remove an eventpoint from the currently selected line. If the **Show Breakpoints** option is active, the Breakpoints column in the editor windows shows a marker next to each line with an eventpoint. The marker represents the eventpoint type.

## Clear Floating Toolbar

The **Clear Floating Toolbar** command removes all the shortcut icons from the floating toolbar. Once the toolbar is cleared, drag shortcut icons from the Commands and Key Bindings window to the toolbar to create a custom floating toolbar.

## Clear Main Toolbar

The **Clear Main Toolbar** command removes all the shortcut icons from the main toolbar. Once the toolbar is cleared, drag shortcut icons from the Commands and Key Bindings window to the toolbar to create a custom main toolbar.

## Clear Watchpoint

The **Clear Watchpoint** command removes a watchpoint from the selected variable or memory range.

## Clear Window Toolbar

The **Clear Window Toolbar** command removes all the shortcut icons from the window toolbar. Once the toolbar is cleared, drag shortcut icons from the Commands and Key Bindings window to the toolbar to create a custom window toolbar.

## Close

The **Close** command closes the active window.

## Close All

The **Close All** command closes all open windows of a certain type. The name of this menu command changes, based on the type of item selected. For example, select one of several open editor windows, the menu command changes its name to **Close All Editor Documents**.

---

**NOTE** Macintosh: Press the Option key to change Close to Close All.

---

## Close Catalog

The **Close Catalog** command closes the current catalog and remove the catalog from the Component Catalog window and the Component Palette.

## Close Workspace

This command closes the current workspace.

You cannot close the default workspace, but you can choose whether to use it by toggling the [Use default workspace](#) option in the [IDE Extras](#) preference panel.

## Commands & Key Bindings

The **Commands and Key Bindings** command opens the Customize IDE Commands window.

## Complete Code

The **Complete Code** command opens the Code Completion window. Use this window to help you automatically complete programming-language symbols as you type them in the active editor window.

## CodeWarrior Glossary

The **CodeWarrior Glossary** command opens and display a list of vocabulary terms used by the CodeWarrior manuals and online help.

## CodeWarrior Help

This command opens the online help for the CodeWarrior IDE.

## Collapse Window

The **Collapse Window** command collapses the active window so that only its title is visible.

## Compare Files

The **Compare Files** command opens the Compare Files Setup window. Use it to choose two files or folders for comparison and merging. After choosing the items, a comparison window appears that shows the differences between the items.

## Component Catalog

The **Component Catalog** command opens a Component Catalog window for use within a RAD project.

## Compile

The **Compile** command compiles selected source files into binary files. The IDE compiles source files that are:

- part of the current project and open in the active Editor window, or
- selected files, segments, or groups in a project window.

## Component Palette

The **Component Palette** command opens a Component Palette window for use within a RAD project.

## Connect

The **Connect** command establishes communications between the IDE and embedded hardware to begin a debugging session.

## Copy

The **Copy** command copies selected text to the system Clipboard. If the Message Window is active, the Copy command copies all the text in the Message Window to the Clipboard.

## Copy to Expression

The **Copy to Expression** command copies the variable selected in the active pane to the Expressions window.

## Create Design

This command creates a new design in the current project. The new design appears in the **Design** tab of the project window. You cannot create a design if each build target in the project already belongs to a design.

## Create Group

The **Create Group** command creates a new group in the current project. This command is only active when the **Files** view is visible in the project window.

## Create Overlay

## Create Segment

These commands create a new segment or overlay in the current project. This command is only active when the **Segments** view or **Overlays** view is visible in the project window.

## Create Target

The **Create Target** command creates a new build target in the current project. This command is only active when the **Targets** view is visible in the project window.

## Customize

The **Customize** command opens a customizing utility for the selected component within a RAD project. For example, after selecting a menu component and choosing this menu command, the Menu editor opens.

## Cut

The **Cut** command copies the selected text to the system Clipboard, replacing the previous Clipboard contents, and removes it from the current document or text box.

## Cycle View

Toggles view among various data formats.

See also:

- [“View Disassembly” on page 519](#)
- [“View Mixed” on page 520](#)
- [“View Raw Data” on page 520](#)
- [“View Source” on page 520](#)

---

## D

## Debug

This command compiles and links a project, then run the CodeWarrior debugger with the project’s code. If debugging is active, The Threads window to examine program information and step through the code as it executes. If debugging is not active, the Threads window appears, but the program executes without stopping in the debugger.

## Delete

The **Delete** command removes the selected text without placing it on the system clipboard. This menu command is equivalent to pressing the Backspace or Delete key.

## Disable Breakpoint

The **Disable Breakpoint** command de-activates the breakpoint at the currently selected line.

## Disable Watchpoint

The **Disable Watchpoint** command de-activates a watchpoint for the selected variable or memory range.

## Disassemble

The **Disassemble** command disassembles the compiled source files selected in the project window. After disassembling a file, the IDE creates a .dump file that contains the file's object code. The .dump file appears in a new window after the IDE completes the disassembly process.

## Display Grid

The **Display Grid** command toggles the visibility of grid lines in the layout window. When checked, the grid lines appear, otherwise, no grid is visible.

---

## E

## Enable Breakpoint

The **Enable Breakpoint** command activates a breakpoint at the currently selected line. The breakpoint appears in the left side of the editor window if the Breakpoint column is visible. The states of the breakpoint marker include:

- ● enabled breakpoint.
  - ○ disabled breakpoint.
-

- – no breakpoint in line.

## Enable Watchpoint

The **Enable Watchpoint** command activates a watchpoint for the selected variable or memory range.

Enabled watchpoints are indicated by an underline of the selected variable or range of memory. Disabled watchpoints do not have the underline. The underline's color can be configured in the Display Settings preference panel of the IDE Preference window.

## Enter Find String

The **Enter Find String** command copies the selected text in the active window directly into the target search string. It will then appear in the **Find** text box of both the **Find and Replace** and **Find in Files** windows. Once done, use any of the find commands to search for matches without opening any Find-related windows.

## Enter Replace String

The **Enter Replace String** command copies the selected text in the active window directly into the target search string. It will then appear in the **Replace with** text box of both the **Find and Replace** and **Find in Files** windows. Once done, use any of the find commands to search for matches without opening any Find-related windows.

---

**NOTE** Macintosh: Press the Shift key to change the Enter Find String command to the Enter Replace String menu command.

---

## Errors & Warnings

### Errors & Warnings Window

These commands open the Errors and Warnings window.

### Exceptions in Targeted Classes

The **Exceptions in Targeted Classes** command of the **Java** submenu to tell the debugger to break only on exceptions thrown by your own classes in the project.

Choose this command to break on the exceptions thrown by your classes, rather than on the exceptions that Java programs throw in the normal course of execution.

## Exit

The **Exit** command exits the CodeWarrior IDE immediately, provided that:

- All changes to the open editor files are already saved, or
- The open editor files are not changed.

If a Project window is open, the IDE saves all changes to the project file before exiting. If an Editor window is open and changes are not saved, the CodeWarrior IDE asks if you want to save your changes before exiting.

## Expand Window

The **Expand Window** command expands a collapsed window (a window with only its title visible). Only available when a collapsed window is currently active.

## Export Project

The **Export Project** command exports a CodeWarrior project to a file in extensible markup language (XML) format. The IDE prompts for a name and location to save the new XML file.

## Export Project as GNU Makefile

This command exports a CodeWarrior project to a GNU makefile. The IDE displays a message that tells you the name of the makefile and its location on the hard disk.

## Expressions

### Expressions Window

These commands open an Expressions window.

---

## F

### Find

The **Find** command opens the Find and Replace window to perform find operations within the active file.

### Find Definition & Reference

The **Find Definition & Reference** command searches for the definition of the selected routine name in the active Editor window. Searching starts within the source files belonging to the open project. If the IDE does not find a definition, a system beep sounds.

If the IDE does not find the routine definition within the project files, searching continues using the online help system specified in the **IDE Extras** preference panel.

---

**NOTE** Macintosh: Press the Option key to change the Find Definition menu command to the Find Definition & Reference menu command.

---

### Find Definition

The **Find Definition** command searches for the definition of the selected routine name in the active window. Searching occurs in the source files belonging to the open project. If the IDE finds the definition, the source file that contains the definition appears in an Editor window, and the routine name appears highlighted.

If the IDE finds more than one definition, a Message window appears warning of multiple definitions. If the IDE does not find a definition, a system beep sounds.

---

**NOTE** Select the **Activate Browser** option in the **Build Extras** target settings panel and re-compile the project in order to use the **Find Definition** command.

---

## Find in Files

The **Find in Files** command opens the Find in Files window. Once open, The Find in Files window to perform find-and-replace operations across multiple files using specified search criteria.

## Find In Next File

The **Find in Next File** command searches for the next occurrence of the **Find** text box string in the next file listed in the Find in Files window. The menu command as an alternative to using the Find in Files window itself.

## Find In Previous File

This command searches for the next occurrence of the **Find** text box string in the previous file listed in the Find in Files window. The menu command as an alternative to using the Find in Files window itself.

---

<b>NOTE</b>	(Macintosh) Press the Shift key to change the Find In Next File menu command to the Find In Previous File menu command.
-------------	---

---

## Find Next

The **Find Next** command searches for the next occurrence of the Find text box string in the active window.

## Find and Open File

Uses the **Find and Open File** command opens the Find and Open File dialog. Enter a filename, click OK, and the IDE searches the current project access paths as specified in the Access Paths panel of the Target Settings window.

## Find and Open ‘Filename’

The **Find and Open ‘Filename’** command opens an existing text file, using the currently selected text in the Editor window as the target filename.

## Find Previous

The **Find Previous** command searches for the previous occurrence of the Find text box string in the active window.

---

**NOTE** Macintosh: Press the Shift key to change the Find Next menu command to the Find Previous menu command.

---

## Find Previous Selection

The **Find Previous Selection** searches for the previous occurrence of the selected text in the active editor window.

---

**NOTE** Macintosh: Press the Shift key to change the Find Selection menu command to the Find Previous Selection menu command.

---

## Find Reference

The **Find Reference** command searches for the definition of the selected routine name in the active Editor window, using the online help system specified in the **IDE Extras** preference panel.

If the IDE does not find a definition, a system beep sounds.

## Find and Replace

The **Find and Replace** command opens the Find and Replace window. Use this window to perform find-and-replace operations within the active file.

## Find Selection

The **Find Selection** command searches for the next occurrence of the selected text in the active Editor window.

**G**

## Get Next Completion

The **Get Next Completion** command acts as a shortcut that bypasses using the Code Completion window. Instead of scrolling through the Code Completion window to select the next symbol from the one currently selected, use this command to insert that next symbol directly into the active editor window.

## Get Previous Completion

The **Get Previous Completion** command acts as a shortcut that bypasses using the Code Completion window. Instead of scrolling through the Code Completion window to select the previous symbol from the one currently selected, use this command to insert that previous symbol directly into the active editor window.

## Global Variables

### Global Variables Window

These commands open the Global Variables window. Use this window to view global variables for an entire project or for a single file. Click a filename in the **Files** list displays the file's global variables in the **Variables** list.

## Go Back

The **Go Back** command returns to the previous view in the CodeWarrior browser.

## Go Forward

The **Go Forward** command moves to the next view in the CodeWarrior Browser (after you The **Go Back** command to return to a previous view).

## Go to Line

The **Go to Line** command opens the **Line Number** dialog box and enter a specific line number to move the text-insertion point to. If the line number specified exceeds the number of lines in the file, the text-insertion point moves to the last line in the file.

## Group

The **Group** command combines selected objects into a single item in the Layout Editor window of a RAD project.

---

# H

## Hide Breakpoints

The **Hide Breakpoints** command conceals the Breakpoints column, which appears to the left of the source code shown in editor windows.

## Hide Floating Toolbar

The **Hide Floating Toolbar** command conceals the IDE's floating toolbar. After concealing the floating toolbar, the command changes to **Show Floating Toolbar**.

## Hide Main Toolbar

The **Hide Main Toolbar** command conceals the IDE's main toolbar. After concealing the main toolbar, the command changes to **Show Main Toolbar**.

## Hide Window Toolbar

The **Hide Window Toolbar** command conceals the toolbar in the active window. After concealing the window toolbar, the command changes to **Show Window Toolbar**.

## Horizontal Center

The **Horizontal Center** command of the **Align** submenu aligns the horizontal centers of the selected components.

I

## Import Components

The **Import Components** command imports components from another catalog for use with the current catalog.

## Import Project

The **Import Project** command imports project files previously saved in extensible markup language (XML) file with the **Export Project** command into the IDE.

## Insert Reference Template

This command inserts a routine template corresponding to the selected Mac OS Toolbox call in the active window. The IDE uses the online reference database application specified in the **Find Reference Using** list pop-up to search for the routine's definition.

---

## K-L

### Kill

The **Kill** command permanently terminates execution of the target program and returns control to the debugger.

### Left Edges

The **Left Edges** command of the **Align** submenu aligns the left edges of the selected components.

---

## M-N

### **Make**

The **Make** command builds the selected project by compiling and linking its modified and touched files. The results of a successful build depend on the selected project type.

### **Maximize Window**

Windows equivalent of Expand Window.

See also:

- [“Expand Window” on page 493](#)

### **Metrowerks Website**

The **Metrowerks Website** command launches a web browser and display the Metrowerks web site.

### **Minimize Window**

Windows equivalent of Collapse Window.

See also:

- [“Collapse Window” on page 488](#)

### **New**

The **New** command opens the **New** window. The **New** window to create new projects, files, components, and objects.

### **New Class**

The **New Class** command opens the New Class wizard. Use this wizard to help create new classes in a project.

## New Class Browser

The **New Class Browser** command opens a Browser window. The IDE grays out this menu command if the CodeWarrior browser is not activated. This menu command is equivalent to the **Class Browser** menu command.

## New Data Member

The **New Data Member** command opens the New Data Member wizard. Use this wizard to help create new data members for a class.

## New Event

The **New Event** command opens the New Event window. Use this window to help create new events for a selected class in a project.

## New Event Set

The **New Event Set** command opens the New Event Set window to create a new event set for a selected class in a project.

## New Expression

The **New Expression** command creates a new entry in the Expressions window, prompting entry of a new expression.

## New Member Function

The **New Member Function** command opens the New Member Function wizard. Use this wizard to help create new member functions for a class.

## New Method

The **New Method** command opens the New Method window. Use this window to create a new method for a selected class in a project.

## New Property

The **New Property** command opens the New Property. Use this window to create a new property for a selected class in a project.

---

## New Text File

The **New Text File** command creates a new editable text file and open a Editor window.

## No Exceptions

The **No Exceptions** command of the **Java** submenu sets the debugger to not break when exceptions occur.

---

# O

## Object Inspector

The **Object Inspector** command opens the Object Inspector for use with RAD projects.

## Online Manuals

This command opens a list of online manuals about the CodeWarrior IDE, compilers, MSL, and target specific information.

## Open

The **Open** command opens an existing project or source file.

## Open Recent

The **Open Recent** menu item reveals a submenu of recently opened projects and files. Choose a file from the submenu to open that item.

If two or more files in the submenu have identical names, the submenu shows the full paths to those files in order to distinguish between them.

## Open Scripts Folder

This command opens the (Scripts) folder. This command is only available if the **Use Scripts menu** option is enabled.

## Open Workspace

This command opens a workspace file that you previously saved.

---

## P-Q

### Page Setup

The **Page Setup** command sets the options used for printing CodeWarrior IDE files.

### Paste

The **Paste** command replaces the selected text with the contents of the system clipboard into the active Editor window or text box. If no text is selected, the IDE places the clipboard contents at the text-insertion point.

The **Paste** command is unavailable if the Message window is active.

### Precompile

The **Precompile** command precompiles the text file in the active Editor window into a precompiled header file.

### Preferences

The **Preferences** command opens the IDE Preferences window. Use this window to change the global preferences used by the CodeWarrior IDE.

### Preprocess

This command preprocesses selected source files in any language that has a preprocessor, such as C, C++, and Java.

### Print

The **Print** command prints CodeWarrior IDE files, as well as Project, Message, and Errors and Warnings window contents.

---

## Processes

### Processes Window

These commands open the Processes window for those platforms that support it.

### Project Inspector

Opens the Project Inspector window so that you can view information about your project. You can also use this window to manipulate file-specific information.

## Properties

The **Properties** command opens the Object Inspector. Use this window to view or modify the properties of the selected component in a RAD project.

## Quit

### Quit CodeWarrior

Mac OS command equivalent of [Exit](#): See “[Exit](#)” on page 493.

---

## R

### Redo

After undoing an operation, you can redo it. For example, after choosing the **Undo Typing** command to remove some text that you typed, you can choose **Redo Typing** to override the undo and restore the text that you typed.

You can enable the **Use multiple undo** option in the **Editor Settings** preference panel to allow greater flexibility with regard to **Undo** and **Redo** operations. After enabling this option, you can choose **Undo** multiple times to undo multiple actions, and you can **Redo** multiple times to redo multiple actions.

## Refresh All Data

This command updates the data that appears in all windows.

## Register Details Window

The **Register Details Window** command opens the Register Details window, which allows you view descriptions of registers, bit fields, and bit values.

## Registers

### Register Window

These commands reveal the Registers submenu, which can be used to view general registers or FPU registers.

See also:

- [“Register Details Window” on page 505](#)

## Remove Object Code

The **Remove Object Code** command shows the Remove Object Code dialog box. Use this dialog box to remove binary object code from the active project, or to mark the project's files for re-compilation.

## 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 the files that belong to the project and project settings.

## Remove Selected Items

The **Remove Selected Items** command removes the currently selected items from the Project window.

---

**CAUTION** You cannot undo this command.

---

## Replace

The **Replace** command opens the Find and Replace dialog box. Use this dialog box to perform find-and-replace operations within the active file.

## Replace All

The **Replace All** command finds all occurrences of the **Find** text box string and replaces them with the **Replace** text box string. If no text is selected in the active Editor window and there is no text in the **Find** text box, the IDE dims this menu command.

## Replace and Find Next

This command substitutes selected text with the text in the **Replace** text box of the Find window, and then performs a **Find Next** operation. If no text is selected in the active Editor window and there is no text in the Find field of the Find window, the IDE grays out this menu command.

## Replace and Find Previous

This command substitutes selected text with the text in the **Replace** text box of the Find window, and then performs a **Find Previous** operation. If no text is selected in the active Editor window and there is no text in the Find field of the Find window, the IDE grays out this menu command.

---

<b>NOTE</b>	(Mac OS) Press the Shift key to change the Replace and Find Next menu command to the Replace and Find Previous menu command.
-------------	--

---

## Replace Selection

The **Replace Selection** command substitutes the selected text in the active window with the text in the **Replace** text box of the Find window. If no text is selected in the active Editor window, the IDE grays out the menu command.

The menu command to replace one instance of a text string without having to open the Find window. Suppose that you replaced all occurrences of the variable `iCount` with `jCount`. While scrolling through your source code, you notice an instance of the variable `iCount` misspelled as `iCont`. To replace this misspelled variable with `jCount`, select `iCont` and The **Replace Selection** menu command.

## Re-search for Files

The **Re-search for Files** command speeds up builds and other project operations, the IDE caches the locations of project files after finding 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 moved several files and you want the IDE to find the files in their new locations.

If the **Save project entries using relative paths** option is enabled, the IDE does not reset the relative-path information stored with each project entry, so re-searching for files finds 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 only re-searches for header files. To force the IDE to also re-search for source files, choose the **Reset Project Entry Paths** menu command.

If the **Save project entries using relative paths** option is disabled, the IDE re-searches for both header files and source files.

## Reset

The **Reset** command resets the program and return control to the IDE.

## Reset Floating Toolbar

The **Reset Floating Toolbar** command restores the default state of the floating toolbar. Use this command to return the floating toolbar to its original default settings.

## Reset Main Toolbar

The **Reset Main Toolbar** command restores the default state of the main toolbar. Use this command to return the main toolbar to its original default settings.

## Reset Project Entry Paths

The **Reset Project Entry Paths** command resets the location information stored with each project entry and forces the IDE to re-search for the project entries in the access paths. This command does nothing if the **Save project entries using relative paths** option is disabled.

## Reset Window Toolbar

The **Reset Window Toolbar** command restores the default state of the toolbar in the active window. Use this command to return the toolbar to its original default settings.

## Resize

Choose **Resize** reveals the Resize submenu.

See also:

- [“To Largest Height” on page 515](#)
- [“To Largest Width” on page 515](#)
- [“To Smallest Height” on page 516](#)
- [“To Smallest Width” on page 516](#)

## Restart

The **Restart** command terminates the current debugging session, then starts a new debugging session.

## Restore Window

The **Restore Window** command restores a minimized window (a window reduced to an item in the task bar).

## Resume

The **Resume** command switches from the IDE to the running application. This menu command only appears after the IDE starts a debugging session and the application being debugged is currently running.

## Revert

The **Revert** command restores the last saved version of the active Editor window.

## Right Edges

The **Right Edges** command of the **Align** submenu aligns the right edges of the selected components.

## Run

The **Run** command compiles, links, and creates a standalone application, and run that application. This command is unavailable if the project creates libraries, shared libraries, code resources, and other non-application binaries.

## Run to Cursor

The **Run to Cursor** command sets a temporary breakpoint at the line of source code that has the text-insertion point, then runs the program.

---

## S

## Save

The **Save** command saves the contents of the active window to disk.

## Save A Copy As

The **Save A Copy As** command saves the active window to a separate file. This command operates in different ways, depending on the active window.

## Save All

The **Save All** command saves all currently open editor files.

---

<b>NOTE</b>	Mac OS: Press the Option key to change the Save command to the Save All menu command.
-------------	---

---

## Save As

The **Save As** command saves the contents of the active window to disk under a different name.

---

## Save Default Window

The **Save Default Window** command saves the settings of the active Browser window. The IDE applies the saved settings to subsequently opened browser windows.

## Save Workspace

This command saves to a file the current state current state of onscreen windows, recent items, and debugging. Use the dialog box that appears to name the workspace and navigate to a location in which to store the workspace.

## Save Workspace As

This command saves a copy of an existing workspace. Use this command to save the workspace under a different name. Select All

The **Select All** command selects all text in the active window or text box. This command is usually used in conjunction with other **Edit** menu commands such as Cut, Copy, and Clear.

## Send To Back

The **Send To Back** command moves the selected objects so that they are displayed behind all other objects.

## Set Breakpoint

The **Set Breakpoint** command sets a breakpoint at the currently selected line. If the **Show Breakpoints** option is active, the Breakpoints column in the editor windows will display a marker next to each line with a breakpoint.

## Set/Clear Breakpoint

The **Set/Clear Breakpoint** command displays the **Set/Clear Breakpoints** dialog that lets you set or clear a breakpoint at a particular address or symbol.

## Set Default Project

The **Set Default Project** command sets a particular project as the default project when more than one project is open. This is the project that all commands are directed.

## Set Default Target

The **Set Default Target** command works with a different build target within the current project. Choose the build target to work with from the submenu. This menu command is useful for switching between multiple build targets in a project and performing a build for each target.

## Set Eventpoint

This command opens a submenu that lets you set an eventpoint at the currently selected line. If the **Show Breakpoints** option is active, the Breakpoints column in the editor windows shows a marker next to each line with an eventpoint. The marker represents the eventpoint type.

## Set Watchpoint

The **Set Watchpoint** command sets a watchpoint for the selected variable or memory range. Watchpoint variables are identified using an underline.

## Shift Left

The **Shift Left** command shifts the selected source code one tab to the left. The amount of shift is controlled by the **Tab Size** option.

## Shift Right

The **Shift Right** command shifts the selected source code one tab to the right. The amount of shift is controlled by the **Tab Size** option.

## Show Breakpoints

The **Show Breakpoints** command displays the Breakpoints column in editor windows. When active, the Breakpoints column appears along the left edge of all editor windows.

## Show Floating Toolbar

The **Show Floating Toolbar** command displays the IDE's floating toolbar. After displaying the floating toolbar, the command changes to **Hide Floating Toolbar**.

## Show Main Toolbar

The **Show Main Toolbar** command displays the IDE's main toolbar. After displaying the main toolbar, the command changes to Hide Main Toolbar.

## Show Types

The **Show Types** command displays the data types of all local and global variables that appear in the active variable pane or variable window.

## Show Window Toolbar

The **Show Window Toolbar** command displays the toolbar in the active window. After displaying the window toolbar, the command changes to Hide Window Toolbar.

## Snap To Grid

The **Snap to Grid** command automatically aligns objects with the grid in the layout window of a RAD project.

## Stack Editor Windows

The **Stack Editor Windows** command arranges open editor windows one on top of another, with their window titles visible.

## Step Into

The **Step Into** command executes a single statement, stepping into function calls.

## Step Out

The **Step Out** command executes the remainder of the current function, then exit to that function's caller.

## Step Over

The **Step Over** command executes a single statement, stepping over function calls.

## Stop

This command temporarily suspends execution of the target program and returns control to the debugger.

## Stop Build

The **Stop Build** command halts the build currently in progress.

## Switch to Monitor

This command transfers control from the CodeWarrior debugger to an external third-party debugger.

## Symbolics

### Symbolics Window

These commands open the Symbolics window. Use this window to examine the executable files in a project.

## Synchronize Modification Dates

The **Synchronize Modification Dates** command updates the modification dates stored in the project file. The IDE checks the modification date of each file in the project and marks for recompiling those files modified since the last successful compile process.

---

## T-U

### Target Settings

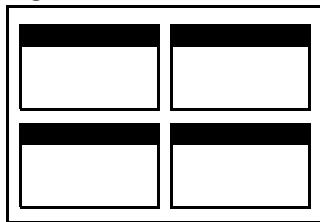
The **Target Settings** command displays the Target Settings window. This window contains settings panels used by the active build target. The name of the menu command changes, based on the name of the current build target. For example, if the name of the current build target is `ReleaseTarget`, the name of the menu command changes to **ReleaseTarget Settings**.

---

## Tile Editor Windows

The **Tile Editor Windows** command arranges and resizes all open editor windows so that none overlap on the monitor.

**Figure 35.1** Tile Editor windows example.



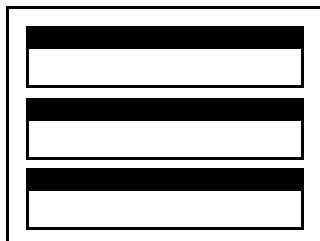
## Tile Editor Windows Vertically

The **Tile Editor Windows Vertically** command resizes all open editor windows to be vertically long, and arranged horizontally across the monitor so that all are viewable.

## Tile Horizontally

This command arranges open editor windows horizontally so that none overlap.

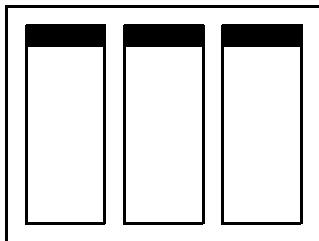
**Figure 35.2** Tile horizontally example.



## Tile Vertically

This command resizes open editor windows vertically and arrange them so that none overlap.

**Figure 35.3 Tile vertically example.**



## To Grid

The **To Grid** command of the **Align** submenu aligns selected components to a grid in the layout. You can display or hide the on screen grid.

## To Largest Height

The **To Largest Height** command of the **Resize** submenu resizes the selected components to match the height of the component with the largest height.

## To Largest Width

The **To Largest Width** command of the **Resize** submenu resizes the selected components to match the width of the component with the largest width.

## Toolbars

Choose **Toolbars** reveals the Toolbars submenu.

See also:

- [“Show Window Toolbar” on page 512](#)
- [“Hide Window Toolbar” on page 498](#)
- [“Reset Window Toolbar” on page 508](#)
- [“Clear Window Toolbar” on page 487](#)
- [“Show Main Toolbar” on page 512](#)
- [“Hide Main Toolbar” on page 498](#)
- [“Reset Main Toolbar” on page 507](#)
- [“Clear Main Toolbar” on page 486](#)

- “[Hide Floating Toolbar](#)” on page 498
- “[Show Floating Toolbar](#)” on page 511
- “[Reset Floating Toolbar](#)” on page 507
- “[Clear Floating Toolbar](#)” on page 486

## ToolServer Worksheet

The **ToolServer Worksheet** command opens the ToolServer Worksheet window for use with the Apple® ToolServer™ application program.

The IDE can disable this command for these reasons:

- You did not install ToolServer on your computer.
- You installed ToolServer on your computer, but you did not yet start it.

## Top Edges

The **Top Edges** command of the **Align** submenu aligns the top edges of the selected components.

## To Smallest Height

The **To Smallest Height** command of the **Resize** submenu resizes the selected components to match the height of the component with the smallest height.

## To Smallest Width

The **To Smallest Width** command of the **Resize** submenu resizes the selected components to match the width of the component with the smallest width.

## Unanchor Floating Toolbar

The **Unanchor Floating Toolbar** command detaches the floating toolbar from beneath the menu bar.

## Unapply Difference

The **Unapply Difference** command reverses the action of the **Apply Difference** command in a file-comparison window.

## Uncaught Exceptions Only

The **Uncaught Exceptions Only** command of the **Java** submenu tells the debugger to break only on unhandled exceptions.

## Undo

The **Undo** command reverses the last action. The name of this menu command changes based upon the editor settings as well as the most recent action. For example, after typing text in an open Editor window, the **Undo** command changes its name to **Undo Typing**. Choose the **Undo Typing** command to remove the just typed text.

By default, only one undo or redo action is allowed. If the **Use multiple undo** option is enabled, undo and redo can act upon multiple actions.

## Ungroup

The **Ungroup** command separates a selected group so that you can move each component independently.

---

## V-Z

## Version Control Settings

The **Version Control Settings** command opens the VCS Settings window.

## Vertical Center

The **Vertical Center** command of the **Align** submenu aligns the vertical centers of the selected components.

## View Array

The **View Array** command to create a separate window displays a selected array.

## View As

The **View As** command displays a selected variable as a value of a specified data type.

---

## **View As Binary**

The **View As Binary** command displays the selected variable as a binary value.

## **View As Character**

The **View As Character** command displays the selected variable as a character value.

## **View As C String**

The **View As C String** command displays the selected variable as a C character string.

## **View As Default**

The **View As Default** command displays the selected variable in its default format, based on the variable's type.

## **View As Enumeration**

The **View As Enumeration** command displays the selected variable as an enumeration.

## **View As Fixed**

The **View As Fixed** command displays the selected variable as a fixed-type numerical value.

## **View As Floating Point**

The **View As Floating Point** command displays the selected variable as a floating-point value.

## **View As Fract**

This command displays the selected variable as a fractional data type.

---

### **NOTE**

The fractional data type is specific to the Mac OS.

---

## **View As Hexadecimal**

The **View As Hexadecimal** command displays the selected variable as a hexadecimal value.

## **View As Pascal String**

The **View As Pascal String** command displays the selected variable as a Pascal character string.

## **View As Signed Decimal**

This command displays the selected variable as a signed decimal value.

## **View As Unicode String**

The **View As Unicode String** command displays the selected variable as a Unicode character string.

## **View As Unsigned Decimal**

The **View As Unsigned Decimal** command displays the selected variable as an unsigned decimal value.

## **View Disassembly**

This command changes the data view to show language disassembly.

## **View Memory**

The **View Memory** command displays the contents of memory as a hexadecimal/ASCII character dump.

## **View Memory As**

The **View Memory As** command displays the memory that a selected variable occupies or the memory to which a selected register points.

## **View Mixed**

This command changes the data view to show source code intermixed with assembly code.

## **View Raw Data**

This command changes the data view to show raw data (instead of formatting that data as source code, disassembly, or another format).

## **View Source**

This command changes the data view to show source code.

## **View Variable**

The **View Variable** command creates a separate window displays a selected variable.

## **Watchpoints**

### **Watchpoints Window**

These commands open a Watchpoints window.

## **Zoom Window**

The **Zoom Window** command expands the active window to its previously set size. Choose **Zoom Window** a second time to return the window to its original size.

# Index

---

## Symbols

#include files  
  caching 428  
%file command-line string 429  
%line command-line string 430  
(Scripts) folder 477, 502  
. \* [\_\_]Data 361  
.mcp, acronym meaning 35  
\(.\*)\ 361  
—throw() 483

## A

about  
  console applications 85  
  dockable windows 69  
  Files view in Project window 47  
  markers 119  
  object wiring 309  
  workspaces 81  
About Metrowerks CodeWarrior menu command 481  
about projects 29  
Absolute Path option  
  in Source Trees preference panel 441  
  in Type pop-up menu 441  
abstract  
  icon for 163  
Access Filter display 166  
Access Paths panel  
  options  
    System Paths 440  
    User Paths 445  
Access Paths settings panel 359, 389  
  Framework column 392  
  options  
    Add 391  
    Add Default 391  
    Always Search User Paths 391  
    Change 391  
    Host Flags 391  
    Interpret DOS and Unix Paths 391  
    Remove 391  
    Require Framework Style Includes 391  
  System Paths 391  
  System Paths list 391  
User Paths 391  
  User Paths list 391  
Recursive Search column 392  
Search Status column 391  
Action 329  
action, defined 309  
Activate Browser 205  
Activate Browser Coloring option 411  
  Text Colors panel 426  
Activate Browser option 494  
Activate Syntax Coloring option 411, 417  
  Text Colors panel 426, 429, 440  
activating  
  automatic code completion 107  
Add button 391  
Add Default button 411  
Add Files menu command 481  
Add Window menu command 481  
adding  
  components to a layout 292  
  remote connections 381  
  source trees 363  
advanced  
  debugging 201  
advanced debugging, defined 202  
advanced projects 39  
advantages  
  of using IDE 23  
Align command 289  
Align submenu 482, 483, 498, 499, 508, 515, 516, 517  
  Horizontal Center command 498  
  Left Edges command 499  
  Vertical Center command 508, 515, 516, 517  
All Exceptions command in Java Exceptions  
  submenu 482  
All Info option  
  of Plugin Diagnostics 430  
alphabetical sorting, or Functions list pop-up 117  
Always Search User Paths option 412  
Ancestor pop-up 168  
Anchor Floating Toolbar command in Toolbar  
  submenu 482  
Appears in Menus 138, 329, 330  
Apple Help Viewer 425

- 
- Apple menu 464
  - Application field 412
  - applications
    - creating for the console 85
    - for the console, about 85
    - for the console, creating 86
  - Apply Difference command 482, 516
  - Arguments field 412
  - Arithmetic Optimizations 400
  - Array window 221
    - opening 223
  - arrays
    - setting default viewing size for unbounded 419
  - Attempt To Use Dynamic Type of C++, Object Pascal And SOM Objects option 412
  - Auto Indent option 412
  - Auto Repeat 329
  - Auto Target Libraries option 412
  - auto-complete code. *See* code completion. 107
  - automatic code completion, activating 107
  - automatic code completion, deactivating 109
  - Automatic Invocation option 413
  - Automatically Launch Applications When SYM File Opened option 413
  - Auto-target Libraries option 412
  - B**
    - Background option 414
    - Balance Flash Delay option 414
      - Editor Settings panel 414
    - Balance menu command 483
    - Balance While Typing option 414
    - balancing
      - punctuation 105
      - balancing punctuation 105
      - balancing punctuation, toggling 106
    - Balloon Help 340, 421, 478
    - Base Classes field 180
    - basic debugging 193
    - Bottom Edges button, in layout editor 287
    - Bottom Edges command 483
    - Branch Optimizations 400
    - Break command 197
    - Break menu command 483
    - Break On C++ Exception menu command 483
    - Break on Java Exceptions command 483
    - breakpoints
      - clearing all 235
      - clearing in source panes 234
      - defined 231
      - icon for active 232
      - icon for inactive 232
      - purpose of 231
      - setting conditional 236
      - setting in source panes 234
      - setting temporary 236
      - status 232
      - tasks 233
      - using 231
      - viewing 233
    - breakpoints column
      - in editor window 98
    - Breakpoints menu command 483
    - Breakpoints window 231, 256, 259
      - Class column 232
      - clearing breakpoints 235
      - Condition column 232
      - Location column 232
      - opening 233
    - Breakpoints Window menu command 483
    - Bring To Front command 289
    - Bring To Front menu command 484
    - Bring Up To Date command 54, 55
    - Bring Up To Date menu command 484
    - Bring Up To Date option 425
    - Browse In Processes Window option 381, 382
    - browser 151
      - Class Browser window 155
      - Classes pane 161
      - collapsing panes 160
      - creating new classes 162, 177, 178
      - creating new data members 186
      - creating new member functions 182, 183, 185
      - expanding panes 160
      - hierarchy windows 168
      - Member Functions pane 163
      - overview 25
      - printing class hierarchies 169
      - purpose of 147
      - setting options 147
      - Source pane 165
      - status area 165
      - using contextual menu 152
      - viewing data by contents 173

---

viewing data by inheritance 168  
working with 147

Browser Access Filters 157

Browser Commands option 415  
Editor Settings panel 428

Browser Contents 156

Browser Contents command 484

Browser Contents window 172  
Symbols list 172

browser database  
defined 147

Browser menu 415, 479

Browser Path option 415

Browser Wizard 177

bug, defined 191

Build Before Running option 415

Build Extras panel  
options  
Initial Directory field 428  
Use External Debugger 443  
Use modification date caching 443

Build Extras settings panel 205, 392  
options  
Application 394  
Arguments 394  
Cache Subprojects 393  
Dump internal browse information after compile 393  
Generate Browser Data From 393  
Initial directory 394  
Use External Debugger 394  
Use modification date caching 393

Build Extras target settings panel 494

Build Progress menu command 484

Build Progress Window menu command 484

Build Settings panel  
options  
Include file cache 428  
Play sound after ‘Bring Up To Date’ & ‘Make’ 433  
Save open files before build 436  
Show message after building up-to-date project 438  
Success 440  
Use Local Project Data Storage 443

Build Settings preference panel 351  
options  
Build before running 353

Compiler thread stack 353  
Failure 353  
Include file cache 353  
Play sound after ‘Bring Up To Date’ & ‘Make’ 353  
Save open files before build 353  
Show message after building up-to-date project 353  
Success 353  
Use Local Project Data Storage 353

build system  
overview 25

build targets 31  
configuring 58  
creating 56  
management 51  
managing 56  
moving 52  
removing 51, 56  
renaming 54, 58  
setting default 57  
strategies for 42

Button  
Choose 336  
Delete 337  
Export 347  
Import 347  
New Binding 346  
Save 338

buttons  
Add 391  
Add Default 411  
Change 391  
Edit 372  
Export Panel 422  
Factory Settings 424  
Installed Products 481  
Purge Cache 434  
Remove 391  
resetting in toolbars 343

**C**

cache  
purging 434

Cache Edited Files Between Debug Sessions option 415

Cache Subprojects option 416

Cache Symbolics Between Runs option 416

---

caching  
    **#include** files 428  
    precompiled headers 428  
Can't Redo menu command 453, 466  
Can't Undo menu command 453, 466  
Cancel  
    in Find and Replace window 128  
    in Find window 126  
Cancel button, in Remove Markers window 120  
Cascade menu command 484  
Case sensitive 126, 128, 131  
Case Sensitive option 416  
Catalog list pop-up, in component palette 291  
Change button 391  
Change Program Counter menu command 484  
changing  
    line views in a hierarchical window 170  
    register data views 217  
    register values 217  
    remote connections 382  
    source trees 364  
Check Syntax command 485  
Checkbox  
    Numeric Keypad Bindings 346  
Checkout Status column  
    in Files view of Project window 48  
child windows, defined 69  
choosing  
    a default project 38  
    linkers 273  
    one character from many in regular  
        expressions 142  
class browser  
    purpose of windows 155  
    working with windows 155  
Class Browser menu command 485  
Class Browser window 155  
    Classes pane 157  
    Data Members pane 157  
    Member Functions pane 157  
    Source pane 157  
    Status area 157  
Class column  
    in Breakpoints window 232  
class data  
    viewing from hierarchy windows 159  
Class Declaration 166  
Class Hierarchy 156  
Class Hierarchy menu command 485  
Class Hierarchy Window menu command 485  
class hierarchy windows  
    purpose of 167  
    working with 167  
classes  
    creating 162, 177, 178  
    hiding pane for 162  
    showing pane for 162  
    sorting list of 163  
Classes option 372  
Classes pane 161  
    in Class Browser window 157  
classes.zip 482  
clear  
    Watchpoint 261  
    watchpoint 246  
Clear All Breakpoints menu command 486  
Clear All Watchpoints menu command 486  
Clear Breakpoint menu command 486  
Clear Eventpoint menu command 486  
Clear Floating Toolbar command in Toolbar  
    submenu 486  
Clear Main Toolbar menu command 486  
Clear menu command 485  
clear Watchpoint 249  
clear Watchpoint from Variable window 249  
clear Watchpoint in Thread window 261  
clear watchpoint in Thread window 246  
Clear Watchpoint menu command 486  
Clear Window Toolbar command in Toolbar  
    submenu 487  
clearing  
    all breakpoints 235  
    breakpoints in Breakpoints window 235  
    breakpoints in source panes 234  
    eventpoints 256  
clearing eventpoint  
    Symbolics window 263  
    Thread window 261  
clearing watchpoint  
    Thread window 246  
client area, defined 69  
Clone Existing Target option 56  
Close All command 65  
Close All Editor Documents menu command 487

---

Close All menu command 487  
Close Catalog button, in component palette 291  
Close Catalog menu command 487  
Close command 39, 65  
Close menu command 487  
Close Non-debugging Windows option 416  
Close Workspace menu command 487  
closing  
    all files 65  
    dockable windows 79  
    files 65  
    projects 39  
    workspaces 84  
Cmds&Bindings file 347  
Code 400  
code  
    adding markers to 120  
    locating 115  
    setting breakpoints in 234  
Code column  
    in Files view of Project window 48  
code completion 107  
    configuration 107  
Code Completion Delay option 416  
Code Completion preference panel 366  
    options  
        Automatic Invocation 366  
        Case sensitive 366  
        Code Completion Delay 366  
        Display deprecated items 366  
        Window follows insertion point 366  
Code Completion window 110  
code completion, activating automatic behavior 107  
code completion, deactivating automatic behavior 109  
code completion, for data members 113  
code completion, for parameter lists 114  
code completion, navigating window 111  
code completion, selecting items 112  
code completion, triggering by keyboard 108  
code completion, triggering from IDE menu bar 108  
Code Generation section, of Target Settings panels 398  
code, navigating 115  
CodeWarrior  
    menu reference 451  
    overview 21  
CodeWarrior Glossary command 488  
CodeWarrior Help menu command 488

CodeWarrior IDE  
    Apple menu 464  
    Browser menu 479  
    CodeWarrior menu 464  
    Data menu 461, 473  
    Debug menu 459, 471  
    Edit menu 453, 466  
    File menu 451, 464  
    Help menu 463, 477  
    Layout menu 478  
    Project menu 457, 469  
    Scripts menu 477  
    Search menu 455, 467  
    Tools menu 476  
    VCS menu 476  
    Window menu 454, 462, 475

CodeWarrior menu 464  
CodeWarriorU.com 17  
Collapse Non-debugging Windows option 417  
Collapse Window menu command 488  
collapsing  
    browser panes 160  
    dockable windows 78  
COM *See* Component Object Model. 419  
command  
    Run 239  
Command Actions  
    Arguments 333  
    Defining (Mac OS) 336  
    Defining (Windows) 332  
    Directory 333  
    Execute 332  
Command Group  
    Delete 337  
Command Groups 336  
    Delete 336  
Commands  
    Import 347  
    Modify 329  
commands 163  
    About Metrowerks CodeWarrior 481  
    Add Files 481  
    Add Window 481  
    Apply Difference 482  
    Balance 483  
    Bottom Edges 483  
    Break 197, 483  
    Break On C++ Exception 483

---

Break on Java Exceptions 483  
Breakpoints 483  
Breakpoints Window 483  
Bring To Front 484  
Bring Up To Date 484  
Browser Contents 156, 484  
Build Progress 484  
Build Progress Window 484  
Can't Redo 453, 466  
Can't Undo 453, 466  
Cascade 484  
Change Program Counter 484  
Check Syntax 485  
Class Browser 485  
Class Declaration 166  
Class Hierarchy 156, 485  
Class Hierarchy Window 485  
Clear 485  
Clear All Breakpoints 486  
Clear All Watchpoints 486  
Clear Breakpoint 486  
Clear Eventpoint 486  
Clear Main Toolbar 486  
Clear Watchpoint 486  
Close 39, 487  
Close All 487  
Close All Editor Documents 487  
Close Catalog 487  
Close Workspace 487  
CodeWarrior Glossary 488  
CodeWarrior Help 488  
Collapse Window 488  
Commands & Key Bindings 487  
Compare Files 488  
Compile 488  
Complete Code 488  
Component Catalog 488  
Component Palette 489  
Connect 489  
Copy 489  
Copy To Expression 489  
Create Design 489  
Create Group 489  
Create Target 490  
Customize 490  
Cut 490  
Cycle View 490  
Debug 195, 490  
Delete 491  
Diagonal Line 170  
Disable Breakpoint 491  
Disable Watchpoint 491  
Disassemble 491  
Display Grid 491  
Enable Breakpoint 491  
Enable Watchpoint 492  
Enter Find String 139, 492  
Enter Replace String 492  
Errors And Warnings 492  
Errors And Warnings Window 492  
Exit 493  
Expand Window 493  
Export Project 38, 493, 499  
Export Project as GNU Makefile 493  
Expressions 493  
Expressions Window 493  
File Path 49  
Find 127, 494  
Find and Open 'Filename' 495  
Find and Open File 495  
Find And Replace 496  
Find Definition 494  
Find Definition & Reference 122, 494  
Find In Files 495  
Find In Next File 495  
Find In Previous File 495  
Find Next 495  
Find Previous 496  
Find Previous Selection 496  
Find Reference 122, 496  
Find Selection 138, 496  
Get Next Completion 497  
Get Previous Completion 497  
Global Variables 497  
Global Variables Window 497  
Go Back 156, 497  
Go Forward 156, 497  
Go To Line 497  
Group 498  
Hide Breakpoints 498  
Hide Classes 162  
Hide Classes pane 166  
Hide Window Toolbar 498  
Import Components 499  
Import Project 39, 499  
Insert Reference Template 499  
Kill 198, 499  
Make 500

---

---

Maximize Window 500  
Metrowerks Website 500  
Minimize Window 500  
New 500  
New Class 500  
New Class Browser 501  
New Data 501  
New Event 501  
New Event Set 501  
New Expression 501  
New Item 161  
New Member Function 501  
New Method 501  
New Property 501  
New Text File 502  
Object Inspector 502  
Online Manuals 502  
Open 502  
Open File 165  
Open In Windows Explorer 49  
Open Recent 502  
Open Scripts Folder 502  
Open Workspace 503  
Page Setup 503  
Pane Collapse 160  
Pane Expand 160  
Precompile 503  
Preferences 503  
Print 503  
Processes 504  
Processes Window 504  
Project Inspector 36  
Properties 504  
RAD 282  
Redo 504  
Refresh All Data 505  
Register Details Window 505  
Register Windows 505  
Registers 505  
Remove Object Code 505  
Remove Object Code & Compact 505  
Remove Toolbar Item 342  
Replace 129, 135, 506  
Replace All 135, 506  
Replace and Find Next 506  
Restart 199  
Resume 198, 508  
Revert 508  
Run 198, 434, 509  
Run To Cursor 509  
Save Default Window 510  
Save Workspace 510  
Save Workspace As 510  
Select All 510  
Send To Back 510  
Set Breakpoint 510  
Set Default Project 38, 510  
Set Default Target 511  
Set Eventpoint 511  
Set Watchpoint 511  
Shift Right 511  
Show Breakpoints 486, 511  
Show Classes 162  
Show Classes pane 166  
Show Inherited 157  
Show private 158  
Show protected 158  
Show public 158  
Show Types 512  
Show Window Toolbar 498  
Single Class Hierarchy Window 156  
Snap To Grid 512  
Sort Alphabetical 161, 163  
Sort Hierarchical 161  
Stack Editor Windows 512  
Step Into 196  
Step Out 196  
Step Over 197, 512  
Stop 197  
Stop Build 513  
Straight Line 170  
Switch To Monitor 513  
Symbolics 513  
Symbolics Window 513  
Synchronize Modification Dates 513  
Toolbars 515  
View Array 517  
View as implementor 158  
View as subclass 158  
View As Unsigned Decimal 517, 518, 519  
View as user 158  
View Disassembly 519  
View Mixed 520  
View Source 520  
View Variable 520  
Watchpoints 520  
Watchpoints Window 520  
Zoom Window 520

---

---

Commands & Key Bindings menu command 487  
Commands tab 327, 329, 344  
Comments option 417  
Common Subexpression Elimination 400  
Compare Files menu command 488  
Compile menu command 488  
compiler  
    avoiding crashes 417  
Compiler option 417  
Compiler option, in Generate Browser Data From menu 426  
compiler thread stack  
    and avoiding compiler crashes 417  
Compiler Thread Stack field 417  
Complete Code menu command 488  
completing  
    the Create Wire wizard 315  
completing code 107  
component catalog  
    adding components to 302  
    removing components from 304  
Component Catalog button, in component palette 291  
Component Catalog menu command 488  
component catalog window  
    opening 301  
Component list, in layout editor 288  
Component Object Model 419  
Component Palette 290  
component palette  
    opening 291  
Component Palette button, in layout editor 288  
Component Palette menu command 489  
component tool buttons, in component palette 291  
components  
    adding to a component catalog 302  
    moving in a layout 293  
    removing from a catalog 304  
    removing from layouts 292  
    resizing 293  
Concurrent Compiles panel  
    options  
        Use Concurrent Compiles 435, 442  
        User Specified 445  
Concurrent Compiles preference panel 353  
    options  
        Recommended 354  
        Use Concurrent Compiles 354  
            User Specified 354  
Condition column  
    of Breakpoints window 232  
Condition field 248, 259  
conditional breakpoints  
    setting 236  
conditional eventpoint 258  
Conditional Watchpoint 248  
configuring  
    build targets 58  
    code completion 107  
    targets 58  
Confirm “Kill Process” When Closing Or Quitting option 418  
Confirm Invalid File Modification Dates When Debugging option 417  
Connect menu command 489  
Connection pop-up menu, in Remote Debugging settings panel 408  
Connection Type option 382  
console applications  
    creating 85, 86  
        applications  
            creating console applications 86  
console applications, about 85  
constant  
    adding to a variable 212  
Constants option 372  
Context Popup Delay option 418  
contextual menu  
    using for browser 152  
contextual menus 203  
File Path command 49  
Layout Editor 289  
Open In Windows Explorer command 49  
    using 204  
    using to dock a window 72  
conventions  
    figures 19  
    for manual 19  
    keyboard shortcuts 20  
Copy And Expression Propagation 400  
Copy menu command 489  
Copy Propagation 400  
Copy To Expression command 489  
cores, debugging multiple 228  
Create Design menu command 489

---

- 
- Create Group menu command 489
  - Create Target command 56
  - Create Target menu command 490
  - Create Wire wizard
    - completing 315
  - Create Wire wizard, using 315
  - creating
    - a new data member 164
    - build targets 56
    - console application 86
    - console applications 86
    - custom project stationery 40
    - empty designs 281
    - empty projects 35
    - files (Macintosh) 62
    - files (Windows) 61
    - member functions 163
    - menu items in a menubar component 306
    - menus in a menubar component 306
    - new classes 162, 177, 178
    - new data member 185
    - new data members 186
    - new member function 182
    - new member functions 183
    - projects from makefiles 33
    - projects using stationery 33
    - subprojects 41
    - targets 56
  - creating console applications 85
  - creating wires with contextual menus 312
  - creating wires with menus 312
  - cross-platform
    - opening projects 36
  - Current Target list pop-up 46
  - Current Target menu 341
  - Custom Keywords settings panel 401
  - custom project stationery 40
  - Customize button, in layout editor 288
  - Customize command 289
  - Customize IDE Commands window 327, 344, 346
    - Action 329
    - Appears in Menus 329, 330
    - Auto Repeat 329
    - Key Bindings 329
    - Name field 329
    - New Binding 329
    - New Group 330
  - Customize menu command 490
  - Cut command 490
  - CVS 361
  - Cycle View menu command 490
- D**
- Data column
    - in Files view of Project window 48
  - data members
    - completing code 113
    - creating 164, 186
    - identifier icons 163
  - Data Members pane 164
    - in Class Browser window 157
  - Data menu 461, 473
  - database
    - navigation for browser 151
  - deactivating
    - automatic code completion 109
  - Dead Code Elimination 400
  - Dead Store Elimination 400
  - Debug column
    - in Files view of Project window 48
  - Debug command 54, 55, 195
  - Debug menu 418, 459, 471
    - Clear All Breakpoints command 460, 472
    - Disable Watchpoint command 460, 473
    - Enable Breakpoint command 460, 472
    - Enable Watchpoint command 460, 472, 473
    - Hide Breakpoints command 460, 472
  - Debug menu command 490
  - debugger 434
    - advanced techniques 201
    - attaching to a process 210
    - choosing for an opened symbolics file 381
    - overview 26
    - starting 195
  - Debugger Commands option 418
  - Debugger section, of IDE preference panels 375
  - Debugger section, of Target Settings panels 403
  - Debugger Settings panel 218, 406
    - options
      - Auto-target Libraries 407
      - Cache symbolics between runs 407
      - Default language entry point 407, 419
      - Location of Relocated Libraries and Code Resources 407, 431
      - Log System Messages 408, 431

---

Program entry point 434  
Stop at Watchpoints 408, 440  
Stop on application launch 407, 440  
Update data every n seconds 442  
Update data every *n* seconds 408  
User specified 407

Debugger window. *See* Thread window. 245, 259

debugger, defined 191

debugging

- advanced, defined 202
- basic skills 193
- introduction 191
- multiple cores 228

debugging information 245, 259

debugging session

- starting 195

Declaration File field 179

Default File Format option 418

default file-name extensions 422

Default Language Entry Point option

- Debugger Settings panel 419

default projects 38

Default Size For Unbounded Arrays option 419

default target, setting 57

default workspace, defined 81

default workspace, using 82

definition

- of action 309
- of advanced debugging 202
- of breakpoints 231
- of bug 191
- of child windows 69
- of client area 69
- of debugger 191
- of default workspace 81
- of dock 69
- of non-modal 71
- of project 29
- of symbolics file 192
- of touch 48
- of wire 309
- of workspace 81

definitions

- file set 134
- IDE 17
- regular expression 140
- symbols 122

definitions, symbol 121

Delete menu command 491

deleting wires with the layout editor 314

Design view 52

designs

- creating empty 281
- removing layouts from 283

Designs view 37, 49

development

- process cycle for software 21

diagnostics

- disabling for plug-ins 481
- enabling for plug-ins 481

Diagonal Line 170

dialog boxes

- New Connection 381

difference from Single-Class Hierarchy window 170

Disable Breakpoint menu command 491

Disable Third Party COM Plugins option 419

Disable Watchpoint menu command 491

disabling

- plug-in diagnostics 481

Disassemble menu command 491

Display Deprecated Items option 419

Display Grid command 289

Display Grid menu command 491

Display Settings panel

- options
  - Show all locals 438
  - Show tasks in separate windows 438
  - Show values as decimal instead of hex 439
  - Show variable location 439
  - Show variable types 439
  - Show variable values in source code 439
  - Sort functions by method name in symbolics window 439
  - Variable Values Change 446
  - Watchpoint Indicator 446

Display Settings preference panel 375

- options
  - Attempt to use dynamic type of C++, Object Pascal and SOM objects 377
  - Default size for unbounded arrays 377
  - Show all locals 376
  - Show tasks in separate windows 377
  - Show values as decimal instead of hex 376
  - Show variable location 376
  - Show variable types 376
  - Show variable values in source code 377

---

Sort functions by method name in symbolics window 377  
Variable values change 376  
Watchpoint indicator 376  
DLL 380, 412  
Do Nothing option 419  
Do Nothing To Project Windows option 419  
dock bars 76  
dock, defined 69  
dockable windows 69, 71  
    closing 79  
    collapsing 78  
    dock bars 76  
    expanding 78  
    moving 78  
    suppressing 76  
    turning off 76  
dockable windows, about 69  
docked window type 70  
docking  
    windows of the same kind 73  
Document Settings list pop-up 96  
document settings pop-up  
    using 96  
documentation  
    formats 18  
    structure 18  
    types 19  
Documents option  
    IDE Extras panel 420  
Don't Step Into Runtime Support Code 420  
Don't Step Into Runtime Support Code option 420  
Done button, in Remove Markers window 120  
Down  
    in Find and Replace window 128  
    in Find window 127  
drag and drop  
    using to dock a window 72  
Drag And Drop Editing option 420  
drawing wires 311  
Dump Internal Browse Information After Compile option 420  
dump memory 519

**E**

Edit button 372  
Edit Commands option 420

Edit Language option 421  
Edit menu 420, 453, 466  
Edit Wire button, in layout editor 287  
Edit Wire command 290, 479  
editing  
    source code 101  
    symbols, shortcuts for 105  
editing layouts 285  
editor 91  
    overview 25  
    third-party support 444  
Editor section, of IDE preference panels 365  
Editor section, of Target Settings panels 401  
Editor Settings panel  
    options  
        Balance Flash Delay 414  
        Browser Commands 428  
        Font Preferences 426  
        Insert Template Commands 428  
        Left margin click selects line 430  
        Project Commands 434  
        Relaxed C popup parsing 435  
        Selection position 437  
        Sort function popup 440  
        Use multiple undo 444  
        VCS Commands 446  
        Window position and size 446  
Editor Settings preference panel 366  
    options  
        Balance Flash Delay 368  
        Balance while typing 368  
        Browser Commands 367  
        Debugger Commands 368  
        Default file format 368  
        Drag and drop editing 368  
        Edit Commands 367  
        Enable Virtual Space 368  
        Font preferences 367  
        Insert Template Commands 368  
        Left margin click selects line 368  
        Project Commands 368  
        Relaxed C popup parsing 368  
        Selection position 367  
        Sort function popup 368  
        Use multiple undo 368  
        VCS Commands 368  
        Window position and size 367  
editor toolbar 93

---

---

editor window 91, 252  
    adding panes to 98  
    breakpoints column 98  
    collapsing toolbar in 94  
    expanding toolbar in 94  
    line and column indicator 98  
    pane splitter controls 98  
    removing panes from 99  
    resizing panes 99  
    text editing area 98

editor windows  
    other 97  
    selecting text in 102

Emacs text editor 429, 430

empty designs  
    creating 281

empty projects  
    creating 35

Empty Target option 56

Enable Automatic Toolbar Help option 421

Enable Breakpoint menu command 491

Enable Browser option 484

Enable Remote Debugging option 421

Enable Virtual Space option 421

Enable Watchpoint menu command 492

enabling  
    plug-in diagnostics 481

enabling and disabling Watchpoint  
    Thread window 247

end-of-line format 418

enlarging panes, in browser 160

Enter Find String command 139

Enter Find String menu command 492

Enter Replace String menu command 492

Enums option 372

Environment Settings option 421

Environment Variable option  
    of Source Trees preference panel 442

Environment Variable option, in Type pop-up menu 442

environment variables  
    Macintosh limitations 442

EOL format 418

Errors And Warnings menu command 492

Errors And Warnings Window menu command 492

Errors Only option  
    of Plugin Diagnostics 430

eventpoint  
    clearing in the Symbolics window 263  
    clearing in the Thread window 261  
    conditional 258  
    setting in the Symbolics window 262  
    setting in the Thread window 260

eventpoint icon 256

Eventpoints  
    purpose of 251  
    using 251

eventpoints  
    clearing 256  
    Log Message 253  
    Log Point 251, 253  
    Log Point, Log Message 253  
    Log Point, Speak Message 253  
    Log Point, Treat as Expression 253  
    Pause Point 251, 254  
    Script Point 251, 254  
    setting 255  
    Skip Point 251, 254  
    Sound Point 251, 254  
    Sound Point, Speak Message 254  
    Speak Message 253  
    Treat as Expression 253

Events tab, in object inspector 296

Exceptions In Targeted Classes command in Java  
    Exceptions submenu 492

executable files  
    adding to the Other Executables list 405  
    changing in the Other Executables list 405  
    removing from the Other Executables list 406

Exit menu command 493

Expand Window menu command 493

expanding  
    browser panes 160  
    dockable windows 78

Export 347

Export Panel button 328, 350, 422

Export Project as GNU Makefile menu command 493

Export Project command 38

Export Project menu command 493, 499

exporting  
    projects to XML files 38

Expression Simplification 400

Expressions menu command 493

Expressions window 210  
    adding expressions 211

---

opening 211  
Expressions Window menu command 493  
Extension field 422  
external editor  
    using on the Macintosh 357  
external editor support 444

**F**

Factory Settings button 424  
Failure option 425  
FDI mode  
    and dockable windows 69  
FDI *See* Floating Document Interface. 444  
Field  
    Run App/Script 336  
field  
    Base Classes 180  
fields  
    Application 412  
    Arguments 412  
    Compiler thread stack 417  
    Condition 248, 259  
    Declaration File 179  
    Extension 422  
    File Type 425  
    IP Address 382, 383  
    Relative to class 179  
figure conventions 19  
File  
    Cmds&Bindings 347  
File column  
    in Files view of Project window 48  
%file command-line string 429  
file management 51  
File Mappings list 417  
File Mappings settings panel 396  
    options  
        Add 397  
        Change 398  
        Compiler 397  
        Edit Language 397  
        Extension 397  
        File Mappings list 397  
        File Type 397  
        Flags 397  
        Ignored By Make flag 397  
        Launchable flag 397  
Precompiled File flag 397  
Remove 398  
Resource File flag 397  
File menu 451, 464  
    New Text File command 465  
file modification icon 97  
File Path command 49  
file paths  
    viewing 49  
file set  
    defined 134  
File Type field 425  
File Type option 417  
file-name extensions  
    default settings 422  
files  
    close all 65  
    closing 65  
    creating (Macintosh) 62  
    creating (Windows) 61  
    file set, definition of 134  
    inspecting 36  
    moving 52  
    opening 62  
    print selections 67  
    printing 66  
    renaming 53  
    replacing text in 129  
    reverting 67  
    save all 64  
    saving 64  
    saving copies 65  
    searching (multiple) 134  
    searching (single) 127  
    touching 54  
    touching all 54  
    untouching 55  
    untouching all 55  
    working with 61  
Files tab 51  
Files view 37, 52, 55  
    Checkout Status column 48  
    Code column 48  
    Data column 48  
    Debug column 48  
    File column 48  
    Interfaces list pop-up 48  
    Sort Order button 48

---

Target column 48  
Touch column 48  
Files view, about 47  
files, tasks for managing 61  
Find 126, 128, 130  
    by text selection 137  
    single-file 126  
Find All 126, 130  
Find and compare operations option  
    Shielded Folders panel 425  
Find And Open ‘Filename’ menu command 495  
Find and Open File command 495  
Find and Replace  
    multiple-file 129  
    single-file 127  
Find And Replace menu command 496  
Find command 127, 494  
Find Definition & Reference command 122  
Find Definition & Reference menu command 494  
Find Definition menu command 494  
Find In Files menu command 495  
Find in Files window  
    In Files tab 134  
    In Folders tab 131  
    In Projects tab 132  
    In Symbolics tab 133  
Find In Next File menu command 495  
Find In Previous File menu command 495  
Find list pop-up 126, 128, 130  
Find Next menu command 495  
Find Previous  
    using 137  
Find Previous menu command 496  
Find Previous Selection menu command 496  
Find Reference command 122  
Find Reference menu command 496  
Find Reference using option  
    IDE Extras panel 425  
Find Selection command 138  
Find Selection menu command 496  
Find symbols with prefix 105  
Find symbols with substring 105  
Find text box 126, 128, 130  
finding text  
    explained 125  
    overview 125  
Flags pop-up menu 397  
Ignored By Make flag 397  
Launchable flag 397  
Precompiled File flag 397  
Resource File flag 397  
floating a window 75  
Floating Document Interface 444  
floating window type 70  
focus bar 53  
folders  
    searching (multiple) 131  
Font & Tabs panel 370  
    options  
        Font 426  
        Scripts 437  
        Size 439  
        Tab indents selection 441  
        Tab Inserts Spaces 441  
        Tab Size 441  
Font & Tabs preference panel 368, 371  
    options  
        Auto Indent 369  
        Font 369  
        Script 369  
        Size 369  
        Tab indents selection 369  
        Tab Inserts Spaces 369  
        Tab Size 369  
Font option  
    Font & Tabs panel 426  
Font Preferences option  
    Editor Settings panel 426  
Font Settings 370  
Foreground option  
    Text Colors panel 426  
format, for end of line (EOL) 418  
formats  
    for documentation 18  
FPU Registers 215  
Framework column, in Access Paths panel 392  
function  
    New Data Member 164  
functions  
    creating new member 163  
    locating 115, 116  
Functions list pop-up 95  
    sorting alphabetically 117  
    using 116  
Functions option 372

---

---

## G

General Registers 215  
General section, of IDE preference panels 351  
Generate Browser Data From option 426  
    Compiler 426  
    Language Parser 427  
    Language Parser, Macro file 427  
    Language Parser, Prefix file 427  
    None 426  
Generate Constructor and Destructor 180  
Get Next Completion menu command 497  
Get next symbol 105  
Get Previous Completion menu command 497  
Get previous symbol 105  
Global Optimizations settings panel 398  
    options  
        Details 399  
        Faster Execution Speed 399  
        Optimization Level slider 399  
        Smaller Code Size 399  
Global Register Allocation 400  
Global Register Allocation Only For Temporary Values 400  
Global Settings panel  
    options  
        Maintain Files in Cache 431  
        Select stack crawl window when task is stopped 437  
Global Settings preference panel  
    options  
        Auto Target Libraries 380  
        Automatically launch applications when SYM file opened 379  
        Cache Edited Files Between Debug Sessions 379  
        Confirm "Kill Process" when closing or quitting 379  
        Confirm invalid file modification dates when debugging 379  
        Don't step into runtime support code 380  
        Maintain files in cache 379  
        Purge Cache 379  
        Select stack crawl window when task is stopped 379  
Global Variables menu command 497  
Global Variables window 213  
    opening 214

Global Variables Window menu command 497  
Globals option 372  
Go Back 156  
Go Back menu command 497  
Go Forward 156  
Go Forward menu command 497  
Go To Line menu command 497  
going back 118  
going forward 118  
going to a particular line 118  
Grid button, in layout editor 288  
Grid Size X option  
    Layout Editor panel 427  
Grid Size Y option  
    Layout Editor panel 427  
Group button, in layout editor 287  
group management 51  
Group menu command 498  
grouping  
    regular expressions 142  
groups  
    moving 52  
    removing 51  
    renaming 53  
    touching 54  
    touching all 54  
    untouching 55  
    untouching all 55

## H

headers  
    caching precompiled headers 428  
Help menu 463, 477  
Help Preferences panel 358  
    options  
        Browser Path 358  
        Set 358  
Hide Breakpoints menu command 498  
Hide Classes 162  
Hide Classes pane 166  
Hide Floating Toolbar command 498  
Hide Main Toolbar command in Toolbar submenu 498  
Hide non-debugging windows option  
    Windowing panel 427  
Hide Window Toolbar command 498  
hiding

---

classes pane 162  
Hierarchy Control 168  
hierarchy window 168  
hierarchy windows  
    changing line views 170  
    using to view class data 159  
Horizontal Center button, in layout editor 287  
Horizontal Center command 498  
Host Application for Libraries & Code Resources  
    option  
        Runtime Settings panel 427  
Host Application For Libraries And Code Resources  
    field  
        of Runtime Settings panel 427  
host-specific registers 216  
how to  
    activate automatic code completion 107  
    add a component to a layout 292  
    add a constant to a variable 212  
    add a keyword to a keyword set 402  
    add an executable file 405  
    add components to a component catalog 302  
    add expressions (Expressions window) 211  
    add markers to a source file 120  
    add panes to an editor window 98  
    add remote connections 381  
    add source trees 363  
    adding subprojects to a project 41  
    alphabetize Functions list pop-up order 117  
    attach the debugger to a process 210  
    balance punctuation 105  
    change an executable file 405  
    change line views in a hierarchical window 170  
    change register data views 217  
    change register values 217  
    change remote connections 382  
    change source trees 364  
    choose a default project 38  
    clear all breakpoints 235  
    clear breakpoints in source panes 234  
    clear breakpoints in the Breakpoints window 235  
    close a docked window 79  
    close a workspace 84  
    close projects 39  
    collapse a docked window 78  
    collapse browser panes 160  
    collapse the editor window toolbar 94  
    complete code for data members 113  
    complete code for parameter lists 114  
    complete the Create Wire wizard 315  
    create a console application 86  
    create a menu in a menubar component 306  
    create a menu item in a menubar component 306  
    create a new class 162, 177, 178  
    create a new data member 185, 186  
    create a new data members 164  
    create a new member function 163, 182, 183  
    create a wire by drawing it 311  
    create a wire by using a contextual menu 312  
    create a wire by using a menu 312  
    create custom project stationery 40  
    create empty projects 35  
    create new projects from makefiles 33  
    create new projects using project stationery 33  
    deactivate automatic code completion 109  
    delete a wire by using the layout editor 314  
    delete wires by using the object inspector 319  
    dock a window by using a contextual menu 72  
    dock a window by using drag and drop 72  
    dock windows of the same kind 73  
    expand a docked window 78  
    expand browser panes 160  
    expand the editor window toolbar 94  
    export projects to XML files 38  
    float a window 75  
    generate project link maps 274  
    go to a particular line 118  
    hide the classes pane 162  
    import projects saved as XML files 39  
    indent text blocks 104  
    insert a reference template 123  
    look up symbol definitions 122  
    make a summation of two variables 212  
    make a window an MDI Child 76  
    modify a wire by using the layout editor 313  
    modify wires by using the object inspector 318  
    move a docked window 78  
    move components in a layout 293  
    navigate browser data 151  
    navigate Code Completion window 111  
    navigate to a marker 120  
    open a layout 289  
    open a recent workspace 84  
    open a single-class hierarchical window 171  
    open a workspace 83  
    open an Array window 223  
    open projects 35

---

open projects created on other hosts 36  
open registers in a separate Registers window 218  
open subprojects 42  
open the Breakpoints window 233  
open the component catalog window 301  
open the component palette 291  
open the Expressions window 211  
open the Global Variables window 214  
open the IDE Preferences window 351  
open the Log window 219  
open the object inspector 296  
open the Processes window 209, 210  
open the Registers window 216  
open the Symbolics window 206  
open the symbols window 175  
open the Target Settings window 387  
open the Variable window 220  
overstrike text (Windows) 103  
print class hierarchies 169  
print projects 37  
remove a component from a layout 292  
remove a keyword from a keyword set 403  
remove a layout from a design 283  
remove a marker from a source file 120  
remove a menu from a menubar component 307  
remove a menu item from a menubar  
    component 307  
remove all markers from a source file 121  
remove an executable file 406  
remove components from a catalog 304  
remove panes from an editor window 99  
remove remote connections 365, 383  
remove source trees 364  
re-order wires by using the object inspector 318  
replace text across multiple items 135  
replace text in a single file 129  
resize a component in a layout 293  
resize panes in an editor window 99  
run a program 198  
save a copy of a workspace 83  
save a workspace 82  
save projects 36  
search a single file 127  
search across multiple files 134  
search across multiple folders 131  
search across multiple projects 132  
search across multiple symbolics files 133  
search with a text selection 138  
search with a text selection across multiple  
    window 139  
select entire routines 103  
select item in Code Completion window 112  
select lines 102  
select multiple lines 102  
select rectangular portions of lines 102  
select text in editor windows 102  
set breakpoints in source panes 234  
set conditional breakpoints 236  
set temporary breakpoints 236  
show the classes pane 162  
sort the classes list 163  
sort the Wires tab by using a contextual menu 320  
suppress dockable windows 76  
toggle automatic punctuation balancing 106  
toggle the symbol hint 203  
trigger code completion by keyboard 108  
trigger code completion from IDE menu bar 108  
undock a window 74  
unfloat a window 75  
unindent text blocks 104  
use an external editor on the Macintosh 357  
use contextual menus 204  
use the browser contextual menu 152  
use the default workspace 82  
use the document settings pop-up 96  
use the Executables pane (Symbolics window) 206,  
    207  
use the Files pane (Symbolics window) 207  
use the Find Previous command 137  
use the Functions list pop-up 116  
use the Functions pane (Symbolics window) 207  
use the Interfaces list pop-up 116  
use the Process pane 209  
use the symbol hint 203  
use the Version Control System (VCS) pop-up 97  
use virtual space 103  
view a file path 49  
view browser data by contents 173  
view browser data by inheritance 168  
view class data from hierarchy window 159  
view registers 216  
view wires by using the object inspector 318

## I

### icon

for Tools menu 477  
for VCS menu 476

---

<p>icons</p> <ul style="list-style-type: none"> <li>eventpoints 256</li> <li>file modification 97</li> <li>for data members 163</li> <li>for member functions 163</li> <li>Inactive Eventpoint 252</li> <li>Log Point 252</li> <li>Pause Point 252</li> <li>Script Point 252</li> <li>Skip Point 252</li> <li>Sound Point 252</li> </ul> <p>IDE</p> <ul style="list-style-type: none"> <li>advantages 23</li> <li>and threading 417</li> <li>Apple menu 464</li> <li>Browser menu 479</li> <li>Code Completion window 110</li> <li>CodeWarrior menu 464</li> <li>Data menu 461, 473</li> <li>Debug menu 459, 471</li> <li>defined 17</li> <li>Edit menu 453, 466</li> <li>editing source code 101</li> <li>editor 91</li> <li>File menu 451, 464</li> <li>Help menu 463, 477</li> <li>Layout menu 478</li> <li>linkers 273</li> <li>Mac-hosted 340</li> <li>menu reference 451</li> <li>preferences, working with 349</li> <li>project manager and build targets 29</li> <li>Project menu 457, 469</li> <li>Scripts menu 477</li> <li>Search menu 455, 467</li> <li>target settings, working with 385</li> <li>Tools menu 476</li> <li>tools overview 24</li> <li>User Guide overview 17</li> <li>VCS menu 476</li> <li>Window menu 454, 462, 475</li> <li>Windows-hosted 341</li> <li>workspaces 81</li> </ul> <p>IDE Extras 476, 477</p> <p>IDE Extras panel</p> <ul style="list-style-type: none"> <li>options</li> <li>    Documents 420</li> <li>    Find Reference using 425</li> <li>    Launch Editor 429</li> </ul>	<p>Launch Editor w/ Line # 430</p> <p>Menu bar layout 431</p> <p>Projects 434</p> <p>Symbolics 440</p> <p>Use Default Workspace' 442</p> <p>Use External Editor 443</p> <p>Use Multiple Document Interface 444</p> <p>Use Script menu 444</p> <p>Use ToolServer menu 445</p> <p>Workspaces 447</p> <p>Zoom windows to full screen 447</p> <p>IDE Extras preference panel 354</p> <p>options</p> <ul style="list-style-type: none"> <li>    Context popup delay 356</li> <li>    Documents 356</li> <li>    Enable automatic Toolbar help 357</li> <li>    Find Reference using 357</li> <li>    Launch Editor 356</li> <li>    Launch Editor w/ Line # 356</li> <li>    Menu bar layout 356</li> <li>    Projects 356</li> <li>    Recent symbolics 356</li> <li>    Use Default workspace 357</li> <li>    Use External Editor 356</li> <li>    Use Multiple Document Interface 356</li> <li>    Use Script menu 356</li> <li>    Use Third Party Editor 356</li> <li>    Use ToolServer menu 356</li> <li>    Zoom windows to full screen 356</li> </ul> <p>Use Third Party Editor option 444</p> <p>IDE Preference Panels list 350</p> <p>IDE Preference Panels, Font &amp; Tabs 370</p> <p>IDE Preference Panels, Font Settings 370</p> <p>IDE preferences</p> <ul style="list-style-type: none"> <li>    Activate Browser Coloring 372</li> <li>    Activate Syntax Coloring 372</li> <li>    Add 360, 362, 381</li> <li>    Attempt to use dynamic type of C++, Object Pascal and SOM objects 377</li> <li>    Auto Indent 369</li> <li>    Auto Target Libraries 380</li> <li>    Automatic Invocation 366</li> <li>    Automatically launch applications when SYM file opened 379</li> <li>    Background 372</li> <li>    Balance Flash Delay 368</li> <li>    Balance while typing 368</li> <li>    Browser Commands 367</li> <li>    Browser Path 358</li> </ul>
--	---

---

Build before running	353
Cache Edited Files Between Debug Sessions	379
Case sensitive	366
Change	361, 362, 381
Choose	362
Classes	372
Close non-debugging windows	378
Code Completion Delay	366
Collapse non-debugging windows	378
Comments	372
Compiler thread stack	353
Confirm "Kill Process" when closing or quitting	379
Confirm invalid file modification dates when debugging	379
Constants	372
Context popup delay	356
Debugger Commands	368
Default file format	368
Default size for unbounded arrays	377
Disable third party COM plugins	359
Display deprecated items	366
Do nothing	378
Do nothing to project windows	378
Documents	356
Don't step into runtime support code	380
Drag and drop editing	368
Edit	372
Edit Commands	367
Enable automatic Toolbar help	357
Enable Virtual Space	368
Enums	372
Failure	353
Find and compare operations	360
Find Reference using	357
Font	369
Font preferences	367
Foreground	372
Functions	372
Globals	372
Grid Size X	384
Grid Size Y	384
Hide non-debugging windows	378
Include file cache	353
Insert Template Commands	368
Keywords	372
Launch Editor	356
Launch Editor w/ Line #	356
Left margin click selects line	368
Level	359
Macros	372
Maintain files in cache	379
Menu bar layout	356
Minimize non-debugging windows	378
Monitor for debugging	378
Move open windows to debugging monitor when debugging starts	378
Name	362
Open windows on debugging monitor during debugging	378
Other	373
Play sound after 'Bring Up To Date' & 'Make'	353
Project Commands	368
Project operations	360
Projects	356
Purge Cache	379
Recent symbolics	356
Recommended	354
Regular Expression	360
Relaxed C popup parsing	368
Remote Connection list	381
Remove	361, 362, 381
Save open files before build	353
Script	369
Select stack crawl window when task is stopped	379
Selection position	367
SEt	358
Set 1, Set 2, Set 3, Set 4	372
Shielded folder list	360
Show all locals	376
Show message after building up-to-date project	353
Show tasks in separate windows	377
Show the component palette when opening a form	384
Show the object inspector when opening a form	384
Show values as decimal instead of hex	376
Show variable location	376
Show variable types	376
Show variable values in source code	377
Size	369
Sort function popup	368
Sort functions by method name in symbolics window	377
Source Tree list	362
Strings	372

---

---

Success 353  
Tab indents selection 369  
Tab Inserts Spaces 369  
Tab Size 369  
Templates 372  
Type 362  
TypeDefs 373  
Use Concurrent Compiles 354  
Use Debugging Monitor 378  
Use Default workspace 357  
Use External Editor 356  
Use Local Project Data Storage 353  
Use Multiple Document Interface 356  
Use multiple undo 368  
Use Script menu 356  
Use Third Party Editor 356  
Use ToolServer menu 356  
User Specified 354  
Variable values change 376  
VCS Commands 368  
Watchpoint indicator 376  
Window follows insertion point 366  
Window position and size 367  
Zoom windows to full screen 356

IDE Preferences window 328, 349, 350  
Apply button 351  
Cancel button 351  
Factory Settings button 328, 350  
IDE Preference Panels list 350  
Import Panel 428  
Import Panel button 328, 350  
OK button 351  
opening 351  
Revert Panel button 328, 350  
Save button 328, 351

Ignored By Make File flag 397  
Import button 347  
Import Commands 347  
Import Components menu command 499  
Import Panel 428  
Import Project command 39  
Import Project menu command 499  
importing  
  projects saved as XML files 39

In Files 131  
In Files tab 134  
In Folders 131  
In Folders tab 131

In Projects 131  
In Projects tab 132  
In Symbolics 131  
In Symbolics tab 133  
Inactive Eventpoint icon 252  
Include file cache option  
  Build Settings panel 428  
Include Files 185  
Include files 181  
#include files  
  caching 428  
indenting  
  text blocks 104

information  
  debugging 245, 259  
  symbolics 249, 261

Initial Directory field  
  Build Extras panel 428

Initializer 187

Insert Reference Template 123  
Insert Reference Template menu command 499  
Insert Template Commands option  
  Editor Settings panel 428

inserting a reference template 123

inspecting  
  project files 36

inspecting wires 317

Installed Products button 481

Instruction Scheduling 401

Integrated Development Environment. *See* IDE.

interface files  
  locating 115, 116

Interface menu 54

Interfaces list pop-up  
  in Files view of Project window 48  
  using 116

interfaces list pop-up 95

introduction  
  debugger 191

IP Address field 382, 383

**J**

Java Exceptions Submenu  
  No Exceptions command 502

Java Exceptions submenu  
  All Exceptions command 482

- 
- Exceptions In Targeted Classes command 492  
Uncaught Exceptions Only command 517  
Java submenu 482, 492, 502, 517
- K**
- Key Bindings 327, 329  
  Add 345  
  Customize 344  
key bindings 122  
keyboard conventions 20  
keyboard shortcuts  
  Find symbols with prefix 105  
  Find symbols with substring 105  
  Get next symbol 105  
  Get previous symbol 105  
keywords  
  adding to a keyword set 402  
  removing from a keyword set 403  
Keywords option  
  Text Colors panel 429  
Kill command 198  
Kill menu command 499
- L**
- Language Parser option, in Generate Browser Data  
  From menu 427  
Launch Editor option  
  IDE Extras panel 429  
Launch Editor w/ Line # option  
  IDE Extras panel 430  
Launch Remote Host Application option  
  Remote Debugging settings panel 430  
Launchable flag 397  
layout  
  moving components in 293  
layout editing 285  
layout editor 286  
Layout Editor contextual menu 289  
Layout Editor panel  
  options  
    Grid Size X 427  
    Grid Size Y 427  
    Show the component palette when opening a form 438  
    Show the object inspector when opening a form 438  
Layout Editor preference panel 384
- options  
  Grid Size X 384  
  Grid Size Y 384  
  Show the component palette when opening a form 384  
  Show the object inspector when opening a form 384
- layout management 51  
Layout manager, in layout editor 288  
Layout menu 478  
layout wizards 285  
layouts  
  adding components to 292  
  layout editor 286  
  moving 52  
  opening 289  
  removing 51  
  removing components from 292  
  removing from a design 283  
  renaming 53
- Left Edges button, in layout editor 287  
Left Edges command 499  
Left margin click selects line option  
  Editor Settings panel 430  
Level option  
  Plugin Settings panel 430  
Lifetime Based Register Allocation 401  
line  
  going to in source code 118  
line and column indicator, in editor window 98  
%line command-line string 430  
Line Display 168  
lines, selecting 102  
lines, selecting multiple 102  
lines, selecting rectangular portions of 102  
link maps  
  generating for projects 274  
Link Order tab 51  
Link Order view 37, 49, 52, 53  
Linker option  
  Target Settings panel 431  
linkers 273  
  choosing 273  
linking projects 274  
Linux  
  modifier key mappings 20  
list

---

of symbols in Browser Contents window 172

list menus

- document settings 96
- functions 95
- interfaces 95
- markers 95
- VCS 96

list pop-up menus

- Current Target 46

list pop-ups

- Ancestor 168
- Browser Access Filters 157
- document settings 96
- functions 95
- interfaces 95
- markers 95
- Symbols 172
- VCS 157

lists

- File Mappings 417

Live Range Splitting 400

locating functions 115, 116

locating interface files 115, 116

locating source code 115

Location column

- in Breakpoints window 232

Location of Relocated Libraries and Code Resources option

- Debugger Settings panel 431

Log Point 251, 253

- Log Message 253
- Speak Message 253
- Treat as Expression 253

Log Point icon 252

Log System Messages 218

Log System Messages option

- Debugger Settings panel 431

Log Window

- Log System Messages option 218

Log window 218

- opening 219

looking up symbol definitions 122

Loop Transformations 400

Loop Unrolling 401

Loop Unrolling (Opt For Speed Only) 401

Loop-Invariant Code Motion 400

**M**

Mac OS

- QuickHelp 121
- QuickView 121, 123
- THINK Reference 123

Mac OS X API 425

Macintosh

- creating files 62
- using an external editor 357

Macintosh menu layout 464

Macro file option, in Generate Browser Data From menu 427

Macros option 372

Maintain Files In Cache option 415

Maintain Files in Cache option

- Global Settings panel 431

Make command 53, 54, 55

Make menu command 500

Make option 425

Make toolbar button 46

Makefile Importer wizard 33

makefiles

- converting into projects 33

managing

- build targets 56
- projects 32
- targets 56

managing files, tasks 61

manipulating text 101

manual conventions 19

markers 119

- adding to a source file 120
- navigating to 120
- removing all from source files 121
- removing from source files 120

Markers list pop-up 95

Markers list, in Remove Markers window 119

Match whole word 126, 128, 131

matching

- any character with regular expressions 141
- line beginnings and endings with regular expressions 143
- replace strings to find strings with regular expressions 143, 144

Maximize Window menu command 500

.mcp, acronym meaning 35

MDI 432

---

MDI child  
    making a window into 76  
MDI child window type 70  
MDI mode  
    and dockable windows 69  
MDI. *See* Multiple Document Interface. 444  
Member Function Declaration 184  
member functions  
    creating 163, 183  
    identifier icons 163  
Member Functions pane 163  
    in Class Browser window 157  
memory dump 519  
Memory window 223, 239  
Menu  
    Current Target 341  
menu  
    Search 122  
Menu bar layout option  
    IDE Extras panel 431  
menu commands  
    About Metrowerks CodeWarrior 481  
    Add Files 481  
    Add Window 481  
    Apply Difference 482  
    Balance 483  
    Bottom Edges 483  
    Break 483  
    Break On C++ Exception 483  
    Break on Java Exceptions 483  
    Breakpoints 483  
    Breakpoints Window 483  
    Bring To Front 484  
    Bring Up To Date 484  
    Browser Contents 484  
    Build Progress 484  
    Build Progress Window 484  
    Can't Redo 453, 466  
    Can't Undo 453, 466  
    Cascade 484  
    Change Program Counter 484  
    Check Syntax 485  
    Class Browser 485  
    Class Hierarchy 485  
    Class Hierarchy Window 485  
    Clear 485  
    Clear All Breakpoints 486  
    Clear All Watchpoints 486  
    Clear Breakpoint 486  
    Clear Eventpoint 486  
    Clear Watchpoint 486  
    Close 487  
    Close All 487  
    Close All Editor Documents 487  
    Close Catalog 487  
    Close Workspace 487  
    CodeWarrior Help 488  
    Collapse Window 488  
    Commands & Key Bindings 487  
    Compare Files 488  
    Compile 488  
    Complete Code 488  
    Component Catalog 488  
    Component Palette 489  
    Connect 489  
    Copy 489  
    Copy To Expression 489  
    Create Design 489  
    Create Group 489  
    Create Target 490  
    Customize 490  
    Cycle View 490  
    Debug 490  
    Delete 491  
    Disable Breakpoint 491  
    Disable Watchpoint 491  
    Disassemble 491  
    Display Grid 491  
    Enable Breakpoint 491  
    Enable Watchpoint 492  
    Enter Find String 139, 492  
    Enter Replace String 492  
    Errors And Warnings 492  
    Errors And Warnings Window 492  
    Exit 493  
    Expand Window 493  
    Export Project 493, 499  
    Export Project as GNU Makefile 493  
    Expressions 493  
    Expressions Window 493  
    Find 127, 494  
    Find and Open 'Filename' 495  
    Find and Open File 495  
    Find And Replace 496  
    Find Definition 494  
    Find Definition & Reference 494  
    Find In Files 495

---

Find In Next File 495  
Find In Previous File 495  
Find Next 495  
Find Previous 496  
Find Previous Selection 496  
Find Reference 496  
Find Selection 138, 496  
Get Next Completion 497  
Get Previous Completion 497  
Global Variables 497  
Global Variables Window 497  
Go Back 497  
Go Forward 497  
Go To Line 497  
Group 498  
Hide Breakpoints 498  
Hide Window Toolbar 498  
Import Components 499  
Import Project 499  
Insert Reference Template 123, 499  
Kill 499  
Make 500  
Maximize Window 500  
Metrowerks Website 500  
Minimize Window 500  
New 500  
New Class 500  
New Class Browser 501  
New Data 501  
New Event 501  
New Event Set 501  
New Expression 501  
New Member Function 501  
New Method 501  
New Property 501  
New Text File 502  
Object Inspector 502  
Online Manuals 502  
Open 502  
Open Recent 502  
Open Scripts Folder 502  
Open Workspace 503  
Page Setup 503  
Precompile 503  
Preferences 503  
Print 503  
Processes 504  
Processes Window 504  
Properties 504  
Redo 504  
Refresh All Data 505  
Register Details Window 505  
Register Windows 505  
Registers 505  
Remove Object Code 505  
Remove Object Code & Compact 505  
Remove Toolbar Item 342  
Replace 129, 135, 506  
Replace All 135, 506  
Replace and Find Next 506  
Resume 508  
Revert 508  
Run 434, 509  
Run To Cursor 509  
Save Default Window 510  
Save Workspace 510  
Save Workspace As 510  
Select All 510  
Send To Back 510  
Set Breakpoint 510  
Set Default Project 510  
Set Default Target 511  
Set Eventpoint 511  
Set Watchpoint 511  
Shift Right 511  
Show Breakpoints 486, 511  
Show Types 512  
Show Window Toolbar 498  
Snap To Grid 512  
Stack Editor Windows 512  
Step Over 512  
Stop Build 513  
Switch To Monitor 513  
Symbolics 513  
Symbolics Window 513  
Synchronize Modification Dates 513  
Toolbars 515  
View Array 517  
View As Unsigned Decimal 517, 518, 519  
View Disassembly 519  
View Mixed 520  
View Source 520  
View Variable 520  
Watchpoints 520  
Watchpoints Window 520  
Zoom Window 520  
menu layouts  
Macintosh 464

---

Windows 451  
menu reference  
    for IDE 451  
menubar component  
    creating menu items in 306  
    creating menus in 306  
    removing menu items from 307  
    removing menus from 307  
menus 157  
    contextual 203  
    VCS 165  
Metrowerks Website command 500  
Minimize non-debugging windows option  
    Windowing panel 432  
Minimize Window menu command 500  
modifying wires with the layout editor 313  
Monitor for debugging option  
    Windowing panel 432  
Move open windows to debugging monitor when  
    debugging starts option  
    Windowing panel 432  
moving  
    build targets 52  
    dockable windows 78  
    files 52  
    groups 52  
    layouts 52  
    targets 52  
Multi-Class Hierarchy window 167, 170  
multi-core debugging 228  
Multiple Document Interface 444  
multiple files, searching 134  
multiple folders, searching 131  
multiple items, replacing text in 135  
multiple projects, searching 132  
multiple Redo 504  
multiple symbolics files, searching 133  
multiple Undo 504  
multiple-file Find and Replace window 129

**N**

Name field 329  
navigating  
    browser data 151  
    Code Completion window 111  
    to markers 120  
navigating data 151  
navigating source code 115  
New Binding 329, 346  
New C++ Class window 179  
New C++ Data Member window 187  
New C++ Member Function window 184  
New Class Browser menu command 501  
New Class menu command 500  
New Class wizard 162, 177, 178  
New Command 331  
New command 61, 86  
New Command Group  
    Create 330  
New Connection dialog box 381  
New Data Member 164, 184, 187  
new data member functions  
    creating 185  
New Data Member wizard 164, 185, 186  
New Data menu command 501  
New Event menu command 501  
New Event Set menu command 501  
New Expression menu command 501  
New Group 330  
New Item 161  
New Layout 282  
New Layout command 282  
New Member Function menu command 501  
New Member Function wizard 163, 182, 183  
new member functions  
    creating 182  
New Menu Command  
    Create 331, 335  
New menu command 500  
New Method menu command 501  
New Property menu command 501  
New Text File command 62  
New Text File menu command 502  
New Wire button, in layout editor 287  
New Wire command 290, 479  
No Exceptions command 502  
None option  
    of Plugin Diagnostics 430  
None option, in Generate Browser Data From  
    menu 426  
non-modal, defined 71  
notes  
    for the latest release 17

---

Numeric Keypad Bindings 346

## O

Object Inspector 294

object inspector

- opening 296
- sorting the Wires tab 320
- using to delete wires 319
- using to modify wires 318
- using to re-order wires 318
- using to view wires 318

Object Inspector menu command 502

Object list pop-up, in object inspector 295

object wiring 309

- creating wires with contextual menus 312
- creating wires with menus 312
- deleting wires with the layout editor 314
- drawing wires 311
- inspecting wires 317
- modifying wires with the layout editor 313
- sorting the Wires tab in the object inspector 320
- using the object inspector to delete wires 319
- using the object inspector to modify wires 318
- using the object inspector to re-order wires 318
- using the object inspector to view wires 318
- working with 310

object wiring, about 309

Online Manuals menu command 502

Open Catalog button, in component palette 291

Open command 62

Open File 165

Open In Windows Explorer command 49

Open menu command 502

Open Recent menu command 502

Open Scripts Folder menu command 502  
Open windows on debugging monitor during debugging option

Windowing panel 433

Open Workspace menu command 503

opening 175

- a layout 289
- a recent workspace 84
- a single-class hierarchical window 171
- component catalog window 301
- component palette 291
- files 62

IDE Preferences window 351

object inspector 296

projects 35

projects from other hosts 36

subprojects 42

Symbolics window 206

symbols window 175

Variable window 220

workspaces 83

opening last project (default workspace) 442

opening last project, preventing (default workspace) 443

openings

registers in a separate Registers window 218

optimizations

Arithmetic Optimizations 400

Branch Optimizations 400

Common Subexpression Elimination 400

Copy And Expression Propagation 400

Copy Propagation 400

Dead Code Elimination 400

Dead Store Elimination 400

Expression Simplification 400

Global Register Allocation 400

Global Register Allocation Only For Temporary Values 400

Instruction Scheduling 401

Lifetime Based Register Allocation 401

Live Range Splitting 400

Loop Transformations 400

Loop Unrolling 401

Loop Unrolling (Opt For Speed Only) 401

Loop-Invariant Code Motion 400

Peephole Optimization 400

Register Coloring 401

Repeated 401

Strength Reduction 400

Vectorization 401

options 420

Access Paths settings panel 359, 389

Activate Browser 494

Activate Browser Coloring 411

Activate Syntax Coloring 411, 417

Add Default 411

Always Search User Paths 412

Application 412

Arguments 412

Attempt to use dynamic type of C++, Object Pascal and SOM objects 412

Auto Indent 412

---

Auto Target Libraries 412  
Automatic Invocation 413  
Automatically Launch Applications When SYM  
    File Opened 413  
Auto-target Libraries 412  
Background 414  
Balance Flash Delay 414  
Balance while typing 414  
Bring Up To Date 425  
Browse in processes window 381, 382  
Browser Commands 415  
Browser Path 415  
Build before running 415  
Build Extras settings panel 392  
Build Settings preference panel 351  
Cache Edited Files Between Debug Sessions 415  
Cache Subprojects 416  
Cache symbolics between runs 416  
Case Sensitive 416  
choosing host application for non-executable  
    files 427  
Classes 372  
Close non-debugging windows 416  
Code Completion Delay 416  
Code Completion preference panel 366  
Code Generation settings panels 398  
Collapse non-debugging windows 417  
Comments 417  
Compiler 417  
Compiler thread stack 417  
Concurrent Compiles preference panel 353  
Confirm “Kill Process” when closing or  
    quitting 418  
Confirm invalid file modification dates when  
    debugging 417  
Connection Type 382  
Constants 372  
Context popup delay 418  
Custom Keywords settings panel 401  
Debugger Commands 418  
Debugger preference panels 375  
Debugger Settings 218  
Debugger Settings panel 406  
Debugger settings panels 403  
Default File Format 418  
Default size for unbounded arrays 419  
Disable third party COM plugins 419  
Display Deprecated Items 419  
Display Settings preference panel 375  
Do nothing 419  
Do nothing to project windows 419  
Drag and drop editing 420  
Dump internal browse information after  
    compile 420  
Edit Commands 420  
Edit Language 421  
Editor preference panels 365  
Editor settings panels 401  
Editor Settings preference panel 366  
Enable automatic Toolbar help 421  
Enable remote debugging 421  
Enable Virtual Space 421  
Enums 372  
Environment Settings 421  
Failure 425  
File Mappings settings panel 396  
File Type 417  
Font & Tabs preference panel 368, 371  
Functions 372  
General preference panels 351  
Generate Browser Data From 426  
Global Optimizations settings panel 398  
Globals 372  
Help Preferences panel 358  
IDE Extras preference panel 354  
Import Panel 428  
Layout Editor preference panel 384  
Macros 372  
Maintain files in cache 415  
Make 425  
Other 373  
Other Executables settings panel 403  
Plugin Settings preference panel 358  
Purge Cache 415  
RAD Tools preference panels 383  
Remote Connections preference panel 380  
Remote Debugging settings panel 408  
Require Framework Style Includes 436  
Runtime Settings panel 394  
Set 1, Set 2, Set 3, Set 4 372  
setting for browser 147  
Shielded Folders preference panel 359  
Source Trees preference panel 361  
System Paths list 391  
Target Settings panel 388  
Target settings panels 387  
Templates 372  
TypeDefs 373

---

- 
- Use Multiple Document Interface 69
  - User Paths list 391
  - User specified 407
  - Window Follows Insertion Point 446
    - Windowing preference panel 377
  - Options list pop-up, in object inspector 296
  - other editor windows 97
  - Other Executables settings panel 403
  - Other option 373
  - Output Directory option
    - Target Settings panel 433
  - Overlays tab 51
  - overstrike 103
  - overstriking text (Windows) 103
  - overtype. *See* overstrike. 103
  - overview
    - of browser 25
    - of build system 25
    - of CodeWarrior 21
    - of debugger 26
    - of editor 25
    - of IDE project manager and build targets 29
    - of IDE tools 24
    - of IDE User Guide 17
    - of project manager 25
    - of RAD tools 26
    - of search engine 25
- P**
- Page Setup command 503
  - PalmQuest reference 425
  - pane
    - Source 250, 263
    - Variables 246, 260
  - Pane Collapse 160
  - Pane Expand 160
  - pane splitter controls, in editor window 98
  - panels
    - Font & Tabs 370
    - for IDE preferences 350
    - for targets 386
  - panes
    - adding to editor window 98
    - removing from editor window 99
    - resizing in an editor window 99
  - parameter lists
    - completing code 114
  - path caption 97
  - Pause Point 251, 254
  - Pause Point icon 252
  - Peephole Optimization 400
  - Play sound after ‘Bring Up To Date’ & ‘Make’ option
    - Build Settings panel 433
  - Plugin Diagnostics
    - All Info option 430
    - Errors Only option 430
    - None option 430
  - plug-in diagnostics
    - disabling 481
    - enabling 481
  - Plugin Settings panel
    - options
      - Level 430
    - Plugin Settings preference panel 358
    - options
      - Disable third party COM plugins 359
      - Level 359
  - plug-ins
    - saving information about those installed in IDE 481
    - viewing those installed in IDE 481
  - pop-up menus
    - document settings 96
    - functions 95
    - interfaces 95
    - markers 95
    - VCS 96
  - pop-ups
    - Ancestor 168
    - Browser Access Filters 157
    - Symbols 172
    - VCS 157
  - Post-linker option
    - Target Settings panel 433
  - Precompile menu command 503
  - Precompiled File flag 397
  - precompiled headers
    - caching 428
  - preference panel 350
  - preference panels
    - Build Settings 351
    - Code Completion 366
    - Concurrent Compiles 353
    - Display Settings 375
    - Editor Settings 366

---

Font & Tabs 368, 371	Disable third party COM plugins 359
Help Preferences 358	Display deprecated items 366
IDE Extras 354	Do nothing 378
Layout Editor 384	Do nothing to project windows 378
Plugin Settings 358	Documents 356
Remote Connections 380	Don't step into runtime support code 380
reverting 436	Drag and drop editing 368
Shielded Folders 359	Edit 372
Source Trees 361	Edit Commands 367
Windowing 377	Editor 365
preferences	Enable automatic Toolbar help 357
Activate Browser Coloring 372	Enable Virtual Space 368
Activate Syntax Coloring 372	Enums 372
Add 360, 362, 381	Export Panel button 328, 350
Apply button 351	Factory Settings button 328, 350
Attempt to use dynamic type of C++, Object Pascal and SOM objects 377	Failure 353
Auto Indent 369	Find and compare operations 360
Auto Target Libraries 380	Find Reference using 357
Automatic Invocation 366	Font 369
Automatically launch applications when SYM file opened 379	Font preferences 367
Background 372	for IDE 349
Balance Flash Delay 368	Foreground 372
Balance while typing 368	Functions 372
Browser Commands 367	General 351
Browser Path 358	Globals 372
Build before running 353	Grid Size X 384
Cache Edited Files Between Debug Sessions 379	Grid Size Y 384
Cancel button 351	Hide non-debugging windows 378
Case sensitive 366	IDE Preference Panels list 350
Change 361, 362, 381	IDE window 349
Choose 362	Import Panel button 328, 350
Classes 372	Include file cache 353
Close non-debugging windows 378	Insert Template Commands 368
Code Completion Delay 366	Keywords 372
Collapse non-debugging windows 378	Launch Editor 356
Comments 372	Launch Editor w/ Line # 356
Compiler thread stack 353	Left margin click selects line 368
Confirm "Kill Process" when closing or quitting 379	Level 359
Confirm invalid file modification dates when debugging 379	Macros 372
Constants 372	Maintain files in cache 379
Context popup delay 356	Menu bar layout 356
Debugger 375	Minimize non-debugging windows 378
Debugger Commands 368	Monitor for debugging 378
Default file format 368	Move open windows to debugging monitor when debugging starts 378
Default size for unbounded arrays 377	Name 362
	OK button 351
	Open windows on debugging monitor during debugging 378

---

---

Other 373  
Play sound after ‘Bring Up To Date’ & ‘Make’ 353  
Project Commands 368  
Project operations 360  
Projects 356  
Purge Cache 379  
RAD Tools 383  
Recent symbolics 356  
Recommended 354  
Regular Expression 360  
Relaxed C popup parsing 368  
Remote Connection list 381  
Remove 361, 362, 381  
Revert Panel button 328, 350  
Save button 328, 351  
Save open files before build 353  
Script 369  
Select stack crawl window when task is stopped 379  
Selection position 367  
Set 358  
Set 1, Set 2, Set 3, Set 4 372  
Shielded folder list 360  
Show all locals 376  
Show message after building up-to-date project 353  
Show tasks in separate window 377  
Show the component palette when opening a form 384  
Show the object inspector when opening a form 384  
Show values as decimal instead of hex 376  
Show variable location 376  
Show variable types 376  
Show variable values in source code 377  
Size 369  
Sort function popup 368  
Sort functions by method name in symbolics window 377  
Source Tree list 362  
Strings 372  
Success 353  
Tab indents selection 369  
Tab Inserts Spaces 369  
Tab Size 369  
Templates 372  
Type 362  
TypeDefs 373  
Use Concurrent Compiles 354  
Use Debugging Monitor 378  
Use Default workspace 357  
Use External Editor 356  
Use Local Project Data Storage 353  
Use Multiple Document Interface 356  
Use multiple undo 368  
Use Script menu 356  
Use Third Party Editor 356  
Use ToolServer menu 356  
User Specified 354  
Variable values change 376  
VCS Commands 368  
Watchpoint indicator 376  
Window follows insertion point 366  
Window position and size 367  
Zoom windows to full screen 356  
Preferences menu command 503  
Prefix file option, in Generate Browser Data From menu 427  
Pre-linker option  
    Target Settings panel 434  
print  
    file selections 67  
Print command 66, 67, 503  
printing  
    class hierarchies 169  
    files 66  
    projects 37  
process  
    attaching debugger to 210  
    suspended 245, 259  
process cycle  
    of software development 21  
Process pane  
    using 209  
Processes menu command 504  
Processes window 208, 381  
    opening 209, 210  
Processes Window menu command 504  
products  
    saving information about those installed in IDE 481  
    viewing those installed in IDE 481  
program  
    running 198  
Program Arguments field  
    of Runtime Settings panel (Windows) 434  
Program Arguments option

- 
- Runtime Settings panel 434
  - Program Entry Point option
    - Debugger Settings panel 434
  - Program windows. *See* Thread window. 245, 259
  - Project Commands option
    - Editor Settings panel 434
  - project data folder 443
  - Project Inspector command 36
  - project manager 29
    - overview 25
  - Project menu 434, 457, 469
    - Remove Object Code command 458, 471
    - Stop Build command 458, 470
  - Project operations option
    - Shielded Folders panel 434
  - project stationery
    - creating 40
    - custom 40
  - Project window
    - about Files view 47
    - Current Target list pop-up 46
    - Files view
      - Checkout Status column 48
      - Code column 48
      - Data column 48
      - Debug column 48
      - File column 48
      - Interfaces list pop-up 48
      - Sort Order button 48
      - Target column 48
      - Touch column 48
    - Make toolbar button 46
    - Synchronize Modification Dates toolbar button 46
    - Target Settings toolbar button 46
  - project window 45
    - Designs view 49
    - Link Order view 49
    - Targets view 50
    - views 47
  - project window, about 45
  - project, defined 29
  - projects
    - about subprojects 41
    - advanced topics 39
    - choosing default 38
    - closing 39
    - creating custom stationery 40
    - creating empty 35
  - creating subprojects 41
  - creating using makefiles 33
  - creating using stationery 33
  - data folder 443
  - exporting to XML files 38
  - generating link maps for 274
  - importing XML versions of 39
  - inspecting files 36
  - linking 274
  - managing 32
  - opening 35
  - opening from other hosts 36
  - printing 37
  - project window 45
  - project window views 47
  - project window, about 45
  - reopening last one used (default workspace) 442
  - reopening last one used, preventing (default workspace) 443
  - saving 36
  - searching (multiple) 132
  - strategies for 42
  - subprojects, strategies for 42
  - working with 29
- Projects option
    - IDE Extras panel 434
  - Properties button, in layout editor 288
  - Properties command 289
  - Properties menu command 504
  - Properties tab, in object inspector 295
  - punctuation balancing, toggling 106
  - punctuation, balancing 105
  - pure virtual
    - icon for 163
  - Purge Cache button 434
  - Purge Cache option 415
  - purging cache 434
  - purpose
    - of Browser Contents window 172
    - of Classes pane in browser 161
    - of Data Members pane 164
    - of Member functions pane 163
    - of Multi-Class Hierarchy window 167
    - of Single-Class Hierarchy window 170
    - of Source pane 165
    - of status area in browser 165
    - of Symbols window 174

---

## **Q**

QuickDraw 432  
QuickHelp (Mac OS) 121  
QuickView 121, 123, 425  
QuickView, Mac OS 123  
QuickView, THINK Reference 123

## **R**

RAD  
    object wiring 309  
RAD tools  
    overview 26  
RAD Tools section, of IDE preference panels 383  
Recursive Search column, in Access Paths panel 392  
Redo menu command 504  
reference information  
    for IDE menus 451  
reference template 123  
reference template, inserting 123  
reference templates (Macintosh) 123  
Refresh All Data menu command 505  
Register Coloring 401  
Register Details Window menu command 505  
Register Windows menu command 505  
registers  
    changing data views of 217  
    changing values of 217  
    FPU Registers 215  
    General Registers 215  
    host-specific 216  
    viewing 216  
Registers menu command 505  
Registers window 214  
    opening 216  
    opening more than one 218  
Registry Key option  
    of Source Trees preference panel 442  
Registry Key option, in Type pop-up menu 442  
Regular expression 126, 128, 131  
Regular Expression option  
    Shielded Folders panel 435  
regular expressions 140  
    .\*[\_]Data 361  
    \(.\*\) 361  
    choosing one character from many 142  
    CVS 361

defined 140  
grouping 142  
matching any character 141  
matching line beginnings and endings 143  
    using the find string in the replace string 143, 144  
Relative to class field 179  
Relaxed C popup parsing option  
    Editor Settings panel 435  
release notes 17  
remembering last project (default workspace) 442  
remembering last project, turning off (default workspace) 443  
remote connections  
    adding 381  
    changing 382  
    removing 365, 383  
Remote Connections preference panel 380  
options  
    Add 381  
    Change 381  
    Remote Connection list 381  
    Remove 381  
Remote Debugging settings panel 408  
Connection pop-up menu 408  
options  
    Launch remote host application 430  
    Remote download path 435  
Remote Download Path option  
    Remote Debugging settings panel 435  
Remove button 391  
Remove button, in Remove Markers window 119  
Remove command 51, 56, 280  
Remove Markers window 119  
    Cancel button 120  
    Done button 120  
    Markers list 119  
    Remove button 119  
Remove Object Code & Compact menu command 505  
Remove Object Code menu command 505  
Remove Toolbar Item 342  
removing  
    a component from a layout 292  
    build targets 51, 56  
    files 51  
    groups 51  
    layouts 51  
    layouts from a design 283  
    menu items from a menubar component 307

---

menus from a menubar component 307  
remote connections 365, 383  
source trees 364  
targets 51, 56  
Rename command 53, 58  
renaming  
    build targets 54, 58  
    files 53  
    groups 53  
    layouts 53  
    targets 53, 54, 58  
reopening last project used (default workspace) 442  
reopening last project used, preventing (default workspace) 443  
Repeated optimizations 401  
Replace 128, 130  
Replace All 130  
Replace All command 135  
Replace All menu command 506  
Replace and Find Next menu command 506  
Replace and Find Previous command 506  
Replace command 129, 135  
Replace menu command 506  
Replace with list pop-up 128, 130  
Replace with text box 128, 130  
replacing  
    text across multiple items 135  
    text in a single file 129  
    text, explained 125  
    text, overview 125  
Require Framework Style Includes 436  
Reset Window Toolbar command in Toolbar  
    submenu 507, 508  
resetting  
    toolbars 343  
Resize command 289  
Resize Component command 293  
resize handle 293  
Resize submenu  
    To Smallest Height command 516  
    To Smallest Width command 516  
resizing  
    components 293  
    panes in an editor window 99  
Resource File flag 397  
Restart command 199  
Restore Window command (Windows) 508  
results  
    of multi-file search 136  
Resume command 198  
Resume menu command 508  
Revert command 67  
Revert menu command 508  
reverting  
    files 67  
    preference panels 436  
    settings panels 436  
revision control 446, 476  
Right Edges button, in layout editor 287  
routine, selecting entirely 103  
Run App/Script 336  
Run command 54, 55, 198, 239  
Run menu command 434, 509  
Run To Cursor menu command 509  
running  
    a program 198  
Runtime Settings panel 394  
    Host Application For Libraries And Code  
        Resources field 427  
    options  
        Add 395  
        Change 396  
        Environment Settings 395  
        Host Application for Libraries & Code  
            Resources 395, 427  
        Program Arguments 395, 434  
        Remove 396  
        Value 396  
        Variable 396  
        Working Directory 395, 447  
    Program Arguments field (Windows) 434

**S**

Save a Copy As command 65  
Save All command 64  
Save command 64  
Save Default Window menu command 510  
Save open files before build option  
    Build Settings panel 436  
Save project entries using relative paths option  
    Target Settings panel 436  
Save Workspace As menu command 510  
Save Workspace menu command 510  
saving

---

a copy of a workspace 83  
all files 64  
file copies 65  
files 64  
information about installed plug-ins 481  
information about installed products 481  
projects 36  
workspaces 82

Script Point 251, 254  
Script Point icon 252  
(Scripts) folder 477, 502  
Scripts menu 477  
Scripts option  
    Font & Tabs panel 437

search  
    matching line beginnings and endings with regular expressions 143  
    single characters with regular expressions 141  
    using finds strings in replace strings with regular expressions 144

search engine  
    overview 25

Search menu 122, 455, 467  
Search Results window 136  
Search selection only 126, 128  
Search Status column, in Access Paths panel 391  
searching  
    choosing one character from many in regular expressions 142  
    grouping regular expressions 142  
    multiple files 134  
    multiple folders 131  
    multiple projects 132  
    multiple symbolics files 133  
    single characters with regular expressions 141  
    single files 127  
    using finds strings in replace strings with regular expressions 143  
    using regular expressions 140

Segments tab 51  
Select All menu command 510  
Select stack crawl window when task is stopped option  
    Global Settings panel 437

selecting  
    Code Completion window items 112  
    text in editor windows 102

selecting entire routines 103  
selecting lines 102

selecting multiple lines 102  
selecting rectangular portions of lines 102  
Selection position option  
    Editor Settings panel 437

selections  
    searching (text) 138  
    searching multiple windows (text) 139

Send To Back command 289  
Send To Back menu command 510

set  
    Watchpoint 260  
    watchpoint 246

Set 1, Set 2, Set 3, Set 4 372  
Set Breakpoint menu command 510  
Set Default Project command 38  
Set Default Project menu command 510  
Set Default Target menu command 511  
Set Eventpoint menu command 511  
set Watchpoint in Thread window 260  
set watchpoint in Thread window 246  
Set Watchpoint menu command 511

setting  
    browser options 147  
    eventpoints 255  
    temporary breakpoints 236

setting breakpoints in 234  
setting eventpoint  
    Symbolics window 262  
    Thread window 260

setting Watchpoint  
    Symbolics window 250  
    Variable window 247

setting watchpoint  
    Thread window 246

settings  
    Add 362, 391, 395, 397  
    Add Default 391  
    Always Search User Paths 391  
    Application 394  
    Apply button 387  
    Arguments 394  
    Auto-target Libraries 407  
    Cache subprojects 393  
    Cache symbolics between runs 407  
    Cancel button 387  
    Change 362, 391, 396, 398  
    Choose 362, 389

---

Clear 389  
Code Generation 398  
Compiler 397  
Debugger 403  
Default language entry point 407  
Details 399  
Dump internal browse information after  
    compile 393  
Edit Language 397  
Editor 401  
Environment Settings 395  
Export Panel button 387  
Extension 397  
Factory Settings button 386  
Faster Execution Speed 399  
File Mappings list 397  
File Type 397  
Flags 397  
Generate Browser Data From 393  
Host Application for Libraries & Code  
    Resources 395  
Host Flags 391  
IDE window 385  
Ignored By Make flag 397  
Import Panel button 387  
Initial directory 394  
Interpret DOS and Unix Paths 391  
Launchable flag 397  
Linker 388  
Location of Relocated Libraries and Code  
    Resources 407  
Log System Messages 408  
Name 362  
OK button 387  
Optimization Level slider 399  
Output Directory 389  
Post-linker 388  
Precompiled File flag 397  
Pre-linker 388  
Program Arguments 395  
Program entry point 407  
Remove 362, 391, 396, 398  
Require Framework Style Includes 391  
Resource File flag 397  
Revert Panel button 387  
Save button 387  
Save project entries using relative paths 389  
Smaller Code Size 399  
Source Tree list 362  
Stop at Watchpoints 408  
Stop on application launch 407  
System Paths 391  
System Paths list 391  
Target 387  
Target Name 388  
Target Settings Panels list 386  
Type 362  
Update data every  $n$  seconds 408  
Use External Debugger 394  
Use modification date caching 393  
User Paths 391  
User Paths list 391  
User specified 407  
Value 396  
Variable 396  
Working Directory 395  
settings panel 386  
settings panels  
    Access Paths 359, 389  
    Build Extras 392, 494  
    Custom Keywords 401  
    Debugger Settings 218, 406  
    File Mappings 396  
    Global Optimizations 398  
    Other Executables 403  
    Remote Debugging 408  
    reverting 436  
    Runtime Settings 394  
    Source Trees 361  
    Target Settings 388  
setup  
    code completion 107  
Shielded Folders panel  
    options  
        Find and compare operations 425  
        Project operations 434  
        Regular Expression 435  
Shielded Folders preference panel 359  
    options  
        Add 360  
        Change 361  
        Find and compare operations 360  
        Project operations 360  
        Regular Expression 360  
        Remove 361  
        Shielded folder list 360  
Shift Right menu command 511  
shortcut conventions 20

---

---

Show all locals option  
    Display Settings panel 438

Show All Wires button, in layout editor 288, 479

Show Breakpoints menu command 486, 511

Show Classes 162

Show Classes pane 166

Show Floating Toolbar command 498

Show Floating Toolbar command in Toolbar submenu 511, 512

Show Incoming Wires button, in layout editor 288, 479

Show Inherited 157

Show Main Toolbar command 498

Show message after building up-to-date project option  
    Build Settings panel 438

Show Options button, in object inspector 296

Show Outgoing Wires button, in layout editor 288, 479

Show private 158

Show protected 158

Show public 158

Show tasks in separate windows option  
    Display Settings panel 438

Show the component palette when opening a form option  
    Layout Editor panel 438

Show the object inspector when opening a form option  
    Layout Editor panel 438

Show Types menu command 512

Show values as decimal instead of hex option  
    Display Settings panel 439

Show variable location option  
    Display Settings panel 439

Show variable types option  
    Display Settings panel 439

Show variable values in source code option  
    Display Settings panel 439

Show Window Toolbar command 498

Show Window Toolbar command in Toolbar submenu 512

showing  
    classes pane 162, 163

shrinking panes, in browser 160

Single Class Hierarchy Window 156

single files, searching 127

single-class hierarchical window  
    opening 171

Single-Class Hierarchy window 170

difference from Multi-Class Hierarchy window 170

single-file Find and Replace window 127

single-file Find window 126

size  
    setting default for unbounded arrays 419

Size option  
    Font & Tabs panel 439

Skip Point 251, 254

Skip Point icon 252

Snap To Grid button, in layout editor 288

Snap To Grid command 289

Snap To Grid menu command 512

software  
    development process cycle 21

Solaris  
    modifier key mappings 20

Sort Alphabetical 161, 163

Sort function popup option  
    Editor Settings panel 440

Sort functions by method name in symbolics window option  
    Display Settings panel 439

Sort Hierarchical 161, 163

Sort Order button  
    in Files view of Project window 48

sorting  
    classes list 163  
    Functions list pop-up (alphabetically) 117

Sound Point 251, 254

Sound Point icon 252

Sound Point, Speak Message 254

source code 234  
    editing 101  
    going to a particular line 118  
    locating 115

source code, navigating 115

source file  
    adding markers to 120

source files  
    removing all markers from 121  
    removing markers from 120

Source pane 165, 250, 263  
    in Class Browser window 157  
    in Symbols window 176

source trees  
    adding 363

---

changing 364  
removing 364

Source Trees panel  
options  
    Add 362  
    Change 362  
    Choose 362  
    Name 362  
    Remove 362  
    Source Tree list 362  
    Type 362, 441

Source Trees preference panel 361  
    Absolute Path option 441  
    Environment Variable option 442  
    Registry Key option 442

Source Trees settings panel 361

Stack Crawl window. *See* Thread window. 245, 259

Stack Editor Windows menu command 512  
starting  
    debugger 195

static  
    icon for 163

stationery  
    creating for projects 40  
    creating projects 33  
    custom 40

status  
    of breakpoints 232

Status area  
    in Class Browser window 157

status area 165

Step Into command 196

Step Out command 196

Step Over command 197

Step Over menu command 512

Stop at end of file 127, 128

Stop at Watchpoints option  
    Debugger Settings panel 440

Stop Build menu command 513

Stop command 197, 513

Stop On Application Launch option  
    Debugger Settings panel 440

Straight Line 170

strategies  
    for build targets 42  
    for projects 42  
    for subprojects 42

Strength Reduction 400

Strings option  
    Text Colors panel 440

structure  
    of documentation 18

submenus  
    Align 482, 483

subproject, defined 41

subprojects  
    creating 41  
    opening 42  
    strategies for 42

Success option  
    Build Settings panel 440

summation, of two variables 212

suspended process 245, 259

Switch To Monitor menu command 513

symbol definitions 121, 122

symbol definitions, looking up 122

Symbol hint 202

symbol hint  
    toggling 203  
    turning off 203  
    turning on 203  
    using 203

symbol implementations  
    viewing all 175

symbol-editing shortcuts 105

symbolics file, defined 192

symbolics files  
    choosing a debugger for 381  
    searching (multiple) 133

symbolics information 249, 261

Symbolics menu command 513

Symbolics option  
    IDE Extras panel 440

Symbolics window 205, 239, 249, 261  
    clear eventpoint 263  
    opening 206  
    set eventpoint 262  
    set Watchpoint 250  
    using the Executables pane 206, 207  
    using the Files pane 207  
    using the Functions pane 207

Symbolics Window menu command 513

symbols  
    shortcuts for editing 105

- 
- viewing all implementations 175
  - Symbols list
    - in Browser Contents window 172
  - Symbols pane 175
  - Symbols pop-up 172
  - Symbols window 174
    - Source pane 176
    - Symbols pane 175
    - toolbar 175
  - symbols window 175
  - Synchronize Modification Dates command 50
  - Synchronize Modification Dates menu command 513
  - Synchronize Modification Dates toolbar button 46
  - System Paths list 391
    - Framework column 392
    - Recursive Search column 392
    - Search Status column 391
  - System Paths option
    - Access Paths panel 440
- T**
- Tab indents selection option
    - Font & Tabs panel 441
  - Tab Inserts Spaces option
    - Font & Tabs panel 441
  - Tab Size option
    - Font & Tabs panel 441
  - Target column
    - in Files view of Project window 48
  - target management 51
  - Target Name option
    - Target Settings panel 441
  - Target section, of Target Settings panels 387
  - target settings
    - Add 362, 391, 395, 397
    - Add Default 391
    - Always Search User Paths 391
    - Application 394
    - Apply button 387
    - Arguments 394
    - Auto-target Libraries 407
    - Cache subprojects 393
    - Cache symbolics between runs 407
    - Cancel button 387
    - Change 362, 391, 396, 398
    - Choose 362, 389
    - Clear 389
  - Compiler 397
  - Connection pop-up menu 408
  - Default language entry point 407
  - Details 399
  - Dump internal browse information after compile 393
  - Edit Language 397
  - Environment Settings 395
  - Export Panel button 387
  - Extension 397
  - Factory Settings button 386
  - Faster Execution Speed 399
  - File Mappings list 397
  - File Type 397
  - Flags 397
    - for IDE 385
  - Generate Browser Data From 393
  - Host Application for Libraries & Code
    - Resources 395
  - Host Flags 391
  - Ignored By Make flag 397
  - Import Panel button 387
  - Initial directory 394
  - Interpret DOS and Unix Paths 391
  - Launchable flag 397
  - Linker 388
  - Location of Relocated Libraries and Code
    - Resources 407
  - Log System Messages 408
  - Name 362
  - OK button 387
  - Optimization Level slider 399
  - Output Directory 389
  - Post-linker 388
  - Precompiled File flag 397
  - Pre-linker 388
  - Program Arguments 395
  - Program entry point 407
  - Remove 362, 391, 396, 398
  - Require Framework Style Includes 391
  - Resource File flag 397
  - Revert Panel button 387
  - Save button 387
  - Save project entries using relative paths 389
  - Smaller Code Size 399
  - Source Tree list 362
  - Source Trees 361
  - Stop at Watchpoints 408
  - Stop on application launch 407

---

System Paths 391  
System Paths list 391  
Target Name 388  
Target Settings Panels list 386  
Type 362  
Update data every *n* seconds 408  
Use External Debugger 394  
Use modification date caching 393  
User Paths 391  
User Paths list 391  
User specified 407  
Value 396  
Variable 396  
Working Directory 395  
Target Settings command 513  
Target Settings panel 58, 388  
options  
    Choose 389  
    Clear 389  
    Linker 388, 431  
    Output Directory 389, 433  
    Post-linker 388, 433  
    Pre-linker 388, 434  
    Save project entries using relative paths 389, 436  
    Target Name 388, 441  
target settings panel 386  
target settings panels  
    Access Paths 389  
    Build Extras 392, 494  
    Custom Keywords 401  
    Debugger Settings 218, 406  
    File Mappings 396  
    Global Optimizations 398  
    Other Executables 403  
    Remote Debugging 408  
    Runtime Settings 394  
    Target Settings 388  
Target Settings Panels list 386  
Target Settings toolbar button 46  
Target Settings window 385, 386  
    Apply button 387  
    Cancel button 387  
    Export Panel button 387  
    Factory Settings button 386  
    Import Panel button 387  
    OK button 387  
    opening 387  
    Revert Panel button 387  
Save button 387  
Target Settings Panels list 386  
targets 31  
    configuring 58  
    creating 56  
    files 51  
    managing 56  
    moving 52  
    removing 51, 56  
    renaming 53, 54, 58  
    setting default 57  
    strategies for 42  
Targets tab 58  
Targets view 37, 50, 52, 56  
tasks  
    activating automatic code completion 107  
    adding a component to a layout 292  
    adding a constant to a variable 212  
    adding a keyword to a keyword set 402  
    adding an executable file 405  
    adding components to a component catalog 302  
    adding expressions (Expressions window) 211  
    adding markers to a source file 120  
    adding panes to an editor window 98  
    adding remote connections 381  
    adding source trees 363  
    adding subprojects to a project 41  
    alphabetizing Functions list pop-up order 117  
    attaching the debugger to a process 210  
    balancing punctuation 105  
    breakpoints 233  
    changing an executable file 405  
    changing line views in a hierarchical window 170  
    changing register data views 217  
    changing register values 217  
    changing remote connections 382  
    changing source trees 364  
    choosing a default project 38  
    clearing all breakpoints 235  
    clearing breakpoints in source panes 234  
    clearing breakpoints in the Breakpoints  
        window 235  
    closing a docked window 79  
    closing a workspace 84  
    closing projects 39  
    collapsing a docked window 78  
    collapsing browser panes 160  
    collapsing the editor window toolbar 94  
    completing code for data members 113

---

---

completing code for parameter lists 114  
completing the Create Wire wizard 315  
creating a console application 86  
creating a menu in a menubar component 306  
creating a menu item in a menubar component 306  
creating a new class 162, 177, 178  
creating a new data member 164, 185, 186  
creating a new member function 163, 182, 183  
creating a wire by drawing it 311  
creating a wire by using a contextual menu 312  
creating a wire by using a menu 312  
creating custom project stationery 40  
creating empty projects 35  
creating new projects from makefiles 33  
creating new projects using project stationery 33  
deactivating automatic code completion 109  
deleting a wire by using the layout editor 314  
deleting wires by using the object inspector 319  
docking a window by using a contextual menu 72  
docking a window by using drag and drop 72  
docking windows of the same kind 73  
expanding a docked window 78  
expanding browser panes 160  
expanding the editor window toolbar 94  
exporting projects to XML files 38  
floating a window 75  
for managing files 61  
generating project link maps 274  
going to a particular line 118  
hiding the classes pane 162  
importing projects saved as XML files 39  
indenting text blocks 104  
inserting a reference template 123  
looking up symbol definitions 122  
making a summation of two variables 212  
making a window an MDI child 76  
modifying a wire by using the layout editor 313  
modifying wires by using the object inspector 318  
moving a component in a layout 293  
moving a docked window 78  
navigating browser data 151  
navigating Code Completion window 111  
navigating to marker 120  
opening a layout 289  
opening a recent workspace 84  
opening a single-class hierarchical window 171  
opening a workspace 83  
opening an Array window 223  
opening projects 35

opening projects created on other hosts 36  
opening registers in a separate Registers window 218  
opening subprojects 42  
opening the Breakpoints window 233  
opening the component catalog window 301  
opening the component palette 291  
opening the Expressions window 211  
opening the Global Variables window 214  
opening the IDE Preferences window 351  
opening the Log window 219  
opening the object inspector 296  
opening the Processes window 209, 210  
opening the Registers window 216  
opening the Symbolics window 206  
opening the symbols window 175  
opening the Target Settings window 387  
opening the Variable window 220  
overstriking text (Windows) 103  
printing class hierarchies 169  
printing projects 37  
removing a component from a layout 292  
removing a keyword from a keyword set 403  
removing a layout from a design 283  
removing a marker from a source file 120  
removing a menu from a menubar component 307  
removing a menu item from a menubar component 307  
removing all markers from a source file 121  
removing an executable file 406  
removing components from a catalog 304  
removing panes from an editor window 99  
removing remote connections 365, 383  
removing source trees 364  
re-ordering wires by using the object inspector 318  
replacing text across multiple items 135  
replacing text in a single file 129  
resizing a component in a layout 293  
resizing panes in an editor window 99  
running a program 198  
saving a copy of a workspace 83  
saving a workspace 82  
saving projects 36  
searching a single file 127  
searching across multiple files 134  
searching across multiple folders 131  
searching across multiple projects 132  
searching across multiple symbolics files 133  
searching with a text selection 138

---

searching with a text selection across multiple window 139  
selecting entire routines 103  
selecting item in Code Completion window 112  
selecting lines 102  
selecting multiple lines 102  
selecting rectangular portions of lines 102  
selecting text in editor windows 102  
setting breakpoints in source panes 234  
setting conditional breakpoints 236  
setting temporary breakpoints 236  
showing the classes pane 162  
sorting the classes list 163  
sorting the Wires tab by using a contextual menu 320  
suppressing dockable windows 76  
toggling automatic punctuation balancing 106  
toggling the symbol hint 203  
triggering code completion by keyboard 108  
triggering code completion from IDE menu bar 108  
undocking a window 74  
unfloating a window 75  
unindenting text blocks 104  
using an external editor on the Macintosh 357  
using contextual menus 204  
using the browser contextual menu 152  
using the default workspace 82  
using the document settings pop-up 96  
using the Executables pane (Symbolics window) 206, 207  
using the Files pane (Symbolics window) 207  
using the Find Previous command 137  
using the Functions list pop-up 116  
using the Functions pane (Symbolics window) 207  
using the Interfaces list pop-up 116  
using the Process pane 209  
using the symbol hint 203  
using the Version Control System (VCS) pop-up 97  
using virtual space 103  
viewing a file path 49  
viewing browser data by contents 173  
viewing browser data by inheritance 168  
viewing class data from hierarchy windows 159  
viewing registers 216  
viewing wires by using the object inspector 318  
techniques  
for advanced debugging 201

Templates option 372  
templates, reference (Macintosh) 123  
temporary breakpoints  
    setting 236  
text  
    find by selecting 137  
    finding 125  
    finding and replacing, explained 125  
    overstriking (Windows) 103  
    replacing 125  
    searching with a selection 138  
    searching with a selection across multiple window 139  
text blocks, indenting 104  
text blocks, unindenting 104  
Text Colors panel  
    options  
        Activate Browser Coloring 426  
        Activate Syntax Coloring 426, 429, 440  
        Foreground 426  
        Keywords 429  
        Strings 440  
Text Colors preference panel  
    options  
        Activate Browser Coloring 372  
        Activate Syntax Coloring 372  
        Background 372  
        Classes 372  
        Comments 372  
        Constants 372  
        Edit 372  
        Enums 372  
        Foreground 372  
        Functions 372  
        Globals 372  
        Keywords 372  
        Macros 372  
        Other 373  
        Set 1, Set 2, Set 3, Set 4 372  
        Strings 372  
        Templates 372  
        TypeDefs 373  
text editing area, in editor window 98  
text manipulation 101  
text-selection Find 137  
THINK Reference 121, 123, 425  
third-party editor support 444  
third-party text editors

---

Emacs 429, 430  
Thread window 245, 247, 259  
  clear eventpoint 261  
  clear Watchpoint 261  
  clear watchpoint 246  
  enable and disable Watchpoint 247  
  set eventpoint 260  
  set Watchpoint 260  
  set watchpoint 246  
  Variables pane 246, 260  
threading in IDE 417  
\_\_throw() 483  
Tile Editor Windows command 514  
Tile Editor Windows Vertically command 514  
Tile Horizontally command 514  
Tile Vertically command 514  
To Smallest Height command in Resize submenu 516  
To Smallest Width command in Resize submenu 516  
toggling  
  symbol hint 203  
toolbar  
  collapsing in editor window 94  
  expanding in editor window 94  
Toolbar (Editor Window) Elements  
  Document Settings 341  
  File Dirty Indicator 341  
  File Path field 341  
  Functions 341  
  Header Files 341  
  Markers 341  
  Version Control Menus 341  
toolbar buttons  
  Browser Contents 156  
  Class Hierarchy 156  
  Go Back 156  
  Go Forward 156  
  Make 46  
  Single Class Hierarchy Window 156  
  Synchronize Modification Dates 46  
  Target Settings 46  
Toolbar Items 327, 341  
Toolbar submenu  
  Anchor Floating Toolbar command 482  
  Clear Floating Toolbar command 486  
  Clear Main Toolbar command 486  
  Clear Window Toolbar command 487  
  Hide Floating Toolbar command 498  
  Hide Main Toolbar command 498  
  Reset Window Toolbar command 507, 508  
  Show Floating Toolbar command 498, 511, 512  
  Show Main Toolbar command 498  
  Show Window Toolbar command 512  
Toolbars  
  Add element 340, 341  
  Clear Elements 342  
  Customize 338  
  Elements 338, 340  
  Icons 341  
  Instances of 339  
  Main (floating) 339  
  Modify 340  
  Project and Window 339  
  Remove element 342  
  Remove single element 340  
  Toolbar Items tab 340  
  Types 339  
toolbars  
  editor 93  
  for Symbols window 175  
  resetting 343  
Toolbars menu command 515  
tools  
  browser 25  
  build system 25  
  debugger 26  
  editor 25  
  project manager 25  
  RAD 26  
  search engine 25  
Tools menu 476  
  icon 477  
ToolServer menu 445  
ToolServer Worksheet command 516  
ToolTip 340  
Top Edges button, in layout editor 287  
touch  
  defined 48  
Touch column 54, 55  
  in Files view of Project window 48  
Touch command 54  
touching  
  all files 54  
  all groups 54  
  files 54  
  groups 54  
triggering

- 
- code completion by keyboard 108
  - code completion from IDE menu bar 108
  - turning off
    - symbol hint 203
  - turning on
    - symbol hint 203
  - Type option
    - Source Trees panel 441
  - Type pop-up menu
    - Absolute Path option 441
    - Environment Variable option 442
    - Registry Key option 442
  - TypeDefs option 373
  - types
    - of documentation 19
  - U**
    - Unanchor Floating Toolbar command 516
    - Unapply Difference command 516
    - unbounded arrays
      - setting default size for viewing 419
    - Uncought Exceptions Only command in Java
      - Exceptions submenu 517
    - Undo command 517
    - undocking windows 74
    - unfloating a window 75
    - Ungroup button, in layout editor 287
    - Ungroup command 517
    - unindenting
      - text blocks 104
    - Untouch command 55
    - untouching
      - all files 55
      - all groups 55
      - files 55
      - groups 55
    - Up
      - in Find and Replace window 128
      - in Find window 127
    - Update Data Every n Seconds option
      - Debugger Settings panel 442
    - Use Concurrent Compiles option
      - Concurrent Compiles panel 435, 442
    - Use Debugging Monitor option
      - Windowing panel 442
    - Use Default Workspace option
      - IDE Extras panel 442
    - Use External Debugger option
      - Build Extras panel 443
    - Use External Editor option
      - IDE Extras panel 443
    - Use Local Project Data Storage option
      - Build Settings panel 443
    - Use modification date caching option
      - Build Extras panel 443
    - Use Multiple Document Interface option 69
      - IDE Extras panel 444
    - Use multiple undo option 517
      - Editor Settings panel 444
    - Use Script menu option
      - IDE Extras panel 444
    - Use Scripts Menu checkbox
      - of IDE Extras preference panel 477
    - Use Third Party Editor option
      - of IDE Extras preference panel 444
    - Use ToolServer Menu checkbox
      - of IDE Extras preference panel 476
    - Use ToolServer menu option
      - IDE Extras panel 445
    - User Paths list 391
      - Framework column 392
      - Recursive Search column 392
      - Search Status column 391
    - User Paths option
      - Access Paths panel 445
    - User Specified option
      - Concurrent Compiles panel 445
    - User specified option 407
    - using
      - breakpoints 231
      - document settings pop-up 96
      - Eventpoints 251
      - Executables pane (Symbolics window) 206, 207
      - Files pane (Symbolics window) 207
      - Find Previous command 137
      - Functions list pop-up 116
      - Functions pane (Symbolics window) 207
      - Interfaces list pop-up 116
      - symbol hint 203
      - Version Control System (VCS) pop-up 97
      - virtual space 103
      - Watchpoints 239
    - using workspaces 81

## V

### Variable

    setting a Watchpoint in the Variable window 247

### variable

    adding a constant to 212

### Variable Values Change option

    Display Settings panel 446

### Variable window 220, 239, 247, 248

    clear Watchpoint 249

    opening 220

    set Watchpoint 247

### variables

    making a summation of 212

    symbol hint 202

### Variables pane 246, 260

### Variables pane in the Thread window 246, 260

### VCS Commands option

    Editor Settings panel 446

### VCS list pop-up 157

### VCS menu 165, 446, 476

    icon 476

### VCS pop-up 96

    using 97

### Vectorization 401

### version control 446, 476

### Version Control Settings command 517

### Version Control System (VCS) pop-up 96

### Version Control System menu. *See* VCS menu. 476

### Vertical Center button, in layout editor 287

### Vertical Center command in Align submenu 508, 515, 516, 517

### View Array menu command 517

### View as implementor 158

### View as subclass 158

### View As Unsigned Decimal menu command 517, 518, 519

### View as user 158

### View Disassembly menu command 519

### View Memory As command 519

### View Memory command 519

### View Mixed menu command 520

### View Source menu command 520

### View Variable menu command 520

### viewing

    all symbol implementations 175

    breakpoints 233

browser data by contents 173

browser data by inheritance 168

file paths 49

registers 216

viewing installed plug-ins 481

viewing installed products 481

views

    in project window 47

virtual

    icon for 163

virtual space, using 103

## W

### Watchpoint 239

    Conditional 248

    enabling and disabling in the Thread window 247

    setting in the Symbolics window 250

watchpoint

    clearing in the Thread window 246

    setting in the Thread window 246

Watchpoint Indicator option

    Display Settings panel 446

watchpoint, clear 246

watchpoint, set 246

Watchpoints

    clear 249

    purpose of 239

    using 239

Watchpoints menu command 520

Watchpoints window 240, 242

Watchpoints Window menu command 520

what is

    a debugger 191

    a symbolics file 192

    advanced debugging 202

Window

    Customize IDE Commands 344

window

    Breakpoints 259

    Memory 239

    Symbolics 239

    Variable 239, 247

    Watchpoints 240

Window Follows Insertion Point option 446

Window Menu

    Restore Window command (Windows) 508

Window menu 454, 462, 475

- 
- Window position and size option
    - Editor Settings panel 446
  - window types
    - docked 70
    - floating 70
    - MDI child 70
  - Windowing panel
    - options
      - Hide non-debugging windows 427
      - Minimize non-debugging windows 432
      - Monitor for debugging 432
      - Move open windows to debugging monitor when debugging starts 432
      - Open windows on debugging monitor during debugging 433
      - Use Debugging Monitor 442
  - Windowing preference panel 377
    - options
      - Close non-debugging windows 378
      - Collapse non-debugging windows 378
      - Do nothing 378
      - Do nothing to project windows 378
      - Hide non-debugging windows 378
      - Minimize non-debugging windows 378
      - Monitor for debugging 378
      - Move open windows to debugging monitor when debugging starts 378
      - Open windows on debugging monitor during debugging 378
      - Use Debugging Monitor 378
  - Windows
    - creating files 61
  - windows
    - Array 221
    - Breakpoints 231, 256
    - Browser Contents 172
    - Class Browser 155
    - Code Completion 110
    - dock bars in dockable windows 76
    - dockable 69
      - dockable, about 69
      - dockable, turning off 76
      - dockable, working with 71
      - docking the same kind of 73
      - docking with a contextual menu 72
      - docking with drag and drop 72
    - editor 91, 252
    - editor, other 97
    - Expressions 210
  - Find (single-file) 126
  - Find and Replace (multiple-file) 129
  - Find and Replace (single-file) 127
  - floating 75
  - Global Variables 213
  - hierarchy 168
  - IDE Preferences 349
  - Log 218
    - making into an MDI child 76
  - Memory 223
  - New C++ Class 179
  - New C++ Data Member 187
  - New C++ Member Function 184
  - Processes 208
  - project window 45
  - Registers 214
  - Remove Markers 119
  - Search results 136
  - Symbolics 205, 249, 261
  - Target Settings 385
  - Thread 245, 247, 259
  - undocking 74
  - unfloating 75
  - Variable 247, 248
  - variable 220
  - Watchpoints 242
- Windows menu layout 451
- WinHelp (Windows) 121
- wire, defined 309
- wires
  - creating with contextual menus 312
  - creating with menus 312
  - deleting with the layout editor 314
  - drawing 311
  - modifying with the layout editor 313
  - sorting the Wires tab in the object inspector 320
  - using the object inspector to delete 319
  - using the object inspector to modify 318
  - using the object inspector to re-order 318
  - using the object inspector to view 318
- Wires tab, in object inspector 296
- wires, inspecting 317
- Wizards
  - Browser 177
- wizards
  - Create Wire, using 315
  - New Class 162, 177, 178
  - New Data Member 164, 185, 186
  - New Member Function 182, 183

---

New Member Functions 163  
working  
    with IDE preferences 349  
    with IDE target settings 385  
Working Directory option  
    Runtime Settings panel 447  
working with 71  
    browser 147  
    class browser windows 155  
    class hierarchy windows 167  
working with dockable windows 71  
working with files 61  
working with object wiring 310  
working with projects 29  
workspace, defined 81  
workspaces 81  
    closing 84  
    opening 83  
    opening recent 84  
    saving 82  
    saving copies of 83  
    using 81  
    using default 82  
Workspaces option  
    IDE Extras panel 447  
workspaces, about 81

## X

XML  
    exporting projects 38  
    importing projects 39

## Z

Zoom Window menu command 520  
Zoom windows to full screen option  
    IDE Extras panel 447